Certifying homological algorithms to study biomedical images*

María Poza López de Echazarreta

Supervisors: Dr. César Domínguez Pérez
Dr. Julio Rubio García

Department of Mathematics and Computer Science
University of La Rioja
Spain

June 14, 2013

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

Table of Contents

Table of Contents

The Formath project

- European project
- Formath: Formalization of Mathematics

The Formath project

- European project
- Formath: Formalization of Mathematics
  - University of Gothenburg
  - Radboud University Nijmegen
  - INRIA
  - Universidad de La Rioja

The Formath project

- European project
- Formath: Formalization of Mathematics
  - University of Gothenburg
  - Radboud University Nijmegen
  - INRIA
  - **Universidad de La Rioja**
    - Integration of Theorem Prover Tools
    - Representation of Simplicial Complexes
    - Certified Computation of Homology Groups
    - The Basic Perturbation Lemma
    - Applications to Medical Imagery

The Formath project

- European project
- Formath: Formalization of Mathematics
  - University of Gothenburg
  - Radboud University Nijmegen
  - INRIA
  - **Universidad de La Rioja**
    - Integration of Theorem Prover Tools
    - Representation of Simplicial Complexes
    - Certified Computation of Homology Groups
    - The Basic Perturbation Lemma
    - Applications to Medical Imagery

The Formath project

- European project
- Formath: Formalization of Mathematics
  - University of Gothenburg
  - Radboud University Nijmegen
  - INRIA
  - **Universidad de La Rioja**
    - Integration of Theorem Prover Tools
    - Representation of Simplicial Complexes
    - Certified Computation of Homology Groups
    - The Basic Perturbation Lemma
    - Applications to Medical Imagery

Verification and Theorem prover tools

$$\begin{tikzcd}[column sep=small] & \text{Program Verification} \arrow[dl] \arrow[dr] & \\ \text{Automated Verification (Model checking, SMT...)} & & \text{Interactive Proof Assistants} \end{tikzcd}$$

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 Color Theorem
    - Kepler Conjecture
    - Feit-Thompsom Theorem (Odd Order Theorem)
    - ...
  - Software and Hardware verification:
    - C compiler
    - AMD5K86 microprocessor
    - ...

Coq/SSReflect

Program Verification

Automated Verification

Interactive Proof Assistants

Coq

Coq/SSReflect

Program Verification

Automated Verification

Interactive Proof Assistants

↓

Coq

- Coq:
  - Based on Calculus of Inductive Constructions
  - Interesting feature: program extraction from a constructive proof

Y. Bertot and P. Castéran. Interactive Theorem Proving and Program Development. Coq'Art: The Calculus of Inductive Constructions Series. Texts in Theoretical Computer Science. An EATCS Series, 2004.

COQ/SSReflect

Program Verification

Automated Verification

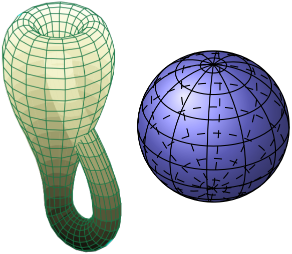
Interactive Proof Assistants

COQ

- COQ:
  - Based on Calculus of Inductive Constructions
  - Interesting feature: program extraction from a constructive proof
- SSReflect:
  - Extension of COQ
  - Developed while formalizing the Four Color Theorem by G. Gonthier
  - Used in the formalization of Feit-Thompson Theorem

Algebraic Topology

Topological Spaces $\longrightarrow$ Invariant Groups



H_0
H_1
$\dots$

Algebraic Topology

Topological Spaces $\longrightarrow$ Invariant Groups

- Kenzo
  - Computer Algebra system devoted to Algebraic Topology implemented in *Common Lisp*
  - Homology groups which have not been obtained by other means
- fKenzo
  - *fKenzo*: graphical user interface for the system *Kenzo*
  - It is not necessary to be an expert in Algebraic Topology or Common Lisp to use it
  - Provides new functionalities to Kenzo such as the homology computation for digital images

Algebraic Topology

Goal

Formalize the analysis of monochromatic digital images

Algebraic Topology

Goal

Formalize the analysis of monochromatic digital images

Context

To deal with biomedical images:

- Reliability
- Efficiency

Algebraic Topology

Goal

Formalize the analysis of monochromatic digital images

Context

To deal with biomedical images:

- Reliability
- Efficiency

Our approach

Formalize a technique to reduce the size of the information about a biomedical image (preserving its homological properties)

Goal

$$\left.\begin{array}{l}\text{Verification}\\ \text{Algebraic Topology}\end{array}\right\} + \left\{\ \text{Digital images}\ \right\} \rightarrow \text{Formalization of Digital Topology}$$

Goal

$$\left.\begin{array}{l}\text{Verification}\\ \text{Algebraic Topology}\end{array}\right\} + \left\{\ \text{Digital images}\ \right\} \rightarrow \text{Formalization of Digital Topology}$$

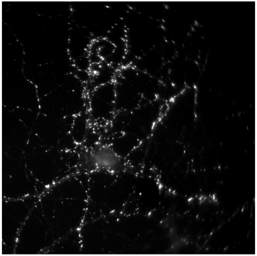
The 95% of this thesis is devoted to formal verification

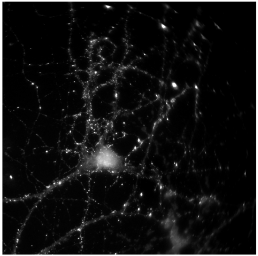
Table of Contents

Biomedical Problem: counting synapses

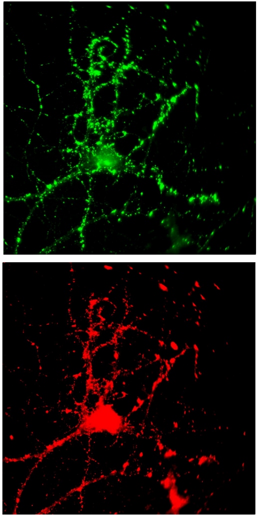
- *Synapses* are the points of connection between neurons
- *Relevance*: Computational capabilities of the brain
- Procedures to modify the synaptic density may be an important asset in the treatment of neurological diseases, such as Alzheimer

Manual process to count synapses

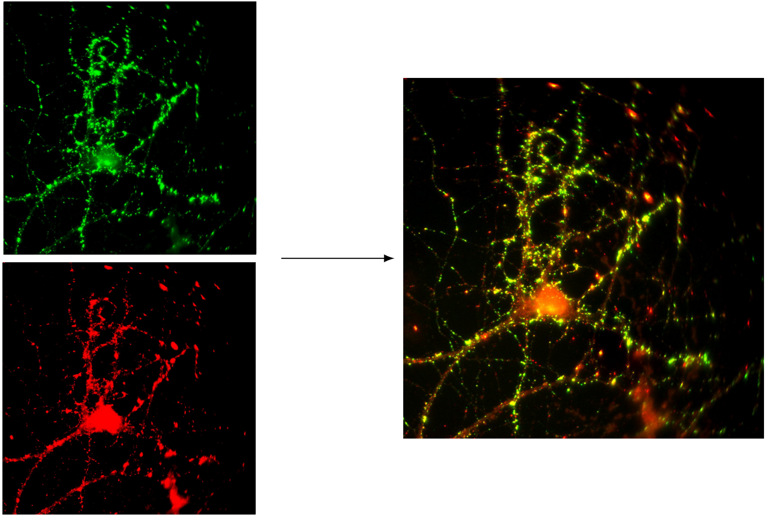




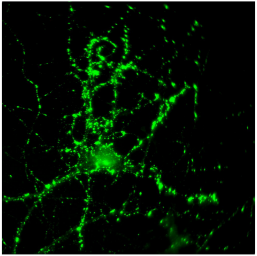
Manual process to count synapses

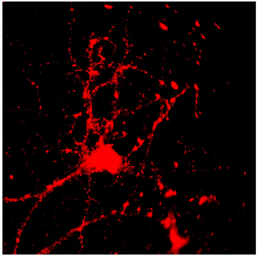


Manual process to count synapses

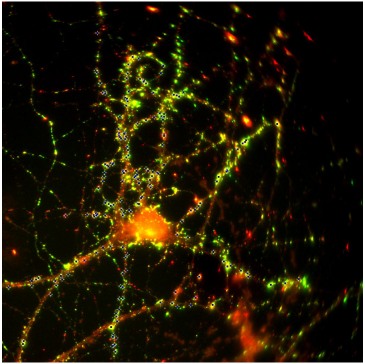


Manual process to count synapses

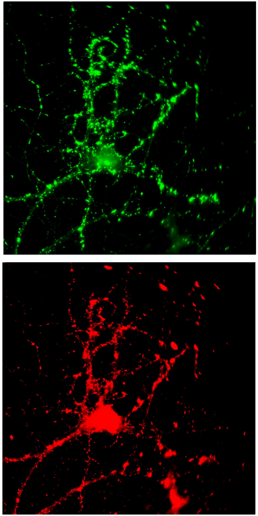


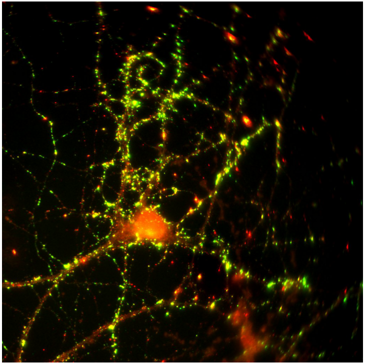


Count the synapses manually

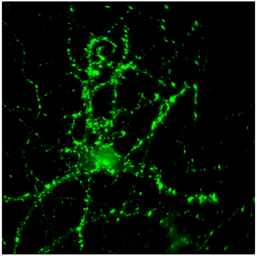


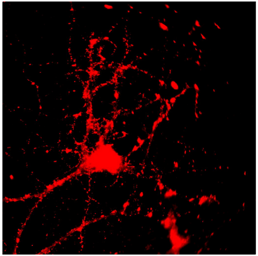
Semiautomatic process to count synapses

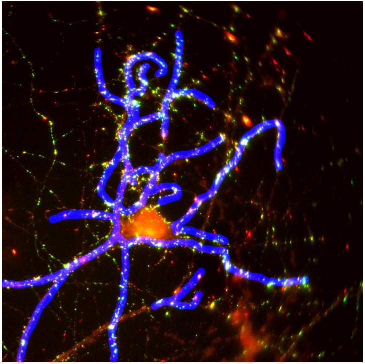




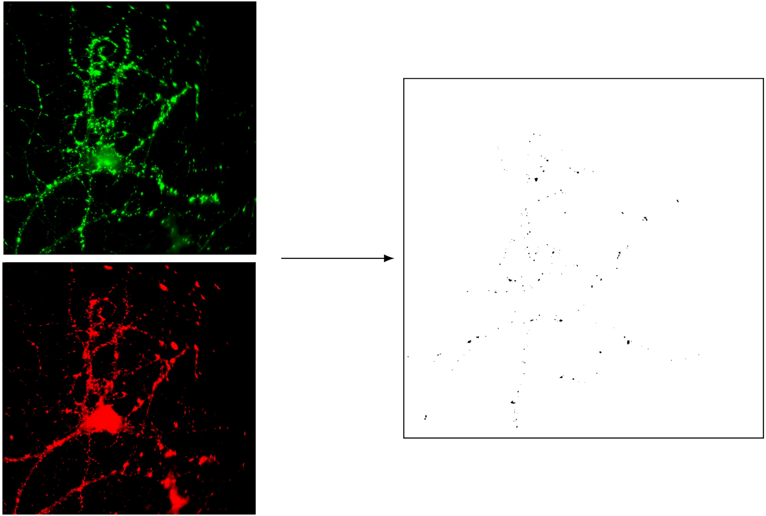
Semiautomatic process to count synapses







Semiautomatic process to count synapses



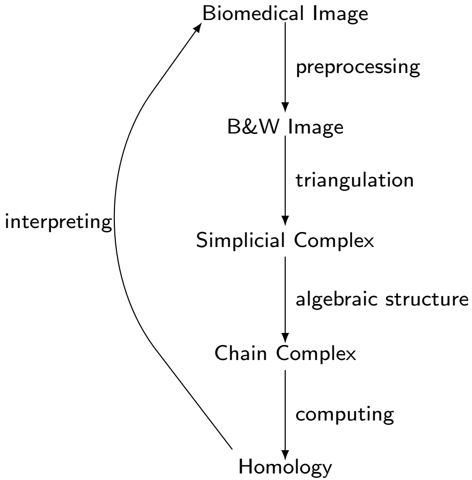
The method

$$
\begin{array}{c}
\text{Biomedical Image} \\
\Big\downarrow \text{preprocessing} \\
\text{B\&W Image}
\end{array}
$$

