Mathematical Knowledge Management in Algebraic Topology

Jónathan Heras Vicente

Supervisors: Dr. Vico Pascual Martínez-Losa
Dr. Julio Rubio García

Department of Mathematics and Computer Science
University of La Rioja
Spain

May 31, 2011

Table of Contents

Table of Contents

Context

- Mathematical Knowledge Management (MKM)
  - Computation
  - Deduction
  - Communication

Context

- Mathematical Knowledge Management (MKM)
  - Computation
  - Deduction
  - Communication
- Algebraic Topology
  - Mathematical subject which studies topological spaces by algebraic means
  - Applications in Coding theory, Robotics, Digital Image analysis

Context

- Mathematical Knowledge Management (MKM)
  - Computation
  - Deduction
  - Communication
- Algebraic Topology
  - Mathematical subject which studies topological spaces by algebraic means
  - Applications in Coding theory, Robotics, Digital Image analysis
- Kenzo
  - Computer Algebra system devoted to Algebraic Topology developed by F. Sergeraert
  - Common Lisp package
  - Homology groups unreachable by any other means

Particularization of MKM to Algebraic Topology

- Communication
    - Human beings
    - Other programs

Particularization of MKM to Algebraic Topology

- Communication
    - Human beings
    - Other programs
- Computation
    - Kenzo

Particularization of MKM to Algebraic Topology

- Communication
    - Human beings
    - Other programs
- Computation
    - Kenzo
    - GAP

Particularization of MKM to Algebraic Topology

- Communication
  - Human beings
  - Other programs
- Computation
  - Kenzo
  - GAP
  - New modules

Particularization of MKM to Algebraic Topology

- Communication
    - Human beings
    - Other programs
- Computation
    - Kenzo
    - GAP
    - New modules
- Deduction

Particularization of MKM to Algebraic Topology

- Communication
    - Human beings
    - Other programs
- Computation
    - Kenzo
    - GAP
    - New modules
- Deduction
    - Certification of Kenzo algorithms
        - Isabelle/Hol
        - Coq

Particularization of MKM to Algebraic Topology

- Communication
    - Human beings
    - Other programs
- Computation
    - Kenzo
    - GAP
    - New modules
- Deduction
    - Certification of Kenzo algorithms
        - Isabelle/Hol
        - Coq
    - Certification of Kenzo programs
        - ACL2

Table of Contents

Motivation

$$\left.\begin{array}{l}\text{Algebraic Topology Expert}\\ \text{Common Lisp Expert}\end{array}\right\} \Rightarrow \text{Makes the most of Kenzo}$$

Motivation

$$\left.\begin{array}{l}\text{Algebraic Topology Expert}\\ \text{Common Lisp Expert}\end{array}\right\} \Rightarrow \text{Makes the most of Kenzo}$$

Non Common Lisp Expert ⇒ Unfriendly front-end

Motivation

$$\left.\begin{array}{l}\text{Algebraic Topology Expert}\\ \text{Common Lisp Expert}\end{array}\right\} \Rightarrow \text{Makes the most of Kenzo}$$

Non Common Lisp Expert ⇒ Unfriendly front-end

Non Algebraic Topology Expert ⇒ Needs guidance

Goal

- Develop a system which increases Kenzo accessibility

Goal

- Develop a system which increases Kenzo accessibility
    - Friendly front-end
    - Mediated access to Kenzo

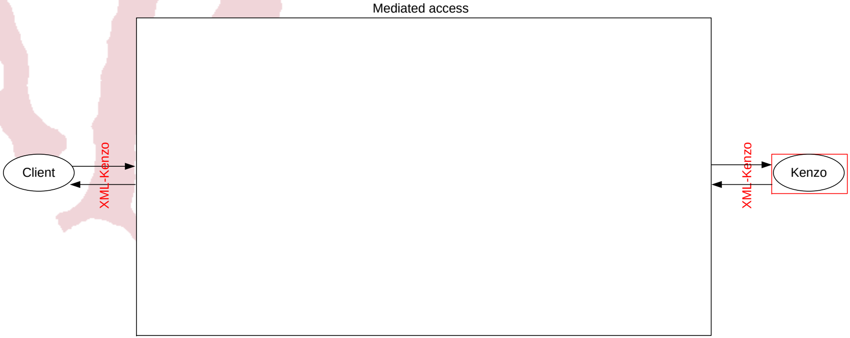
Providing a mediated access to Kenzo

Providing a mediated access to Kenzo

Representation of mathematical knowledge

$$\left.\begin{array}{l}\text{Independent of programming language}\\ \text{Easily interchangeable}\end{array}\right\} \Rightarrow \text{XML} \Rightarrow \left\{\begin{array}{l}\text{OpenMath}\\ \text{MathML}\\ \text{New language}\end{array}\right.$$

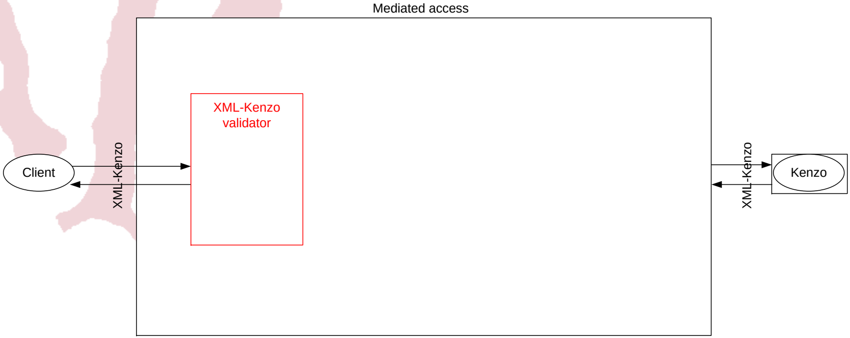
Providing a mediated access to Kenzo



XML-Kenzo

- *construction* and *computation* requests
- Includes some mathematical knowledge
  - Small Type System: CC, SS, SG, ASG
  - Some restrictions about arguments (S^n)

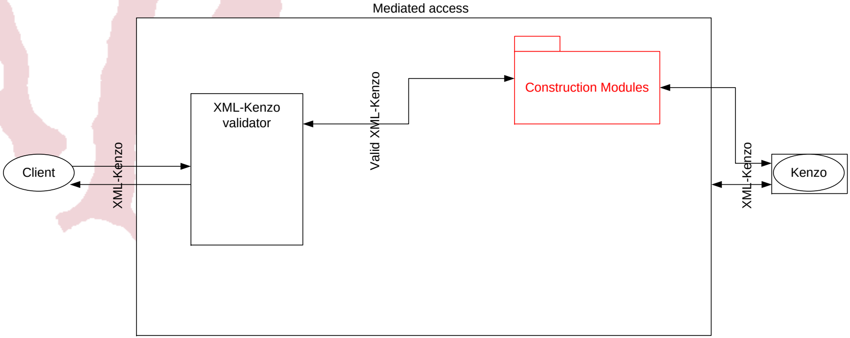
Providing a mediated access to Kenzo



XML-Kenzo validator

- Common Lisp module
- Receives XML-Kenzo requests → validates them against XML schema definition
  - Type restrictions
  - Mathematical restrictions
  - Kenzo restrictions

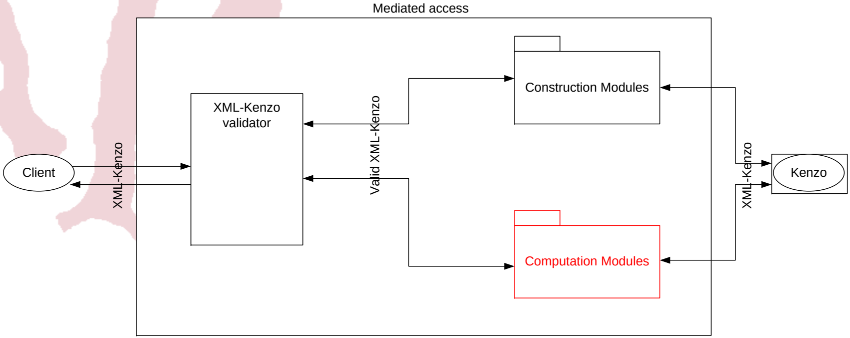
Providing a mediated access to Kenzo



Construction modules

- Process construction requests
- Check restrictions not included in XML-Kenzo (functional dependencies)
- 4 modules: CC, SS, SG and ASG