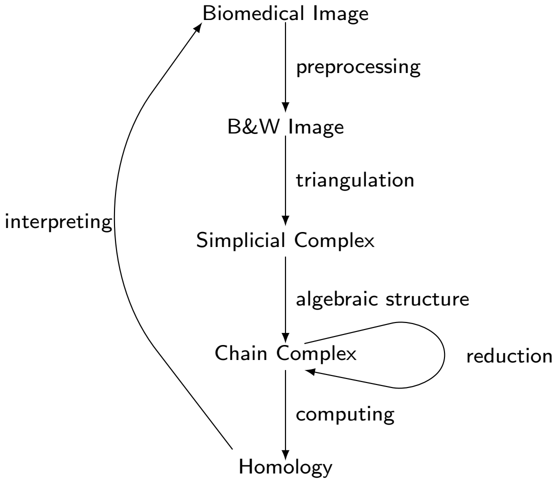
The method

$$
\begin{tikzcd}
\text{Biomedical Image} \arrow[d, "\text{preprocessing}"] \\
\text{B\&W Image} \arrow[d, "\text{triangulation}"] \\
\text{Simplicial Complex} \arrow[d, "\text{algebraic structure}"] \\
\text{Chain Complex} \arrow[d, "\text{computing}"] \\
\text{Homology}
\end{tikzcd}
$$

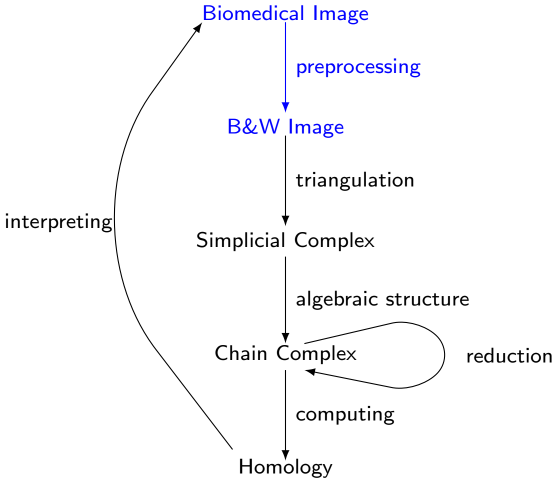
The method


Biomedical Image
preprocessing
B&W Image
triangulation
Simplicial Complex
algebraic structure
Chain Complex
computing
Homology
interpreting

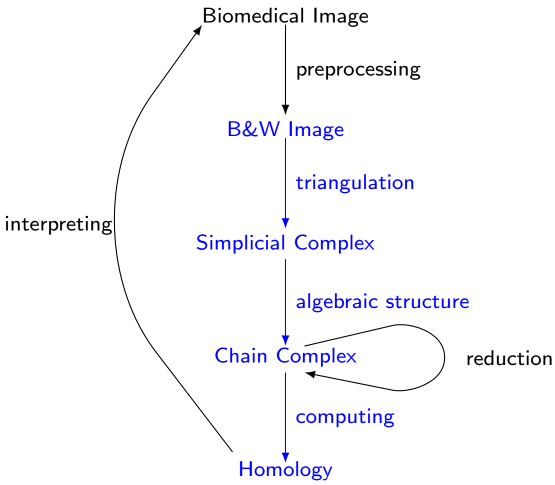
The method


Biomedical Image
preprocessing
B&W Image
triangulation
Simplicial Complex
algebraic structure
Chain Complex
reduction
computing
Homology
interpreting

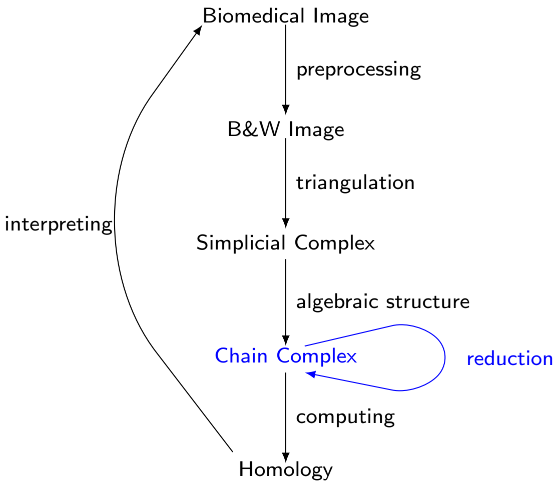
The method


Biomedical Image
preprocessing
B&W Image
triangulation
Simplicial Complex
algebraic structure
Chain Complex
reduction
computing
Homology
interpreting

The method

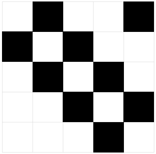

Biomedical Image
preprocessing
B&W Image
triangulation
Simplicial Complex
algebraic structure
Chain Complex
reduction
computing
Homology
interpreting

The method

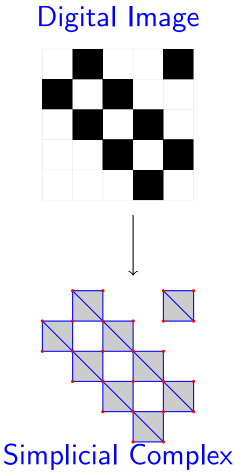

Biomedical Image
preprocessing
B&W Image
triangulation
Simplicial Complex
algebraic structure
Chain Complex
reduction
computing
Homology
interpreting

Digital Algebraic Topology

Digital Image

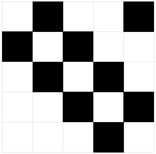


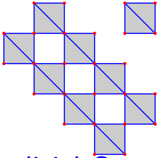
Digital Algebraic Topology



Digital Algebraic Topology

Digital Image





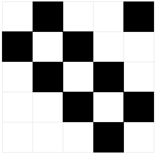
Simplicial Complex

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

Chain Complex

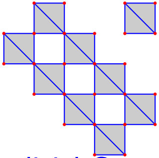
Digital Algebraic Topology

Digital Image



Homology Groups

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



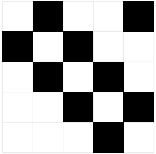
Simplicial Complex

C_0 = vertices
C_1 = edges
C_2 = triangles

Chain Complex

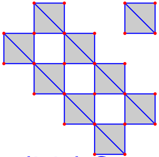
Digital Algebraic Topology

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

Simplicial Complex

Definition

Let V be a set, called the vertex set, a *simplex* over V is any finite subset of V

Simplicial Complex

Definition

Let V be a set, called the vertex set, a *simplex* over V is any finite subset of V

Definition

An *(abstract) simplicial complex* over V is a set of simplices C over V satisfying the property:

$$\forall \alpha \in C, \ \textit{si} \ \beta \subseteq \alpha \Rightarrow \beta \in C$$

Simplicial Complex

Definition

Let V be a set, called the vertex set, a *simplex* over V is any finite subset of V

Definition

An *(abstract) simplicial complex* over V is a set of simplices C over V satisfying the property:

$$\forall \alpha \in C,\ si\ \beta \subseteq \alpha \Rightarrow \beta \in C$$

```
Variable V : finType.
Definition simplex := {set V}.
Definition simplicial_complex (c : {set simplex}) :=
  forall x, x \in c -> forall y : simplex, y \subset x -> y \in c.
```

Chain Complex

Definition

A chain complex C_* is a pair of sequences $C_* = (C_q, d_q)_{q \in \mathbb{Z}}$ where:

- For every $q \in \mathbb{Z}$, the component C_q is a R-module, the chain group in degree q
- For every $q \in \mathbb{Z}$, the component d_q is a morphism $d_q : C_q \to C_{q-1}$, the differential function
- For every $q \in \mathbb{Z}$, the composition $d_q d_{q+1}$ is null: $d_q d_{q+1} = 0$

Chain Complex

Definition

A chain complex C_* is a pair of sequences $C_* = (C_q, d_q)_{q \in \mathbb{Z}}$ where:

- For every $q \in \mathbb{Z}$, the component C_q is a R-module, the chain group in degree q
- For every $q \in \mathbb{Z}$, the component d_q is a morphism $d_q : C_q \to C_{q-1}$, the differential function
- For every $q \in \mathbb{Z}$, the composition $d_q d_{q+1}$ is null: $d_q d_{q+1} = 0$

Definition

Let $\mathcal{K}$ be a finite simplicial complex, $C_n(\mathcal{K})$ is a free module and the n-simplices of $\mathcal{K}$ form the standard basis of it. Then, given an order, for all n we can represent the differential map $d_n : C_n(\mathcal{K}) \to C_{n-1}(\mathcal{K})$ relative to the standard basis of the chain groups as a $\mathbb{Z}_2$ matrix. Such a matrix is called the *n-th incidence matrix* of a simplicial complex.