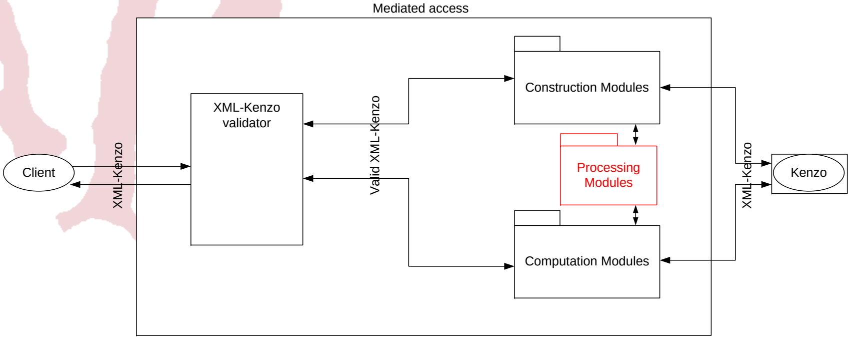
Providing a mediated access to Kenzo



Computation modules

- Process computation requests
- Check restrictions not included in XML-Kenzo (reduction degree)
- 2 modules: Homology and Homotopy

Providing a mediated access to Kenzo



Processing modules

- Help construction and computation modules
- 2 modules: Reduction degree and Homotopy assistant

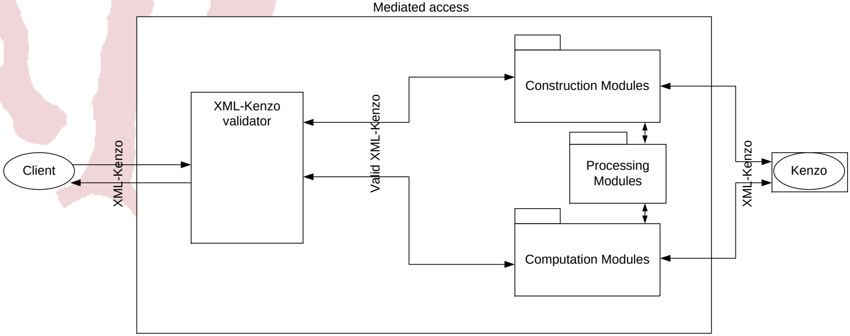
Towards the whole framework

- This architecture provides
  - Mediated access to Kenzo
  - Friendly front-end
  - but . . .

Towards the whole framework

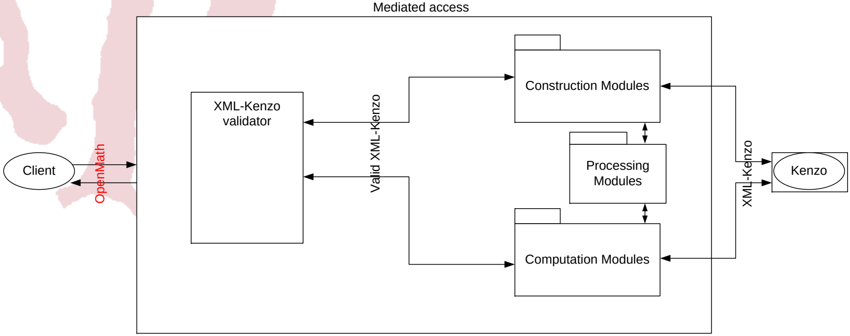
- This architecture provides
  - Mediated access to Kenzo
  - Friendly front-end
  - but . . .
- Desirable features
  - Different clients
  - Efficient
  - Extensible
  - Adaptable to different needs

Towards different clients



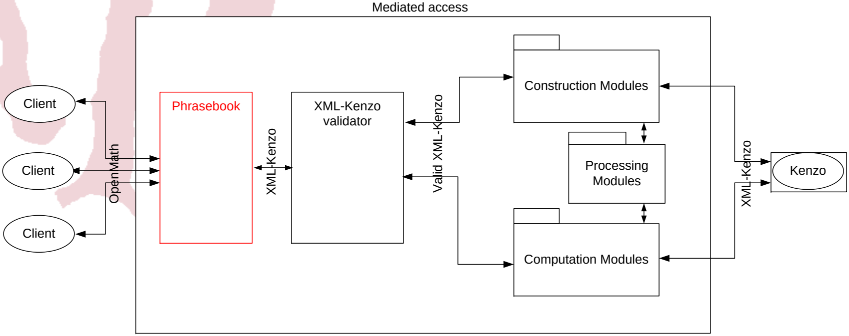
- XML-Kenzo is ad-hoc

Towards different clients



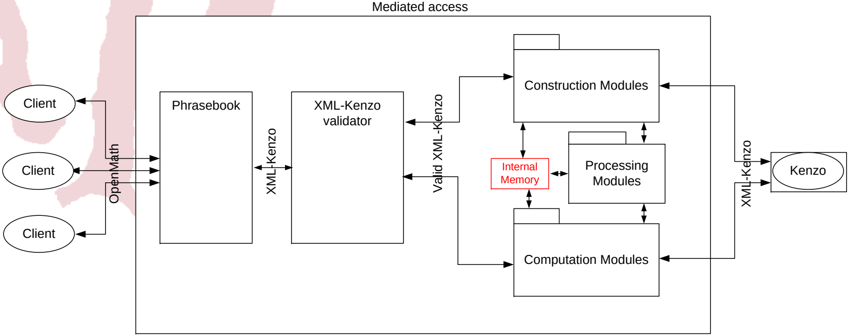
- XML-Kenzo is ad-hoc
- OpenMath
  - Standard
  - Communication with the outside world