Chain Complex

Definition

A chain complex C_* is a pair of sequences $C_* = (C_q, d_q)_{q \in \mathbb{Z}}$ where:

- For every $q \in \mathbb{Z}$, the component C_q is a R-module, the chain group in degree q
- For every $q \in \mathbb{Z}$, the component d_q is a morphism $d_q : C_q \to C_{q-1}$, the differential function
- For every $q \in \mathbb{Z}$, the composition $d_q d_{q+1}$ is null: $d_q d_{q+1} = 0$

Definition

Let $\mathcal{K}$ be a finite simplicial complex, $C_n(\mathcal{K})$ is a free module and the n-simplices of $\mathcal{K}$ form the standard basis of it. Then, given an order, for all n we can represent the differential map $d_n : C_n(\mathcal{K}) \to C_{n-1}(\mathcal{K})$ relative to the standard basis of the chain groups as a $\mathbb{Z}_2$ matrix. Such a matrix is called the *n-th incidence matrix* of a simplicial complex.

```
Definition incidenceMatrix :=
  \matrix_(i < m, j < n)
  if (nth set0 Left i) \in (boundary (nth set0 Top j)) then 1 else 0:
       bool.
Definition incidence_mx_n :=
   incidenceMatrix (enum n_1_simplices)(enum n_simplices).
```

Chain Complex

Definition

A chain complex C_* is a pair of sequences $C_* = (C_q, d_q)_{q\in\mathbb{Z}}$ where:

- For every $q \in \mathbb{Z}$, the component C_q is a R-module, the chain group in degree q
- For every $q \in \mathbb{Z}$, the component d_q is a morphism $d_q : C_q \to C_{q-1}$, the differential function
- For every $q \in \mathbb{Z}$, the composition $d_q d_{q+1}$ is null: $d_q d_{q+1} = 0$

Definition

Let $\mathcal{K}$ be a finite simplicial complex, $C_n(\mathcal{K})$ is a free module and the n-simplices of $\mathcal{K}$ form the standard basis of it. Then, given an order, for all n we can represent the differential map $d_n : C_n(\mathcal{K}) \to C_{n-1}(\mathcal{K})$ relative to the standard basis of the chain groups as a $\mathbb{Z}_2$ matrix. Such a matrix is called the *n-th incidence matrix* of a simplicial complex.

```
Theorem incidence_matrices_sc_product:
  forall (V:finType) (n:nat) (sc: {set (simplex V)}),
  simplicial_complex sc ->
  (incidence_mx_n sc n) *m (incidence_mx_n sc (n.+1)) = 0.
```

Chain Complex

Definition

A chain complex C_* is a pair of sequences $C_* = (C_q, d_q)_{q \in \mathbb{Z}}$ where:

- For every $q \in \mathbb{Z}$, the component C_q is a R-module, the chain group in degree q
- For every $q \in \mathbb{Z}$, the component d_q is a morphism $d_q : C_q \to C_{q-1}$, the differential function
- For every $q \in \mathbb{Z}$, the composition $d_q d_{q+1}$ is null: $d_q d_{q+1} = 0$

Definition

Let $\mathcal{K}$ be a finite simplicial complex, $C_n(\mathcal{K})$ is a free module and the n-simplices of $\mathcal{K}$ form the standard basis of it. Then, given an order, for all n we can represent the differential map $d_n : C_n(\mathcal{K}) \to C_{n-1}(\mathcal{K})$ relative to the standard basis of the chain groups as a $\mathbb{Z}_2$ matrix. Such a matrix is called the *n-th incidence matrix* of a simplicial complex.

J. Heras, M. Poza, M. Dénès and L. Rideau. Incidence simplicial matrices formalized in COQ/SSREFLECT, Proceedings 18th Symposium on the Integration of Symbolic Computation and Mechanised Reasoning (Calculemus'11), Lecture Notes in Computer Science, vol. 6824, pages 30-44, 2011.

Homology

Definition

If $C_* = (C_q, d_q)_{q \in \mathbb{Z}}$ is a chain complex:

- *The image $B_q = im\ d_{q+1} \subseteq C_q$ is the (sub)module of q-boundaries*
- *The kernel $Z_q = ker\ d_q \subseteq C_q$ is the (sub)module of q-cycles*

Definition

Let $C_* = (C_q, d_q)_{q \in \mathbb{Z}}$ be a chain complex. For each degree $n \in \mathbb{Z}$, the n-homology module of C_* is defined as the quotient module

$$H_n(C_*) = \frac{Z_n}{B_n}$$

Homology

Definition

If $C_* = (C_q, d_q)_{q\in\mathbb{Z}}$ *is a chain complex:*

- *The image* $B_q = im\ d_{q+1} \subseteq C_q$ *is the (sub)module of q-boundaries*
- *The kernel* $Z_q = ker\ d_q \subseteq C_q$ *is the (sub)module of q-cycles*

Definition

Let $C_* = (C_q, d_q)_{q\in\mathbb{Z}}$ *be a chain complex. For each degree* $n \in \mathbb{Z}$*, the n-homology module of* C_* *is defined as the quotient module*

$$H_n(C_*) = \frac{Z_n}{B_n}$$

```
Variable (K : fieldType) (V1 V2 V3 : vectType K)
         (f : linearApp V1 V2) (g : linearApp V2 V3).
Definition Homology := ((lker g) :\: (limg f)).
```

Homology

Definition

If $C_* = (C_q, d_q)_{q\in\mathbb{Z}}$ is a chain complex:

- *The image $B_q = im\ d_{q+1} \subseteq C_q$ is the (sub)module of q-boundaries*
- *The kernel $Z_q = ker\ d_q \subseteq C_q$ is the (sub)module of q-cycles*

Definition

Let $C_* = (C_q, d_q)_{q\in\mathbb{Z}}$ be a chain complex. For each degree $n \in \mathbb{Z}$, the n-homology module of C_* is defined as the quotient module

$$H_n(C_*) = \frac{Z_n}{B_n}$$

```
Definition dim_homology (mxf:'M[K]_(v1,v2)) (mxg:'M[K]_(v2,v3)) :=
  v2 - \rank mxg - \rank mxf.

Lemma dimHomologyrankE: mxf *m mxg = 0 ->
  \dim Homology (LinearApp mxf)(LinearApp mxg) = dim_homology mxf mxg.
```

Homology

Definition

If $C_* = (C_q, d_q)_{q \in \mathbb{Z}}$ is a chain complex:

- *The image $B_q = im\ d_{q+1} \subseteq C_q$ is the (sub)module of q-boundaries*
- *The kernel $Z_q = ker\ d_q \subseteq C_q$ is the (sub)module of q-cycles*

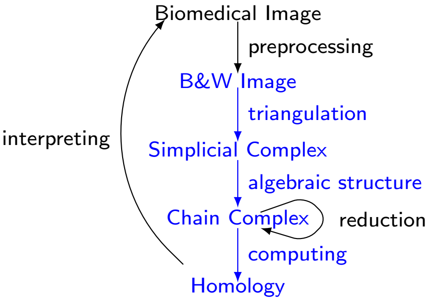
Definition

Let $C_* = (C_q, d_q)_{q \in \mathbb{Z}}$ be a chain complex. For each degree $n \in \mathbb{Z}$, the n-homology module of C_* is defined as the quotient module

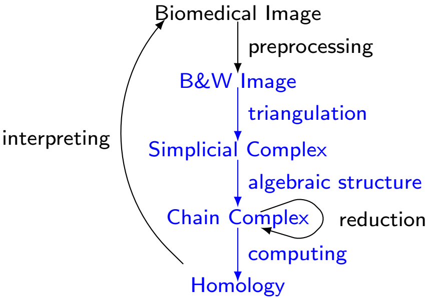
$$H_n(C_*) = \frac{Z_n}{B_n}$$

J. Heras, M. Dénès, G. Mata, A. Mörtberg, M. Poza and V. Siles. Towards a certified computation of homology groups for digital images, Proceedings 4th International Workshoph on Computational Topology in Image Context (CTIC'12), Lecture Notes in Computer Science, vol. 7309, pages 49-57, 2012.

Summary and problems

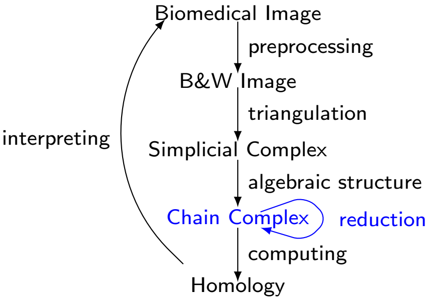


Summary and problems



- This process can be applied to any digital image
    - Reliability

Summary and problems



- This process can be applied to any digital image
  - Reliability
- Biomedical images:
  - Reliability
  - Efficiency: size of the images
- Solution to our approach to the tackle the efficiency problem

Table of Contents

Reduction

Definition

A reduction ρ between two chain complexes C_* y D_* (denoted by $\rho : C_* \Rrightarrow D_*$) is a triple $\rho = (f, g, h)$

$$\begin{tikzcd} C_* \arrow[loop left, "h"] \arrow[rr, bend left, "f"] & & D_* \arrow[ll, bend left, "g"] \end{tikzcd}$$

satisfying the following relations:

1) $fg = id_{D_*}$;
2) $d_{C_*} h + h d_{C_*} = id_{C_*} - gf$;
3) $fh = 0; \quad hg = 0; \quad hh = 0.$

Theorem

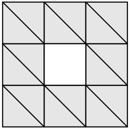
If $C_* \Rrightarrow D_*$, then $C_* \cong D_* \oplus A_*$, with A_* acyclic, this implies that $H_n(C_*) \cong H_n(D_*)$ for all n.

Discrete Vector Fields: intuitive idea

- Reduce the amount of information but keeping the homological properties
- Discrete Morse Theory
  - Vector fields are a tool to cancel “useless” information

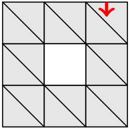
Discrete Vector Fields: intuitive idea

- Reduce the amount of information but keeping the homological properties
- Discrete Morse Theory
  - Vector fields are a tool to cancel “useless” information



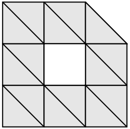
Discrete Vector Fields: intuitive idea

- Reduce the amount of information but keeping the homological properties
- Discrete Morse Theory
  - Vector fields are a tool to cancel "useless" information



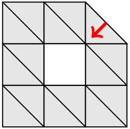
Discrete Vector Fields: intuitive idea

- Reduce the amount of information but keeping the homological properties
- Discrete Morse Theory
  - Vector fields are a tool to cancel “useless” information



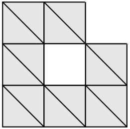
Discrete Vector Fields: intuitive idea

- Reduce the amount of information but keeping the homological properties
- Discrete Morse Theory
  - Vector fields are a tool to cancel "useless" information



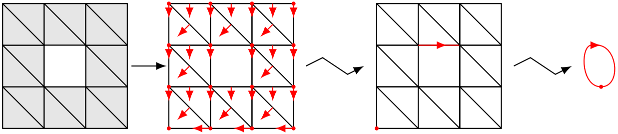
Discrete Vector Fields: intuitive idea

- Reduce the amount of information but keeping the homological properties
- Discrete Morse Theory
  - Vector fields are a tool to cancel "useless" information



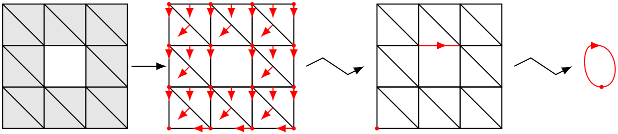
Discrete Vector Fields: intuitive idea

- Reduce the amount of information but keeping the homological properties
- Discrete Morse Theory
  - Vector fields are a tool to cancel "useless" information



Discrete Vector Fields: intuitive idea

- Reduce the amount of information but keeping the homological properties
- Discrete Morse Theory
  - Vector fields are a tool to cancel "useless" information



$$
\begin{array}{c}
0 \leftarrow \mathbb{Z}^{16} \xleftarrow{d_1} \mathbb{Z}^{32} \xleftarrow{d_2} \mathbb{Z}^{16} \leftarrow 0 \\
\downarrow \\
0 \leftarrow \mathbb{Z} \xleftarrow{\widehat{d}_1} \mathbb{Z} \xleftarrow{\widehat{d}_2} 0 \leftarrow 0
\end{array}
$$

Discrete Vector Fields

Definition

Let $C_* = (C_p, d_p)_{p\in\mathbb{Z}}$ be a free chain complex with distinguished $\mathbb{Z}-$basis $\beta_p \subset C_p$. A $(p-1)$-cell σ is a *face* of a p-cell τ if the coefficient of σ in $d\tau$ is non-null. It is a *regular face* if this coefficient is $+1$ or -1.

Definition

A *discrete vector field* on C_* is a collection of pairs $V = \{(\sigma_i, \tau_i)\}_{i\in\beta}$ satisfying the conditions:

1. Every σ_i is some element of β_p, in which case the other corresponding component $\tau_i \in \beta_{p+1}$. The degree p depends on i and in general is not constant
2. Every component σ_i is a *regular face* of the corresponding component τ_i
3. A generator of C_* appears at most one time in V

Discrete Vector Fields

Definition

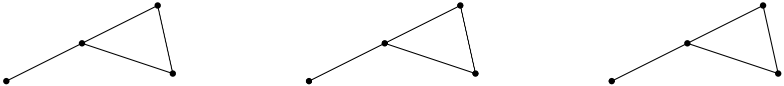
A *V-path of degree p* is a sequence $\pi = ((\sigma_{i_k}, \tau_{i_k}))_{0 \leq k < m}$ satisfying:

1. Every pair $((\sigma_{i_k}, \tau_{i_k}))$ is a component of V and the cell τ_{i_k} is a p-cell
2. For every $0 < k < m$, the component σ_{i_k} is a face of $\tau_{i_{k-1}}$, non necessarily regular, but different from $\sigma_{i_{k-1}}$

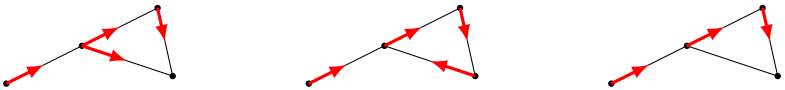
Definition

A discrete vector field V is *admissible* if for every $p \in \mathbb{Z}$, a function $\lambda_p : \beta_p \to \mathbb{Z}$ is provided satisfying the property: every V-path starting from $\sigma \in \beta_p$ has a length bounded by $\lambda_p(\sigma)$.

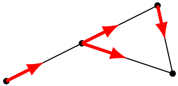
Example: an admissible discrete vector field



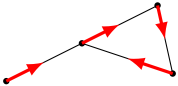
Example: an admissible discrete vector field

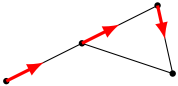


Example: an admissible discrete vector field

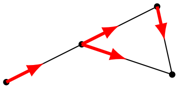


Dvf x

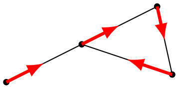




Example: an admissible discrete vector field

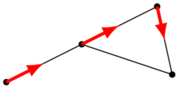


Dvf x

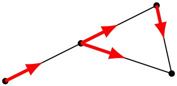


Dvf ✓

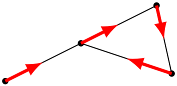
Admissible x



Example: an admissible discrete vector field

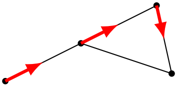


Dvf x



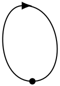
Dvf ✓

Admissible x



Dvf ✓

Admissible ✓



Discrete Vector Fields

Definition

A cell χ which does not appear in a discrete vector field $V = \{(\sigma_i, \tau_i)\}_{i \in \beta}$ is called a *critical cell*.

Vector-Field Reduction Theorem

Let $C_* = (C_p, d_p, \beta_p)_p$ be a free chain complex and $V = \{(\sigma_i, \beta_i)\}_{i \in \beta}$ be an admissible discrete vector field on C_*. Then the vector field V defines a canonical reduction $\rho = (f, g, h) : (C_p, d_p) \Longrightarrow (C_p^c, d_p')$ where $C_p^c = \mathbb{Z}\left[\beta_p^c\right]$ is the free $\mathbb{Z}-$module generated by the critical p-cells.

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

Vector fields and matrices

Differential maps of a chain complex of finite type can be represented as matrices

$$\ldots \leftarrow \mathbb{Z}_2^m \xleftarrow{M} \mathbb{Z}_2^n \leftarrow \ldots$$

Vector fields and matrices

Differential maps of a chain complex of finite type can be represented as matrices

$$\ldots \leftarrow \mathbb{Z}_2^m \xleftarrow{M} \mathbb{Z}_2^n \leftarrow \ldots$$

Definition

An admissible discrete vector field V for M is nothing but a set of integer pairs $\{(a_i, b_i)\}$ satisfying these conditions:

1. $1 \leq a_i \leq m$ *and* $1 \leq b_i \leq n$
2. *The entry* $M[a_i, b_i]$ *of the matrix is* 1
3. *The indices* a_i *(resp.* b_i*) are pairwise different*
4. *Non existence of loops*

Vector fields and matrices

Differential maps of a chain complex of finite type can be represented as matrices

$$\ldots \leftarrow \mathbb{Z}_2^m \xleftarrow{M} \mathbb{Z}_2^n \leftarrow \ldots$$

Definition

An admissible discrete vector field V for M is nothing but a set of integer pairs $\{(a_i, b_i)\}$ satisfying these conditions:

1. $1 \leq a_i \leq m$ *and* $1 \leq b_i \leq n$
2. *The entry $M[a_i, b_i]$ of the matrix is* 1
3. *The indices a_i (resp. b_i) are pairwise different*
4. *Non existence of loops*

Vector fields and matrices

Differential maps of a chain complex of finite type can be represented as matrices

$$\ldots \leftarrow \mathbb{Z}_2^m \xleftarrow{M} \mathbb{Z}_2^n \leftarrow \ldots$$

Definition

An admissible *discrete vector field V for M is nothing but a set of integer pairs $\{(a_i, b_i)\}$ satisfying these conditions:*

1. $1 \leq a_i \leq m$ *and* $1 \leq b_i \leq n$
2. *The entry $M[a_i, b_i]$ of the matrix is* 1
3. *The indices a_i (resp. b_i) are pairwise different*
4. *Non existence of loops*

Vector fields and matrices

Differential maps of a chain complex of finite type can be represented as matrices

$$\ldots \leftarrow \mathbb{Z}_2^m \xleftarrow{M} \mathbb{Z}_2^n \leftarrow \ldots$$

Definition

An admissible discrete vector field V for M is nothing but a set of integer pairs $\{(a_i, b_i)\}$ satisfying these conditions:

1. $1 \leq a_i \leq m$ *and* $1 \leq b_i \leq n$
2. *The entry* $M[a_i, b_i]$ *of the matrix is* 1
3. *The indices* a_i *(resp.* b_i*) are pairwise different*
4. *Non existence of loops*

```
Definition admissible_dvf (M: 'M[Z2]_(m,n))
                   (V: seq ('I_m * 'I_n)) (ords : simpl_rel 'I_m) :=
  all [pred p | M p.1 p.2 == 1] V &&
  uniq (map (@fst _ _) V) && uniq (map (@snd _ _) V) &&
  all [pred i | ~~ (connect ords i i)] (map (@fst _ _) V).
```

Main algorithms

Algorithm

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

Algorithm

Input: A chain complex C_* and an admissible discrete vector field of C_*
Output: A reduction from C_* to a reduced chain complex $\hat{C}_*$

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

Romero-Sergeraert's Algorithm

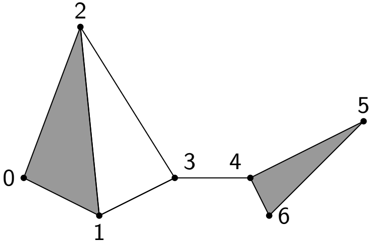
Algorithm (The RS Algorithm)

Input: a matrix M with coefficients in $\mathbb{Z}$.
Output: an admissible discrete vector field V for M and a list of relations r.

1. *Initialize the vector field V to the void vector field and the relations r to empty.*
2. *For every row i of M:*
   1. *For every column j, which is different from the second components of V, such that $M[i,j] = 1$ or $M[i,j] = -1$:*
      - *Look for the rows $k \neq i$ such as $M[k,j] \neq 0$ and obtain the relations $i > k$. Then, build the transitive closure of r and these relations.*

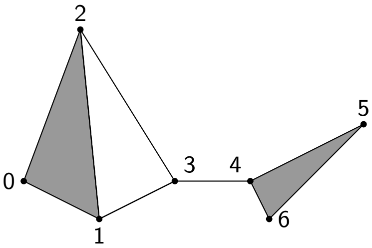
      `If` *there is no loop in that transitive closure:*
      `then:` *Add (i,j) to V, let r be that transitive closure, and repeat from Step 2 with the next row.*
      `else:` *Repeat from Step 2.1. with the next column.*

Example



$$
\begin{array}{cc}
 & \begin{array}{ccccccccc} \{0,1\} & \{0,2\} & \{1,2\} & \{1,3\} & \{2,3\} & \{3,4\} & \{4,5\} & \{4,6\} & \{5,6\} \end{array} \\
\begin{array}{c} \{0\} \\ \{1\} \\ \{2\} \\ \{3\} \\ \{4\} \\ \{5\} \\ \{6\} \end{array} &
\begin{pmatrix}
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1
\end{pmatrix}
\end{array}
$$

Example



$$\begin{array}{cc} & \begin{array}{cc} \{0,1,2\} & \{4,5,6\} \end{array} \\ \begin{array}{c} \{0,1\} \\ \{0,2\} \\ \{1,2\} \\ \{1,3\} \\ \{2,3\} \\ \{3,4\} \\ \{4,5\} \\ \{4,6\} \\ \{5,6\} \end{array} & \left(\begin{array}{cc} 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \end{array}\right) \end{array}$$

Example

$$\begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

1. dvf $= \{\}$, orders $= \{\}$

Example

$$\begin{pmatrix}
{\color{blue}1} & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
{\color{green}1} & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1
\end{pmatrix}$$

1. dvf $= \{\}$, orders $= \{\}$
2. dvf $= \{(1,1)\}$, orders $= \{\{1 > 2\}\}$

Example

$$\begin{pmatrix} \color{blue}{1} & \color{red}{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \color{red}{1} & 0 & \color{blue}{1} & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & \color{green}{1} & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

1. dvf $= \{\}$, orders $= \{\}$
2. dvf $= \{(1,1)\}$, orders $= \{\{1 > 2\}\}$
3. dvf $= \{(1,1),(2,3)\}$, orders $= \{\{1 > 2\},\{2 > 3\}\}$

Example

$$\begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

1. dvf $= \{\}$, orders $= \{\}$
2. dvf $= \{(1,1)\}$, orders $= \{\{1 > 2\}\}$
3. dvf $= \{(1,1),(2,3)\}$, orders $= \{\{1 > 2\},\{2 > 3\}\}$
4. dvf $= \{(1,1),(2,3),(3,5)\}$,
   orders $= \{\{1 > 2\},\{2 > 3\},\{3 > 4\}\}$
5. ...
6. dvf $= \{(1,1),(2,3),(3,5),(4,6),(5,7),(6,9)\}$
   orders $= \{\{1 > 2\},\{2 > 3\},\{3 > 4\},\{4 > 5\},\{5 > 6\},\{6 > 7\}\}$

The abstract specification

```
Fixpoint genDvfOrders M V (ords : simpl_rel _) k :=
  if k is l.+1 then
    let P := [pred ij | admissible (ij::V) M
                        (relU ords (gen_orders M ij.1 ij.2))] in
    if pick P is Some (i,j)
       then genDvfOrders M ((i,j)::V)
                         (relU ords (gen_orders M i j)) l
    else (V, ords)
  else (V, ords).

Definition gen_adm_dvf M :=
  genDvfOrders M [::] [rel x y | false] (minn m n).
```