Towards different clients



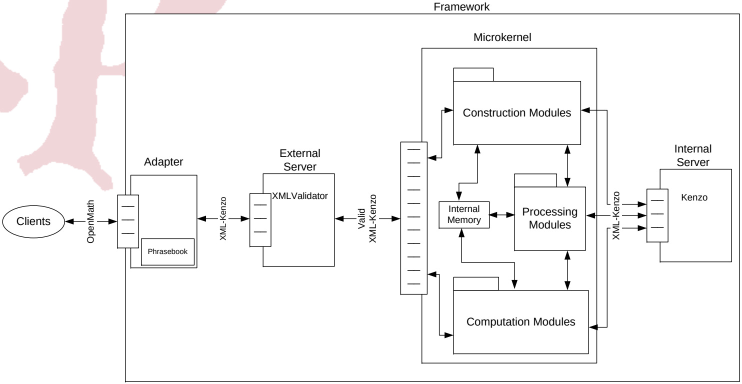
- XML-Kenzo is ad-hoc
- OpenMath
  - Standard
  - Communication with the outside world
- Phrasebook

Towards Efficiency



- System roughly equivalent to Kenzo
- Internal memory
  - Memoization technique
  - Store spaces and computations

The Kenzo framework



- Based on well-known patterns and methods

Towards extensibility

Include new functionality

Towards extensibility

Include new functionality

- Increase Kenzo functionality

Towards extensibility

Include new functionality

- Increase Kenzo functionality
- Include new systems
    - Computer Algebra systems
    - Theorem Prover tools

Towards extensibility

Include new functionality

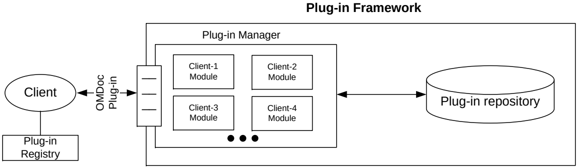
- Increase Kenzo functionality
- Include new systems
    - Computer Algebra systems
    - Theorem Prover tools
- Interoperability

Towards extensibility

Include new functionality

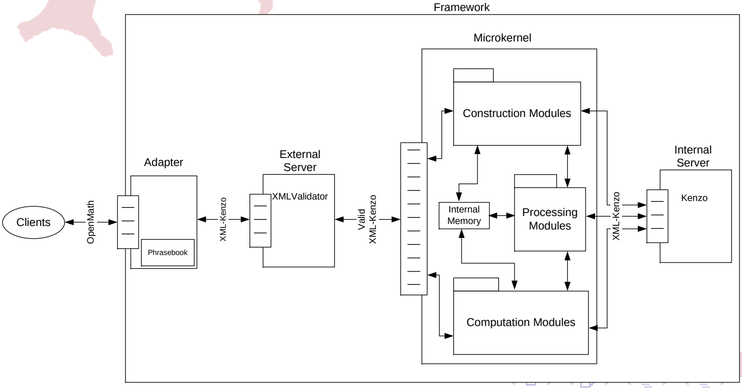
- Increase Kenzo functionality
- Include new systems
  - Computer Algebra systems
  - Theorem Prover tools
- Interoperability

Kenzo framework as a client of a plug-in framework



Including new Kenzo functionality

```
<code id="new-constructor">
   <data format="Kf/internal-server"> new-constructor-is.lisp </data>
   <data format="Kf/microkernel"> new-constructor-m.lisp </data>
   <data format="Kf/external-server"> XML-Kenzo.xsd </data>
   <data format="Kf/adapter"> new-constructor-a.lisp </data>
</code>
```



A distinguished client of our frameworks

- A Graphical User Interface implemented in Common Lisp

A distinguished client of our frameworks

- A Graphical User Interface implemented in Common Lisp
- Design decisions
  - Functionality (Common Lisp) + Structure (XUL) + Layout (stylesheet)