The abstract specification

```
Fixpoint genDvfOrders M V (ords : simpl_rel _) k :=
  if k is l.+1 then
    let P := [pred ij | admissible (ij::V) M
                        (relU ords (gen_orders M ij.1 ij.2))] in
    if pick P is Some (i,j)
       then genDvfOrders M ((i,j)::V)
                         (relU ords (gen_orders M i j)) l
    else (V, ords)
  else (V, ords).

Definition gen_adm_dvf M :=
  genDvfOrders M [::] [rel x y | false] (minn m n).

Lemma admissible_gen_adm_dvf m n (M : 'M[Z2]_(m,n)) :
 let (vf,ords) := gen_adm_dvf M in admissible vf M ords.
```

The abstract specification

```
Fixpoint genDvfOrders M V (ords : simpl_rel _) k :=
  if k is l.+1 then
    let P := [pred ij | admissible (ij::V) M
                        (relU ords (gen_orders M ij.1 ij.2))] in
    if pick P is Some (i,j)
       then genDvfOrders M ((i,j)::V)
                         (relU ords (gen_orders M i j)) l
    else (V, ords)
  else (V, ords).

Definition gen_adm_dvf M :=
  genDvfOrders M [::] [rel x y | false] (minn m n).

Lemma admissible_gen_adm_dvf m n (M : 'M[Z2]_(m,n)) :
 let (vf,ords) := gen_adm_dvf M in admissible vf M ords.
```

Problem

It is not an executable algorithm

Verifying the RS-algorithm

```
Definition Z2 := Fp_fieldType 2.
Record matZ2:=
  {M:> seq (seq Z2);
   m:nat;
   n:nat;
   is_matrix: M = [::] \/
     [/\ m = size M & forall i, i < m -> size (rowseqmx M i) = n]
  }.
Definition vectorfield:= seq (prod nat nat).
Definition rels:= seq (seq nat).
```

Verifying the RS-algorithm

```
Definition Z2 := Fp_fieldType 2.
Record matZ2:=
  {M:> seq (seq Z2);
   m:nat;
   n:nat;
   is_matrix: M = [::] \/
     [/\ m = size M & forall i, i < m -> size (rowseqmx M i) = n]
  }.
Definition vectorfield:= seq (prod nat nat).
Definition rels:= seq (seq nat).

Definition Vecfieldadm (M: matZ2)(vf: vectorfield)(r:rels) :=
  (all [pred i | 0<= i < (M m)](getfirstseq vf)) /\
  (all [pred i | 0<= i < (M n)](getsndseq vf)) /\
  (forall i j:nat, (i,j) \in vf -> (nth 0 (nth nil M i) j) = 1%R) /\
  (uniq (getfirstseq vf)) /\
  (uniq (getsndseq vf)) /\
  (forall i j l:nat, (i,j) \in vf -> i!=l
     -> (nth 0 (nth nil M l) j)!= 0%R -> (i::l::nil) \in r) /\
  prop_cat2 r /\
  (all uniq r) /\
  (ordered glMax vf r).
```

Verifying the RS-algorithm

```
Theorem dvfordisVecfieldadm (M:matZ2):
  Vecfieldadm M (dvford M)(genOrders M).
```

Verifying the RS-algorithm

```
Theorem dvfordisVecfieldadm (M:matZ2):
  Vecfieldadm M (dvford M)(genOrders M).

Lemma v_in_genDvf_Mv1 (M: matZ2): (forall a b:nat,
  ((a,b) \in (dvford M)) -> nth 0 (nth nil M a) b = 1%R).
```

Verifying the RS-algorithm

```
Theorem dvfordisVecfieldadm (M:matZ2):
  Vecfieldadm M (dvford M)(genOrders M).

Lemma inDvf_compij1 (p a b:nat) (M : matZ2):
  (a,b) \in (fst (genDvfOrders p 0 0 M M [::] [::]))
  -> nth 0 (nth nil M a) b = 1%R.
```

Verifying the RS-algorithm

```
Theorem dvfordisVecfieldadm (M:matZ2):
  Vecfieldadm M (dvford M)(genOrders M).

Lemma inDvf_compij1 (p a b:nat) (M : matZ2):
  (a,b) \in (fst (genDvfOrders p 0 0 M M [::] [::]))
  -> nth 0 (nth nil M a) b = 1%R.

Lemma inDvf_compij1_general (p i j a b :nat) (M M2: matZ2)
   (vf: vectorfield)(r: rels):
 (forall k2, nth 0 (nth nil M (i + a)) k2 = nth 0 (nth nil M2 a) k2)
 -> (i + a,j + b) \in fst (genDvfOrders p i j M M2 vf r)
 -> nth 0 (nth nil M (i + a)) (j + b) = 1%R.
```

Verifying the RS-algorithm

```
Theorem dvfordisVecfieldadm (M:matZ2):
  Vecfieldadm M (dvford M)(genOrders M).
```

The proofs detailed in this section involve 49 definitions and 109 lemmas. In general, the development takes up 3772 code lines.

Verifying the RS-algorithm

```
Theorem dvfordisVecfieldadm (M:matZ2):
  Vecfieldadm M (dvford M)(genOrders M).
```

The proofs detailed in this section involve 49 definitions and 109 lemmas. In general, the development takes up 3772 code lines.

J. Heras, M. Poza and J. Rubio. Verifying an algorithm computing Discrete Vector Fields for digital imaging. Proceedings Conferences on Intelligence Computer Mathematics (CICM'12), Lecture Notes in Computer Science, vol. 7362, pages 215-229,2012. http://arxiv.org/abs/1005.5685v1.

General idea of the Vector-Field Reduction Theorem

Vector-Field Reduction Theorem

Let $C_* = (C_p, d_p, \beta_p)_p$ be a free chain complex and $V = \{(\sigma_i, \beta_i)\}_{i \in \beta}$ be an admissible discrete vector field on C_*. Then the vector field V defines a canonical reduction $\rho = (f, g, h) : (C_p, d_p) \Longrightarrow (C_p^c, d_p')$ where $C_p^c = \mathbb{Z}\left[\beta_p^c\right]$ is the free $\mathbb{Z}-$module generated by the critical p-cells.

General idea of the Vector-Field Reduction Theorem

Vector-Field Reduction Theorem

Let $C_* = (C_p, d_p, \beta_p)_p$ be a free chain complex and $V = \{(\sigma_i, \beta_i)\}_{i \in \beta}$ be an admissible discrete vector field on C_*. Then the vector field V defines a canonical reduction $\rho = (f, g, h) : (C_p, d_p) \Longrightarrow (C_p^c, d_p')$ where $C_p^c = \mathbb{Z}\left[\beta_p^c\right]$ is the free $\mathbb{Z}-$module generated by the critical p-cells.

$$\begin{tikzcd}
\text{Chain complex} \arrow[d, "\text{Isomorphism}"] \\
\text{Ordered chain complex}
\end{tikzcd}$$

General idea of the Vector-Field Reduction Theorem

Vector-Field Reduction Theorem

Let $C_* = (C_p, d_p, \beta_p)_p$ be a free chain complex and $V = \{(\sigma_i, \beta_i)\}_{i \in \beta}$ be an admissible discrete vector field on C_*. Then the vector field V defines a canonical reduction $\rho = (f, g, h) : (C_p, d_p) \Longrightarrow (C_p^c, d_p')$ where $C_p^c = \mathbb{Z}\left[\beta_p^c\right]$ is the free $\mathbb{Z}-$module generated by the critical p-cells.

$$\begin{tikzcd}
\text{Chain complex} \arrow[d, "\text{Isomorphism}"] \\
\text{Ordered chain complex} \arrow[d, "\text{Reduction}"] \\
\text{Reduced chain complex}
\end{tikzcd}$$

General idea of the Vector-Field Reduction Theorem

Vector-Field Reduction Theorem

Let $C_* = (C_p, d_p, \beta_p)_p$ be a free chain complex and $V = \{(\sigma_i, \beta_i)\}_{i \in \beta}$ be an admissible discrete vector field on C_*. Then the vector field V defines a canonical reduction $\rho = (f, g, h) : (C_p, d_p) \Longrightarrow (C_p^c, d_p')$ where $C_p^c = \mathbb{Z}\left[\beta_p^c\right]$ is the free $\mathbb{Z}-$module generated by the critical p-cells.

$$
\begin{tikzcd}
\text{Chain complex} \arrow[d, "\text{Isomorphism}"] \arrow[dd, bend right=60, "\text{Composition}"'] \\
\text{Ordered chain complex} \arrow[d, "\text{Reduction}"] \\
\text{Reduced chain complex}
\end{tikzcd}
$$

General idea of the Vector-Field Reduction Theorem

Vector-Field Reduction Theorem

Let $C_* = (C_p, d_p, \beta_p)_p$ be a free chain complex and $V = \{(\sigma_i, \beta_i)\}_{i \in \beta}$ be an admissible discrete vector field on C_*. Then the vector field V defines a canonical reduction $\rho = (f, g, h) : (C_p, d_p) \Longrightarrow (C_p^c, d'_p)$ where $C_p^c = \mathbb{Z}\left[\beta_p^c\right]$ is the free $\mathbb{Z}-$module generated by the critical p-cells.

$$\begin{tikzcd}
\text{Chain complex} \arrow[d, "\text{Isomorphism}"] \arrow[dd, bend right, "\text{Composition}"'] \\
\text{Ordered chain complex} \arrow[d, "\text{Reduction}"] \\
\text{Reduced chain complex}
\end{tikzcd}$$

- Hexagonal Lemma
- Basic Perturbation Lemma

General idea of the Vector-Field Reduction Theorem

Vector-Field Reduction Theorem

Let $C_* = (C_p, d_p, \beta_p)_p$ be a free chain complex and $V = \{(\sigma_i, \beta_i)\}_{i \in \beta}$ be an admissible discrete vector field on C_*. Then the vector field V defines a canonical reduction $\rho = (f, g, h) : (C_p, d_p) \Longrightarrow (C_p^c, d_p')$ where $C_p^c = \mathbb{Z}\left[\beta_p^c\right]$ is the free $\mathbb{Z}-$module generated by the critical p-cells.

$$\begin{tikzcd}
\text{Chain complex} \arrow[d, "\text{Isomorphism}"] \arrow[dd, bend right, "\text{Composition}"'] \\
\text{Ordered chain complex} \arrow[d, "\text{Reduction}"] \\
\text{Reduced chain complex}
\end{tikzcd}$$

- Hexagonal Lemma
- Basic Perturbation Lemma
  - Decomposition Theorem
  - Generalization of the Hexagonal Lemma

Vector-Field Reduction Theorem using HL

Hexagonal Lemma

Let $C = (C_p, d_p)_p$ be a chain complex. For some $k \in \mathbb{Z}$, the chain groups C_k and C_{k+1} are given with decompositions $C_k = C'_k \oplus C''_k$ and $C_{k+1} = C'_{k+1} \oplus C''_{k+1}$, so that between the degrees $k-1$ and $k+2$ this chain complex is described by the diagram:

$$\begin{tikzcd}
& C''_k \arrow[dl, "\delta"'] \arrow[rr, shift left, "\varepsilon^{-1}"] \arrow[d, dash, dotted] & & C''_{k+1} \arrow[ll, shift left, "\varepsilon"] \arrow[d, dash, dotted] \arrow[ddll, "\psi"] & \\
C_{k-1} & \oplus \arrow[l, "d"'] & & \oplus \arrow[ll, "d"'] & C_{k+2} \arrow[l, "d"'] \arrow[ul, "\eta"'] \arrow[dl, "\gamma"] \\
& C'_k \arrow[ul, "\alpha"] \arrow[u, dash, dotted] & & C'_{k+1} \arrow[ll, "\beta"] \arrow[uull, "\varphi"] \arrow[u, dash, dotted] &
\end{tikzcd} \qquad (1)$$

The partial differential $\varepsilon : C''_{k+1} \to C''_k$ is assumed to be an isomorphism. Then a canonical reduction can be defined $\rho : C \Rrightarrow C'$ where C' is the same chain complex as C except between the degrees $k-1$ and $k+2$:

$$\ldots \longleftarrow C_{k-2} \longleftarrow C_{k-1} \xleftarrow{\alpha} C'_k \xleftarrow{\beta-\psi\varepsilon^{-1}\varphi} C'_{k+1} \xleftarrow{\gamma} C_{k+2} \longleftarrow C_{k+3} \longleftarrow \ldots$$

Vector-Field Reduction Theorem using HL

Hexagonal Lemma

Let $C = (C_p, d_p)_p$ be a chain complex. For some $k \in \mathbb{Z}$, the chain groups C_k and C_{k+1} are given with decompositions $C_k = C'_k \oplus C''_k$ and $C_{k+1} = C'_{k+1} \oplus C''_{k+1}$, so that between the degrees $k-1$ and $k+2$ this chain complex is described by the diagram:

$$
\begin{tikzcd}
 & C''_k \arrow[dl, "\delta"'] \arrow[r, shift left, "\varepsilon^{-1}"] \arrow[d, dotted, no head] & C''_{k+1} \arrow[l, shift left, "\varepsilon"] \arrow[dl, "\psi"] \arrow[d, dotted, no head] & \\
C_{k-1} & \oplus \arrow[l, "d"'] & \oplus \arrow[l, "d"'] & C_{k+2} \arrow[l, "d"'] \arrow[ul, "\eta"'] \arrow[dl, "\gamma"] \\
 & C'_k \arrow[ul, "\alpha"] \arrow[u, dotted, no head] & C'_{k+1} \arrow[l, "\beta"] \arrow[ul, "\varphi"] \arrow[u, dotted, no head] &
\end{tikzcd}
\tag{1}
$$

The partial differential $\varepsilon : C''_{k+1} \to C''_k$ is assumed to be an isomorphism. Then a canonical reduction can be defined $\rho : C \Rrightarrow C'$ where C' is the same chain complex as C except between the degrees $k-1$ and $k+2$:

$$\ldots \longleftarrow C_{k-2} \longleftarrow C_{k-1} \xleftarrow{\alpha} C'_k \xleftarrow{\beta - \psi\varepsilon^{-1}\varphi} C'_{k+1} \xleftarrow{\gamma} C_{k+2} \longleftarrow C_{k+3} \longleftarrow \ldots$$

Development: 303 definitions, 361 lemmas and 7511 lines

Basic Perturbation Lemma

Basic Perturbation Lemma (BPL)

Let us consider a reduction $\rho = (f, g, h) : C_* \Rrightarrow \widehat{C}_*$ between two chain complexes (C_*, d) and $(\widehat{C}_*, \widehat{d})$, and δ a perturbation of d. Furthermore, the composite function δh is assumed *locally nilpotent*. Then, a perturbation $\widehat{\delta}$ can be defined for the differential map $\widehat{d}$ and a new reduction $\rho' = (f', g', h') : (C_*, d + \delta) \Rrightarrow (\widehat{C}_*, \widehat{d} + \widehat{\delta})$ can be constructed.

Basic Perturbation Lemma

Basic Perturbation Lemma (BPL)

Let us consider a reduction $\rho = (f, g, h) : C_* \Rrightarrow \widehat{C}_*$ between two chain complexes (C_*, d) and $(\widehat{C}_*, \widehat{d})$, and δ a perturbation of d. Furthermore, the composite function δh is assumed *locally nilpotent*. Then, a perturbation $\widehat{\delta}$ can be defined for the differential map $\widehat{d}$ and a new reduction $\rho' = (f', g', h') : (C_*, d + \delta) \Rrightarrow (\widehat{C}_*, \widehat{d} + \widehat{\delta})$ can be constructed.

- The non-graded case of this lemma was proved in *Isabelle/HOL*.
  - J. Aransay, C. Ballarin and J. Rubio. A mechanized proof of the Basic Perturbation Lemma, Journal of Automated Reasoning, volume 40-4, pages 271-292, 2008.
- A particular case of the BPL was also proved in Coq using bicomplexes.
  - C. Domínguez and J. Rubio. Effective Homology of Bicomplexes, formalized in Coq, Theoretical Computer Science, volume 412, pages 962-970, 2011.

Basic Perturbation Lemma

Basic Perturbation Lemma (BPL)

Let us consider a reduction $\rho = (f, g, h) : C_* \Rrightarrow \widehat{C}_*$ between two chain complexes (C_*, d) and $(\widehat{C}_*, \widehat{d})$, and δ a perturbation of d. Furthermore, the composite function δh is assumed *locally nilpotent*. Then, a perturbation $\widehat{\delta}$ can be defined for the differential map $\widehat{d}$ and a new reduction $\rho' = (f', g', h') : (C_*, d + \delta) \Rrightarrow (\widehat{C}_*, \widehat{d} + \widehat{\delta})$ can be constructed.

- The non-graded case of this lemma was proved in *Isabelle/HOL*.

J. Aransay, C. Ballarin and J. Rubio. A mechanized proof of the Basic Perturbation Lemma, Journal of Automated Reasoning, volume 40-4, pages 271-292, 2008.

- A particular case of the BPL was also proved in Coq using bicomplexes.

C. Domínguez and J. Rubio. Effective Homology of Bicomplexes, formalized in Coq, Theoretical Computer Science, volume 412, pages 962-970, 2011.

Goal

A formalization of the general case in SSReflect with finitely generated structures.

Basic Perturbation Lemma

Basic Perturbation Lemma (BPL)

Let us consider a reduction $\rho = (f, g, h) : C_* \Rrightarrow \widehat{C}_*$ between two chain complexes (C_*, d) and $(\widehat{C}_*, \widehat{d})$, and δ a perturbation of d. Furthermore, the composite function δh is assumed *locally nilpotent*. Then, a perturbation $\widehat{\delta}$ can be defined for the differential map $\widehat{d}$ and a new reduction $\rho' = (f', g', h') : (C_*, d + \delta) \Rrightarrow (\widehat{C}_*, \widehat{d} + \widehat{\delta})$ can be constructed.

```
Variable K: fieldType.
Variable rho : FGReduction K.
Variable delta : forall i:Z, 'M[K]_(m (C rho)(i+1), m (C rho) i).
Hypothesis boundary_dp : forall i:Z,
 ((diff (C rho)(i+1) + delta (i+1)) *m ((diff (C rho)i + delta i) = 0.

Variable (n : nat).
Hypothesis nilpotency_hp : forall i:Z,
  (pot_matrix (delta i *m (Ho (H rho) i)) n = 0).
```

Basic Perturbation Lemma

Basic Perturbation Lemma (BPL)

Let us consider a reduction $\rho = (f, g, h) : C_* \Rrightarrow \widehat{C}_*$ between two chain complexes (C_*, d) and $(\widehat{C}_*, \widehat{d})$, and δ a perturbation of d. Furthermore, the composite function δh is assumed *locally nilpotent*. Then, a perturbation $\widehat{\delta}$ can be defined for the differential map $\widehat{d}$ and a new reduction $\rho' = (f', g', h') : (C_*, d + \delta) \Rrightarrow (\widehat{C}_*, \widehat{d} + \widehat{\delta})$ can be constructed.

```
Variable K: fieldType.
Variable rho : FGReduction K.
Variable delta : forall i:Z, 'M[K]_(m (C rho)(i+1), m (C rho) i).
Hypothesis boundary_dp : forall i:Z,
 ((diff (C rho)(i+1) + delta (i+1)) *m ((diff (C rho)i + delta i) = 0.

Variable (n : nat).
Hypothesis nilpotency_hp : forall i:Z,
  (pot_matrix (delta i *m (Ho (H rho) i)) n = 0).

Definition quasi_bpl :=
  (rhoHL (Di:= Di_pert) (boundary_Di := boundary_dp_new)
       inverse_dp_12_inverse).
```

Basic Perturbation Lemma

Basic Perturbation Lemma (BPL)

Let us consider a reduction $\rho = (f, g, h) : C_* \Rrightarrow \widehat{C}_*$ between two chain complexes (C_*, d) and $(\widehat{C}_*, \widehat{d})$, and δ a perturbation of d. Furthermore, the composite function δh is assumed *locally nilpotent*. Then, a perturbation $\widehat{\delta}$ can be defined for the differential map $\widehat{d}$ and a new reduction $\rho' = (f', g', h') : (C_*, d + \delta) \Rrightarrow (\widehat{C}_*, \widehat{d} + \widehat{\delta})$ can be constructed.

- Development
  - 63 definitions
  - 117 lemmas
  - 2416 lines

M. Poza, C. Domínguez, J. Heras, and J. Rubio. A certified reduction strategy for homological image processing. Submitted, 2013, http://www.unirioja.es/cu/cedomin/crship/

Key aspects of the formalization

- The role of SSReflect
- Different representations
- Casts
- Dealing with kernels

The role of SSReflect

- Libraries:
  - matrix.v: theory matrix, determinant, matrix decomposition,...

    $d'_1 * d'_2 = 0 \rightarrow \left(\begin{array}{c|c} \varepsilon & \varphi \\ \hline \psi & \beta \end{array}\right) * \left(\begin{array}{c} \eta \\ \hline \gamma \end{array}\right) = 0$. Therefore,

    1. $\varepsilon * \eta + \varphi * \gamma = 0$ which implies that $\varphi * \gamma = -\varepsilon * \eta$
    2. $\psi * \eta + \beta * \gamma = 0$

```
Definition block_mx Aul Aur Adl Adr : 'M_(m1 + m2, n1 + n2) :=
  col_mx (row_mx Aul Aur) (row_mx Adl Adr).
```

The role of SSReflect

- Libraries:
  - `matrix.v`: theory matrix, determinant, matrix decomposition...
  - `finset.v` and `fintype.v`

```
Variable V : finType.
Definition simplex := {set V}.
Definition simplicial_complex (c : {set simplex}) :=
  forall x, x \in c -> forall y : simplex, y \subset x -> y \
       in c.
```

The role of SSReflect

- Libraries:
  - `matrix.v`: theory matrix, determinant, matrix decomposition...
  - `finset.v` and `fintype.v`
  - `bigop.v`

$$\sum_{i \in r | P_i} F_i = \sum_{i \in r | P_i \wedge a_i} F_i + \sum_{i \in r | P_i \wedge \sim a_i} F_i$$

  - ...