A distinguished client of our frameworks

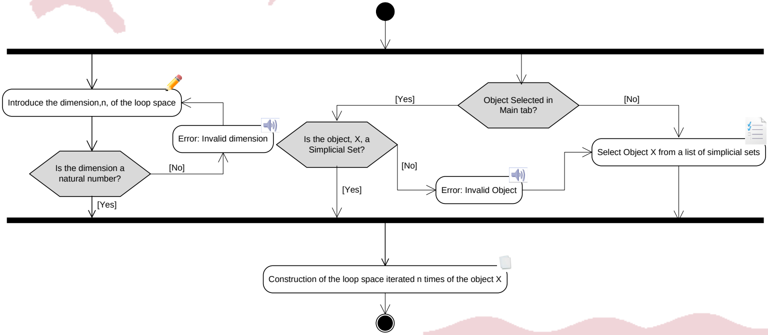
- A Graphical User Interface implemented in Common Lisp
- Design decisions
    - Functionality (Common Lisp) + Structure (XUL) + Layout (stylesheet)
    - Guided by heuristics

A distinguished client of our frameworks

- A Graphical User Interface implemented in Common Lisp
- Design decisions
  - Functionality (Common Lisp) + Structure (XUL) + Layout (stylesheet)
  - Guided by heuristics
  - Design of interaction: Noesis method

A distinguished client of our frameworks

- A Graphical User Interface implemented in Common Lisp
- Design decisions
    - Functionality (Common Lisp) + Structure (XUL) + Layout (stylesheet)
    - Guided by heuristics
    - Design of interaction: Noesis method



A customizable GUI

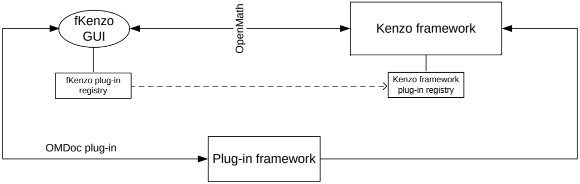
- Features
  - Extensibility
  - Adaptability

A customizable GUI

- Features
  - Extensibility
  - Adaptability
- The Graphical User Interface is
  - client of Kenzo framework
  - client of plug-in framework
    - GUI organized through modules: basic and experimental

A customizable GUI

- Features
    - Extensibility
    - Adaptability
- The Graphical User Interface is
    - client of Kenzo framework
    - client of plug-in framework
        - GUI organized through modules: basic and experimental



The whole system is called *fKenzo*

fKenzo: A tool adaptable to different needs

- Beginner student of Algebraic Topology
    - Load and remove basic modules

fKenzo: A tool adaptable to different needs

- Beginner student of Algebraic Topology
  - Load and remove basic modules
- Advanced student or researcher of Algebraic Topology
  - Load and remove basic modules
  - Load and remove experimental modules

fKenzo: A tool adaptable to different needs

- Beginner student of Algebraic Topology
  - Load and remove basic modules
- Advanced student or researcher of Algebraic Topology
  - Load and remove basic modules
  - Load and remove experimental modules
    - Digital images
    - GAP
    - ACL2
    - ...

fKenzo summary

- *fKenzo*

fKenzo summary

- *fKenzo*
  - Integral assistant for Algebraic Topology

fKenzo summary

- *fKenzo*
  - Integral assistant for Algebraic Topology
  - Constrains Kenzo functionality

fKenzo summary

- *fKenzo*
    - Integral assistant for Algebraic Topology
    - Constrains Kenzo functionality
    - Provides guidance in the interaction and a friendly front-end

fKenzo summary

- *fKenzo*
    - Integral assistant for Algebraic Topology
    - Constrains Kenzo functionality
    - Provides guidance in the interaction and a friendly front-end
    - Extensible

fKenzo summary

- *fKenzo*
    - Integral assistant for Algebraic Topology
    - Constrains Kenzo functionality
    - Provides guidance in the interaction and a friendly front-end
    - Extensible
    - Adaptable to different needs

fKenzo summary

- *fKenzo*
    - Integral assistant for Algebraic Topology
    - Constrains Kenzo functionality
    - Provides guidance in the interaction and a friendly front-end
    - Extensible
    - Adaptable to different needs
- Our architecture can be applied in other contexts

Table of Contents

New Kenzo modules

- Increase Kenzo computational capabilities
  - Simplicial Complexes
  - Digital Images
  - Pushout of Simplicial Sets