The role of SSReflect

- Libraries:
  - `matrix.v`: theory matrix, determinant, matrix decomposition...
  - `finset.v` and `fintype.v`
  - `bigop.v`

$$\sum_{i \in r | P_i} F_i = \sum_{i \in r | P_i \wedge a_i} F_i + \sum_{i \in r | P_i \wedge \sim a_i} F_i$$

  - ...
- Efficiency when writing of proofs

The role of SSReflect

- Libraries:
  - `matrix.v`: theory matrix, determinant, matrix decomposition...
  - `finset.v` and `fintype.v`
  - `bigop.v`

$$\sum_{i \in r \mid P_i} F_i = \sum_{i \in r \mid P_i \wedge a_i} F_i + \sum_{i \in r \mid P_i \wedge \sim a_i} F_i$$

  - ...
- Efficiency when writing of proofs
- Definitions are blocked not to be expanded during type checking
- Definitions lack direct effective computation

Two SSReflect matrix representations

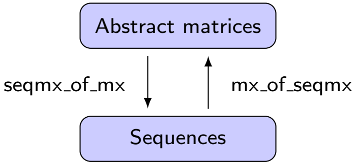
- As functions
  - Definition of different operations and proved properties about them
  - Not directly executable
- As sequences of sequences
  - Operations can be executed
  - Prove properties is much harder
  - There is not an extensive SSReflect development

Two SSReflect matrix representations

- As functions
  - Definition of different operations and proved properties about them
  - Not directly executable
- As sequences of sequences
  - Operations can be executed
  - Prove properties is much harder
  - There is not an extensive SSReflect development

Conclusions

To compute $\mapsto$ Sequences
To prove $\mapsto$ Abstract matrices



Chain complexes representations

The chain complex associated with a simplicial complex related to a 2D image

$$\ldots \leftarrow 0 \leftarrow 0 \leftarrow C_0 \xleftarrow{d_1} C_1 \xleftarrow{d_2} C_2 \leftarrow 0 \leftarrow 0 \leftarrow \ldots$$

A *truncated* chain complex is

$$C_0 \xleftarrow{d_1} C_1 \xleftarrow{d_2} C_2$$

Chain complexes representations

The chain complex associated with a simplicial complex related to a 2D image

$$\ldots \leftarrow 0 \leftarrow 0 \leftarrow C_0 \xleftarrow{d_1} C_1 \xleftarrow{d_2} C_2 \leftarrow 0 \leftarrow 0 \leftarrow \ldots$$

A *truncated* chain complex is

$$C_0 \xleftarrow{d_1} C_1 \xleftarrow{d_2} C_2$$

```
Definition is_chaincomplex (d1 d2: matZ2) (m n p: nat):=
  is_matrix m n d1 /\
  is_matrix n p d2 /\
  (mx_of_seqmx m n d1) *m (mx_of_seqmx n p d2) = 0.

Record chaincomplex:=
  {d1: matZ2;
   d2: matZ2;
   m: nat;
   n: nat;
   p: nat;
   chaincomplex_proof: is_chaincomplex d1 d2 m n p}.
```

Chain complexes representations

The chain complex associated with a simplicial complex related to a 2D image

$$\ldots \leftarrow 0 \leftarrow 0 \leftarrow C_0 \xleftarrow{d_1} C_1 \xleftarrow{d_2} C_2 \leftarrow 0 \leftarrow 0 \leftarrow \ldots$$

A *truncated* chain complex is

$$C_0 \xleftarrow{d_1} C_1 \xleftarrow{d_2} C_2$$

```
Variable K : fieldType.

Record FGChain_Complex :=
  { m : Z -> nat;
    diff : forall i:Z, 'M[K]_(m (i + 1), m i);
    boundary : forall i:Z, (diff (i + 1)) *m (diff i) = 0}.
```

Casts

There exists a rigid typing strategy in Coq

`M_i` $\not\Rightarrow$ `M_(i+1-1)`

Casts

There exists a rigid typing strategy in Coq

$$\texttt{M\_i} \not\Rightarrow \texttt{M\_(i+1-1)}$$

In a reduction, $g \circ f + d \circ h + h \circ d = \mathrm{id}$ where $d_i : C_i \to C_{i-1}$ and $h_i : C_i \to C_{i+1}$
Then, $d_{i+1}h_i : C_i \to C_{i+1-1}$ `'M_(m C i, m C (i+1-1))` $\not\Rightarrow$ `'M_(m C i)`

Casts

There exists a rigid typing strategy in Coq

$$\texttt{M\_i} \not\Rightarrow \texttt{M\_(i+1-1)}$$

In a reduction, $g \circ f + d \circ h + h \circ d = \text{id}$ where $d_i : C_i \to C_{i-1}$ and $h_i : C_i \to C_{i+1}$
Then, $d_{i+1}h_i : C_i \to C_{i+1-1}$ $\qquad$ `'M_(m C i, m C (i+1-1))` $\not\Rightarrow$ `'M_(m C i)`

```
Lemma cast1: m C i = m C i.
Lemma cast2: m C (i+1-1) = m C i.
castmx (cast1, cast2) ((Ho H i) *m (diff C (i+1)))
```

Casts

There exists a rigid typing strategy in Coq

$$\texttt{M\_i} \not\Rightarrow \texttt{M\_(i+1-1)}$$

In a reduction, $g \circ f + d \circ h + h \circ d = \text{id}$ where $d_i : C_i \to C_{i-1}$ and $h_i : C_i \to C_{i+1}$
Then, $d_{i+1}h_i : C_i \to C_{i+1-1}$ 'M_(m C i, m C (i+1-1)) $\not\Rightarrow$ 'M_(m C i)

```
Lemma cast1: m C i = m C i.
Lemma cast2: m C (i+1-1) = m C i.
castmx (cast1, cast2) ((Ho H i) *m (diff C (i+1)))
```

Re-indexing structures

If $d_i : C_{i+1} \to C_i$ then $d_{i+1}h_i : C_i \to C_i$ and the obtained matrix is 'M_(m C i).

Notations

An *n-suspended up* or *n-suspended down* chain complex for a chain complex is built moving up or down n degrees of a chain complex.

Dealing with kernels

Decomposition Theorem

Let $\rho = (f, g, h) : (C_*, d) \Rrightarrow (\widehat{C}_*, \widehat{d})$ be a reduction. This reduction is equivalent to a decomposition: $C_* = A_* \oplus B_* \oplus C'_*$ where $A_* = \ker f \cap \ker h$, $B_* = \ker f \cap \ker d$ and $C'_* = \operatorname{im} g$.

Dealing with kernels

Decomposition Theorem

Let $\rho = (f, g, h) : (C_*, d) \Rrightarrow (\widehat{C}_*, \widehat{d})$ be a reduction. This reduction is equivalent to a decomposition: $C_* = A_* \oplus B_* \oplus C'_*$ where $A_* = \ker f \cap \ker h$, $B_* = \ker f \cap \ker d$ and $C'_* = \operatorname{im} g$.

The kernel of a finite map is defined by the kernel of the matrix which represents this map.

Dealing with kernels

Decomposition Theorem

Let $\rho = (f, g, h) : (C_*, d) \Rightarrow (\widehat{C}_*, \widehat{d})$ be a reduction. This reduction is equivalent to a decomposition: $C_* = A_* \oplus B_* \oplus C'_*$ where $A_* = \ker f \cap \ker h$, $B_* = \ker f \cap \ker d$ and $C'_* = \operatorname{im} g$.

The kernel of a finite map is defined by the kernel of the matrix which represents this map.

```
Definition kermx m n (A: 'M_(m,n)): 'M_m :=
  copid_mx (\rank A) *m invmx (col_ebase A).
Lemma mulmx_ker m n (A : 'M_(m, n)) : kermx A *m A = 0.
```

Dealing with kernels

Decomposition Theorem

Let $\rho = (f, g, h) : (C_*, d) \Rightarrow\!\!\!\Rightarrow (\widehat{C}_*, \widehat{d})$ be a reduction. This reduction is equivalent to a decomposition: $C_* = A_* \oplus B_* \oplus C'_*$ where $A_* = \ker f \cap \ker h$, $B_* = \ker f \cap \ker d$ and $C'_* = \operatorname{im} g$.

The kernel of a finite map is defined by the kernel of the matrix which represents this map.

```
Definition kermx m n (A: 'M_(m,n)): 'M_m :=
  copid_mx (\rank A) *m invmx (col_ebase A).
Lemma mulmx_ker m n (A : 'M_(m, n)) : kermx A *m A = 0.
```

- Drawbacks
  - The kernel consists of the elements that are made null when they are applied to the left
  - It is necessary to work with transposed matrices
  - The product is reversed
  - Only *partial* identities are obtained
- Advantages
  - Some useful lemmas about `kermx` are already proven in the SSReflect library

Dealing with kernels

Decomposition Theorem

Let $\rho = (f, g, h) : (C_*, d) \Rightarrow\!\!\!\Rightarrow (\widehat{C}_*, \widehat{d})$ be a reduction. This reduction is equivalent to a decomposition: $C_* = A_* \oplus B_* \oplus C'_*$ where $A_* = \ker f \cap \ker h$, $B_* = \ker f \cap \ker d$ and $C'_* = \mathrm{im}\, g$.

The kernel of a finite map is defined by the kernel of the matrix which represents this map.

```
Definition kermx m n (A: 'M_(m,n)): 'M_m :=
  copid_mx (\rank A) *m invmx (col_ebase A).
Lemma mulmx_ker m n (A : 'M_(m, n)) : kermx A *m A = 0.
```

```
Definition ker_min (m n : nat) (M : 'M_(m,n)) :=
  (castmx ((Logic.eq_refl (m-\rank M)),(\rank M) + (m-(\rank M))) = m)
  (row_mx (@const_mx _ (m-\rank M) (\rank M) 0) 1%:M)) *m (kermx M).
```

Dealing with kernels

Decomposition Theorem

Let $\rho = (f, g, h) : (C_*, d) \Rightarrow\!\!\!\!\!\Rightarrow (\widehat{C}_*, \widehat{d})$ be a reduction. This reduction is equivalent to a decomposition: $C_* = A_* \oplus B_* \oplus C'_*$ where $A_* = \ker f \cap \ker h$, $B_* = \ker f \cap \ker d$ and $C'_* = \operatorname{im} g$.