New Kenzo modules

- Increase Kenzo computational capabilities
  - Simplicial Complexes
  - Digital Images
  - Pushout of Simplicial Sets
- Integrated in $fKenzo$

New Kenzo modules

- Increase Kenzo computational capabilities
    - Simplicial Complexes
    - Digital Images
    - Pushout of Simplicial Sets
- Integrated in *fKenzo*
- Verified using ACL2

New Kenzo modules

- Increase Kenzo computational capabilities
  - Simplicial Complexes
  - Digital Images
  - Pushout of Simplicial Sets
- Integrated in *fKenzo*
- Verified using ACL2

Effective Homology preliminaries: Intuitive Idea

Compute homology groups of a space X

- $H_*(X) = H_*(C_*(X))$

Effective Homology preliminaries: Intuitive Idea

Compute homology groups of a space X

- $H_*(X) = H_*(C_*(X))$
    - $C_*(X)$ has finite nature
        - Differential maps can be expressed as integer matrices
        - Homology groups: Smith Normal Form

Effective Homology preliminaries: Intuitive Idea

Compute homology groups of a space X

- $H_*(X) = H_*(C_*(X))$
  - $C_*(X)$ has finite nature
    - Differential maps can be expressed as integer matrices
    - Homology groups: Smith Normal Form
  - $C_*(X)$ has non finite nature
    - Previous methods cannot be applied
    - Effective Homology
      - Sergeraert's ideas
      - Provides real algorithms to compute homology groups
      - Implemented in the Kenzo system

Effective Homology preliminaries: Intuitive Idea

Compute homology groups of a space X

- $H_*(X) = H_*(C_*(X))$
  - $C_*(X)$ has finite nature (Effective Chain Complex)
    - Differential maps can be expressed as integer matrices
    - Homology groups: Smith Normal Form
  - $C_*(X)$ has non finite nature (Locally Effective Chain Complex)
    - Previous methods cannot be applied
    - Effective Homology
      - Sergeraert's ideas
      - Provides real algorithms to compute homology groups
      - Implemented in the Kenzo system

Effective Homology preliminaries

Definition

An effective chain complex is a free chain complex of $\mathbb{Z}$-modules, $C_* = (C_n, d_n)_{n\in\mathbb{N}}$, where each group C_n is finitely generated and

- *an algorithm returns a $\mathbb{Z}$-base in each grade n*
- *an algorithm provides the differentials d_n*

Definition

A locally effective chain complex is a free chain complex of $\mathbb{Z}$-modules $C_* = (C_n, d_n)_{n\in\mathbb{N}}$ where each group C_n can have infinite nature, but there exists an algorithm such that $\forall x \in C_n$, we can compute $d_n(x)$

Effective Homology preliminaries

Definition

An effective chain complex is a free chain complex of $\mathbb{Z}$-modules, $C_* = (C_n, d_n)_{n \in \mathbb{N}}$, where each group C_n is finitely generated and

- *an algorithm returns a $\mathbb{Z}$-base in each grade n*
- *an algorithm provides the differentials d_n*

- differentials $d_n : C_n \to C_{n-1}$ can be expressed as integer matrices
- possible to compute *Ker* d_n and *Im* d_{n+1}
- possible to compute the homology groups

Definition

A locally effective chain complex is a free chain complex of $\mathbb{Z}$-modules $C_* = (C_n, d_n)_{n \in \mathbb{N}}$ where each group C_n can have infinite nature, but there exists an algorithm such that $\forall x \in C_n$, we can compute $d_n(x)$

- impossible to compute *Ker* d_n and *Im* d_{n+1}
- possible to perform local computations, differential of a generator

Effective Homology preliminaries

Definition

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

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

satisfying the following relations

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

Effective Homology preliminaries

Definition

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

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

satisfying the following relations

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

Theorem

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

Effective Homology preliminaries

Definition

A (strong chain) equivalence ε *between* C_* *and* D_*, $\varepsilon : C_* \Leftarrow\!\!\!\Rightarrow D_*$, *is a triple* $\varepsilon = (B_*, \rho, \rho')$ *where* B_* *is a chain complex,* $\rho : B_* \Rightarrow C_*$ *and* $\rho' : B_* \Rightarrow D_*$.

$$\begin{tikzcd} & B_* \arrow[dl, Rightarrow] \arrow[dr, Rightarrow] & \\ C_* & & D_* \end{tikzcd}$$

$$\begin{tikzcd} & \frac{42}{30} \arrow[dl, Rightarrow] \arrow[dr, Rightarrow] & \\ \frac{14}{10} & & \frac{21}{15} \end{tikzcd}$$

Effective Homology preliminaries

Definition

A (strong chain) equivalence ε *between* C_* *and* D_*, $\varepsilon : C_* \Leftrightarrow D_*$, *is a triple* $\varepsilon = (B_*, \rho, \rho')$ *where* B_* *is a chain complex,* $\rho : B_* \Rightarrow C_*$ *and* $\rho' : B_* \Rightarrow D_*$.

$$\begin{tikzcd} & B_* \arrow[dl, Rightarrow] \arrow[dr, Rightarrow] & \\ C_* & & D_* \end{tikzcd}$$

$$\begin{tikzcd} & \frac{42}{30} \arrow[dl, Rightarrow] \arrow[dr, Rightarrow] & \\ \frac{14}{10} & & \frac{21}{15} \end{tikzcd}$$

Definition

An object with effective homology is a quadruple $(X, C_*(X), HC_*, \varepsilon)$ *where*

- X *is a locally effective object*
- $C_*(X)$ *is a (locally effective) chain complex canonically associated with* X, *which allows the study of the homological nature of* X
- HC_* *is an effective chain complex*
- ε *is a equivalence* $\varepsilon : C_*(X) \Leftrightarrow HC_*$

Effective Homology preliminaries

Definition

A (strong chain) equivalence ε *between* C_* *and* D_*, $\varepsilon : C_* \Leftrightarrow D_*$, *is a triple* $\varepsilon = (B_*, \rho, \rho')$ *where* B_* *is a chain complex,* $\rho : B_* \Rightarrow C_*$ *and* $\rho' : B_* \Rightarrow D_*$.

$$\begin{tikzcd} & B_* \arrow[dl, Rightarrow] \arrow[dr, Rightarrow] & \\ C_* & & D_* \end{tikzcd}$$

$$\begin{tikzcd} & \frac{42}{30} \arrow[dl, Rightarrow] \arrow[dr, Rightarrow] & \\ \frac{14}{10} & & \frac{21}{15} \end{tikzcd}$$

Definition

An object with effective homology is a quadruple $(X, C_*(X), HC_*, \varepsilon)$ *where*

- X *is a locally effective object*
- $C_*(X)$ *is a (locally effective) chain complex canonically associated with* X, *which allows the study of the homological nature of* X
- HC_* *is an effective chain complex*
- ε *is a equivalence* $\varepsilon : C_*(X) \Leftrightarrow HC_*$

Theorem

Let an object with effective homology $(X, C_*(X), HC_*, \varepsilon)$ *then* $H_n(X) \cong H_n(HC_*)$ *for all* n.

Pushout preliminaries

Definition

Let f, g morphisms,

$$\begin{tikzcd}
X \arrow[r, "f"] \arrow[d, "g"] & Y \\
Z &
\end{tikzcd}$$

Pushout preliminaries

Definition

Let f, g morphisms, the pushout of f, g is an object $P_{(f,g)}$ for which the diagram

$$\begin{tikzcd}
X \arrow[r, "f"] \arrow[d, "g"] & Y \arrow[d, "f'"] \\
Z \arrow[r, "g'"] & P_{(f,g)}
\end{tikzcd}$$

- *commutes*

Pushout preliminaries

Definition

Let f, g morphisms, the pushout of f, g is an object $P_{(f,g)}$ for which the diagram

$$\begin{tikzcd}
X \arrow[r, "f"] \arrow[d, "g"] & Y \arrow[d, "f'"] \arrow[ddr, bend left, "f''"] & \\
Z \arrow[r, "g'"] \arrow[drr, bend right, "g''"'] & P_{(f,g)} \arrow[dr, dashed, "\exists! p"] & \\
& & Q
\end{tikzcd}$$

- *commutes*
- *respects the universal property*

Pushout preliminaries

Definition

Let f, g morphisms, the pushout of f, g is an object $P_{(f,g)}$ for which the diagram

$$\begin{tikzcd}
X \arrow[r, "f"] \arrow[d, "g"] & Y \arrow[d, "f'"] \arrow[ddr, bend left, "f''"] & \\
Z \arrow[r, "g'"] \arrow[drr, bend right, "g''"'] & P_{(f,g)} \arrow[dr, dashed, "\exists ! p"] & \\
 & & Q
\end{tikzcd}$$