The kernel of a finite map is defined by the kernel of the matrix which represents this map.

```
Definition kermx m n (A: 'M_(m,n)): 'M_m :=
  copid_mx (\rank A) *m invmx (col_ebase A).
Lemma mulmx_ker m n (A : 'M_(m, n)) : kermx A *m A = 0.


Definition ker_min (m n : nat) (M : 'M_(m,n)) :=
  (castmx ((Logic.eq_refl (m-\rank M)),(\rank M) + (m-(\rank M))) = m)
  (row_mx (@const_mx _ (m-\rank M) (\rank M) 0) 1%:M)) *m (kermx M).

Lemma ker_min_kermx (m n : nat) (M : 'M_(m,n)) :
  (kermx M :=: (ker_min M))%MS.
```

Table of Contents

Methodology

Goal

A methodology to verify a software program which smooths the "steep learning curve"

1. Implement a version of our algorithms in Haskell
2. Test properties about the Haskell programs
3. Verify the programs using Coq and its SSReflect library

Methodology

Goal

A methodology to verify a software program which smooths the "steep learning curve"

1. Implement a version of our algorithms in Haskell
2. Test properties about the Haskell programs
3. Verify the programs using Coq and its SSReflect library

Advantages of Haskell

- Both the code and the way of working are similar to the ones in Coq
- The clean semantics of purely functional languages
- Haskell functions often satisfy simple algebraic properties
- Provides a profiling system
  - Where the system wastes time
  - Which parts of the proof should be improved
  - Data structures are suitable

Methodology

Goal

A methodology to verify a software program which smooths the "steep learning curve"

1. Implement a version of our algorithms in Haskell.
2. Test properties about the Haskell programs
3. Verify the programs using COQ and its SSREFLECT library

Testing

Goal: The testing process can be useful in order to detect bugs

- Manual testing
  - Small images
  - Very tedious and requires a lot of time

Methodology

Goal

A methodology to verify a software program which smooths the "steep learning curve"

1. Implement a version of our algorithms in Haskell
2. Test properties about the Haskell programs
3. Verify the programs using Coq and its SSReflect library

Testing

Goal: The testing process can be useful in order to detect bugs

- Manual testing
- Automated testing
  - Use fKenzo
    - File with a battery of pairs of matrices (2D images randomly generated)
    - Compute homology
  - Use Haskell
    - Compute homology in Haskell with or without applying a reduction process

Methodology

Goal

A methodology to verify a software program which smooths the "steep learning curve"

1. Implement a version of our algorithms in Haskell
2. Test properties about the Haskell programs
3. Verify the programs using Coq and its SSReflect library

Testing

Goal: The testing process can be useful in order to detect bugs

- Manual testing
- Automated testing

Study over 250 images	d1	d2
% of reduction	98	49

Methodology

Goal

A methodology to verify a software program which smooths the "steep learning curve"

1. Implement a version of our algorithms in Haskell
2. Test properties about the Haskell programs
3. Verify the programs using Coq and its SSReflect library

Testing

Goal: The testing process can be useful in order to detect bugs

- Manual testing
- Automated testing
- QuickCheck
    - A specification of the properties which must be satisfied by our programs is given
    - Testing the properties included in the specification

Methodology

Goal

A methodology to verify a software program which smooths the "steep learning curve"

1. Implement a version of our algorithms in Haskell
2. Test properties about the Haskell programs
3. Verify the programs using Coq and its SSReflect library

Testing

Goal: The testing process can be useful in order to detect bugs

- Manual testing
- Automated testing
- QuickCheck

This step is **not enough** to ensure that a program is correct

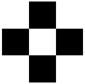
Alternatives in the methodology

- Efficiency: Haskell is a programming language
- Reliability: we are not sure of the correctness of our programs

Haskell as an oracle

The heavy computations are done in Haskell and then the properties of the output of the computation are proved in SSREFLECT

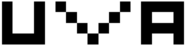
Computing over digital images



	Haskell	*SSReflect*
Without advf	**0.06**	**0.46**
With advf	**0.046875**	**1.016**

- We can compute in both systems

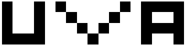
Computing over digital images



		Haskell	*SSReflect*
Without advf		**0.9**	**9**
With advf	Computation advf	2.484	101
	Ordered matrices	0	14
	Reduced matrix d_1	0	$\star$
	Reduced matrix d_2	0.0624	5
	H_0 and H_1	1.252	3
	Total	**3.9**	**139**

- SSReflect cannot compute the inverse of a matrix 64×64

Computing over digital images

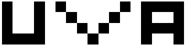


		Haskell	*SSReflect*
Without advf		**0.9**	**9**
With advf	Computation advf	2.484	101
	Ordered matrices	0	14
	Reduced matrix d_1	0	$\star$
	Reduced matrix d_2	0.0624	5
	H_0 and H_1	1.252	3
	Total	**3.9**	**139**

- Haskell as an oracle:
  - Compute the admissible discrete vector field and reorder the matrix in Coq
  - The top-left block `ulM` is extracted to Haskell and the inverse matrix is returned `invmx_ulM`
  - Prove in Coq:
    - The inverse of a matrix is unique
    - $\forall M, MM^{-1} = \text{id} \Rightarrow M^{-1}M = \text{id}$

```
Lemma m_invmxm : (mulseqmx ulM invmx_ulM)
== (scalar_seqmx 64 (Ordinal (ltn_pmod 1 (ltn0Sn 1)))).
```

Computing over digital images



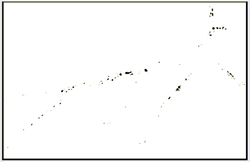
		Haskell	*SSReflect*
Without advf		**0.9**	**9**
With advf	Computation advf	2.484	101
	Ordered matrices	0	14
	Reduced matrix d_1	0	$\star$
	Reduced matrix d_2	0.0624	5
	H_0 and H_1	1.252	3
	Total	**3.9**	**139**

- The RS algorithm is not necessary
- We can compute the dimension of the homology groups with the direct method

Computing over biomedical images

Biomedical context

Counting the number of synapses in a neuron



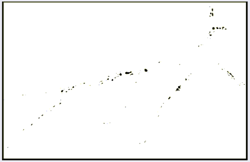
Problem

- SSReflect cannot compute the homology directly

Computing over biomedical images

Biomedical context

Counting the number of synapses in a neuron



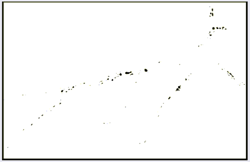
Goal

Obtain the H_0 by means of a certified process

Computing over biomedical images

Biomedical context

Counting the number of synapses in a neuron



Goal

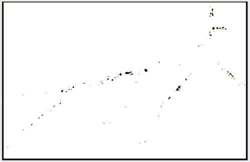
Obtain the H_0 by means of a certified process

1. Computing in Haskell the reduced matrix *Mred* and the components $f0$, $f1$, $g0$, $g1$ and $h0$ which define a 2-truncated reduction
2. Transform the matrices to Coq/SSReflect
3. Prove in Coq/SSReflect that these matrices establish a reduction
4. Compute H_0 from the reduced matrix *Mred* in Coq/SSReflect

Computing over biomedical images

Biomedical context

Counting the number of synapses in a neuron



		Haskell (min)	*SSReflect*
Without advf		**0.68**	**Not available**
With advf	Computation advf	108	Proofs 12h 4min 55sec
	Reduced matrices	26	
	H_0	0.012	5sec
	Total	**130**	**12h 5min**

- We need to use the reduction method to obtain H_0 in a reliable way
- The matrix 743×1424 is reduced to 59×740

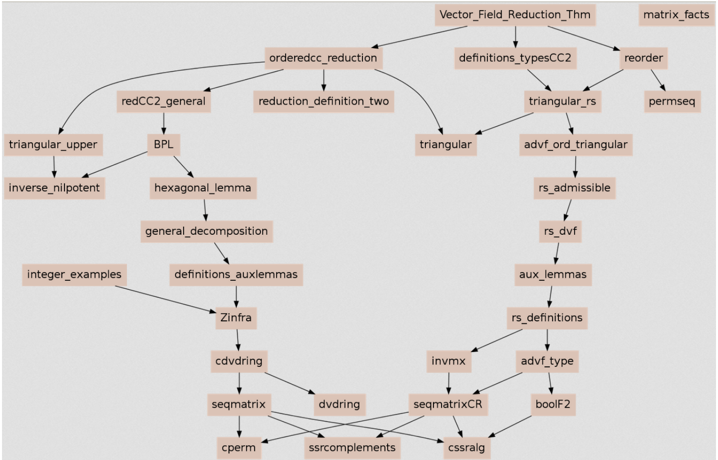
Sources of inefficiency

- Data types in languages of functional programming
  - Matrices
- Use of simplicial complexes instead of cubical complexes
- Execution inside the proof assistant
- Coq is a Proof Assistant and not a Computer Algebra system
- Concrete algorithms
  - Kenzo (ad-hoc algorithm to compute an admissible discrete vector field)
  - Heuristic techniques
- Proving needs more redundancy in algorithms

Table of Contents

Conclusions

- Development effort



Conclusions

- Development effort
- The implementation in COQ/SSREFLECT of the RS algorithm

Conclusions

- Development effort
- The implementation in Coq/SSReflect of the RS algorithm
- The complete formalization in Coq/SSReflect of the BPL

Conclusions

- Development effort
- The implementation in Coq/SSReflect of the RS algorithm
- The complete formalization in Coq/SSReflect of the BPL
- Two formalizations of the Vector-Field Reduction Theorem for matrices

Conclusions

- Development effort
- The implementation in Coq/SSReflect of the RS algorithm
- The complete formalization in Coq/SSReflect of the BPL
- Two formalizations of the Vector-Field Reduction Theorem for matrices
- A verified program to compute homology groups of a simplicial complex obtained from a digital image

Conclusions

- Development effort
- The implementation in Coq/SSReflect of the RS algorithm
- The complete formalization in Coq/SSReflect of the BPL
- Two formalizations of the Vector-Field Reduction Theorem for matrices
- A verified program to compute homology groups of a simplicial complex obtained from a digital image
- A discussion on different methods to overcome the efficiency problems appearing when executing programs inside proof assistants

Conclusions

- Development effort
- The implementation in Coq/SSReflect of the RS algorithm
- The complete formalization in Coq/SSReflect of the BPL
- Two formalizations of the Vector-Field Reduction Theorem for matrices
- A verified program to compute homology groups of a simplicial complex obtained from a digital image
- A discussion on different methods to overcome the efficiency problems appearing when executing programs inside proof assistants
- An application of Algebraic Topology to study biomedical images

Further work

- Efficiency issues
  - Other algorithms to compute the main objects
    - Discrete vector fields
    - Inverse of a matrix

Further work

- Efficiency issues
  - Other algorithms to compute the main objects
    - Discrete vector fields
    - Inverse of a matrix
  - Data structures and better representations
    - Work with cubical complexes
    - Data refinements could be considered

Further work

- Efficiency issues
  - Other algorithms to compute the main objects
    - Discrete vector fields
    - Inverse of a matrix
  - Data structures and better representations
    - Work with cubical complexes
    - Data refinements could be considered
  - Running environments in proof assistants

Further work

- Efficiency issues
  - Other algorithms to compute the main objects
  - Data structures and better representations
  - Running environments in proof assistants
- Formalization aspects
  - Matrices with coefficients over $\mathbb{Z}$
  - Integer homology computation

Further work

- Efficiency issues
  - Other algorithms to compute the main objects
  - Data structures and better representations
  - Running environments in proof assistants
- Formalization aspects
  - Matrices with coefficients over $\mathbb{Z}$
  - Integer homology computation
- Homology certified programs applied to biomedical cases
  - Homology group in dimension 1 (structure detection)
  - Persistent Homology (denoising)

Further work

- Efficiency issues
  - Other algorithms to compute the main objects
  - Data structures and better representations
  - Running environments in proof assistants
- Formalization aspects
  - Matrices with coefficients over $\mathbb{Z}$
  - Integer homology computation
- Homology certified programs applied to biomedical cases:
  - Homology group in dimension 1 (structure detection)
  - Persistent Homology (denoising)
- Integration between Coq and ACL2

Certifying homological algorithms to study biomedical images*

María Poza López de Echazarreta

Supervisors: Dr. César Domínguez Pérez
Dr. Julio Rubio García

Department of Mathematics and Computer Science
University of La Rioja
Spain

June 14, 2013

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