- *commutes*
- *respects the universal property*

Standard Construction

$P_{(f,g)} \cong (Y \amalg (X \times I) \amalg Z)/\sim$ *where*

- *I is the unit interval*
- *for every $x \in X$, $\sim$*
  - $(x, 0) \sim f(x) \in Y$
  - $(x, 1) \sim g(x) \in Z$

Construction of the Effective Homology of the Pushout I

Given $f : X \to Y$ and $g : X \to Z$ simplicial morphisms where X, Y and Z are simplicial sets

$$\begin{tikzcd}
X \arrow[r, "f"] \arrow[d, "g"] & Y \\
Z &
\end{tikzcd}$$

Construction of the Effective Homology of the Pushout I

Given $f : X \rightarrow Y$ and $g : X \rightarrow Z$ simplicial morphisms where X, Y and Z are simplicial sets

$$\begin{tikzcd} X \arrow[r, "f"] \arrow[d, "g"] & Y \arrow[d, "f'"] \\ Z \arrow[r, "g'"] & P_{(f,g)} \end{tikzcd}$$

Construction of the Effective Homology of the Pushout I

Given $f : X \to Y$ and $g : X \to Z$ simplicial morphisms where X, Y and Z are simplicial sets

$$\begin{tikzcd} X \arrow[r, "f"] \arrow[d, "g"] & Y \arrow[d, "f'"] \\ Z \arrow[r, "g'"] & P_{(f,g)} \end{tikzcd}$$

Algorithm (Standard Construction)

Input: two simplicial morphisms $f : X \to Y$ and $g : X \to Z$ where X, Y and Z are simplicial sets
Output: the pushout $P_{(f,g)}$, a simplicial set

Construction of the Effective Homology of the Pushout II

Given $f : X \to Y$ and $g : X \to Z$ simplicial morphisms where X, Y and Z are simplicial sets with effective homology

$$\begin{tikzcd}
(X, C_*(X), HX_*, \varepsilon_X) \arrow[r, "f"] \arrow[d, "g"] & (Y, C_*(Y), HY_*, \varepsilon_Y) \\
(Z, C_*(Z), HZ_*, \varepsilon_Z) &
\end{tikzcd}$$

Construction of the Effective Homology of the Pushout II

Given $f : X \to Y$ and $g : X \to Z$ simplicial morphisms where X, Y and Z are simplicial sets with effective homology

$$\begin{tikzcd}
(X, C_*(X), HX_*, \varepsilon_X) \arrow[r, "f"] \arrow[d, "g"] & (Y, C_*(Y), HY_*, \varepsilon_Y) \arrow[d] \\
(Z, C_*(Z), HZ_*, \varepsilon_Z) \arrow[r] & (P_{(f,g)}, C_*(P_{(f,g)}), \quad , \quad)
\end{tikzcd}$$

Construction of the Effective Homology of the Pushout II

Given $f : X \to Y$ and $g : X \to Z$ simplicial morphisms where X, Y and Z are simplicial sets with effective homology

$$\begin{tikzcd}
(X, C_*(X), HX_*, \varepsilon_X) \arrow[r, "f"] \arrow[d, "g"] & (Y, C_*(Y), HY_*, \varepsilon_Y) \arrow[d] \\
(Z, C_*(Z), HZ_*, \varepsilon_Z) \arrow[r] & (P_{(f,g)}, C_*(P_{(f,g)}), HP_*, \varepsilon_P)
\end{tikzcd}$$

Construction of the Effective Homology of the Pushout II

Given $f : X \rightarrow Y$ and $g : X \rightarrow Z$ simplicial morphisms where X, Y and Z are simplicial sets with effective homology

$$\begin{tikzcd}
(X, C_*(X), HX_*, \varepsilon_X) \arrow[r, "f"] \arrow[d, "g"] & (Y, C_*(Y), HY_*, \varepsilon_Y) \arrow[d] \\
(Z, C_*(Z), HZ_*, \varepsilon_Z) \arrow[r] & (P_{(f,g)}, C_*(P_{(f,g)}), HP_*, \varepsilon_P)
\end{tikzcd}$$

Algorithm (joint work with F. Sergeraert)

Input: two simplicial morphisms $f : X \rightarrow Y$ and $g : X \rightarrow Z$ where X, Y and Z are simplicial sets with effective homology
Output: the effective homology version of $P_{(f,g)}$, that is, an equivalence $C_*(P_{(f,g)}) \Leftrightarrow HP_*$, where HP_* is an effective chain complex

Sketch of the algorithm

Theorem (SES Theorems)

Let

$$\begin{tikzcd}
0 & A_* \arrow[l, "0"'] \arrow[r, shift left, "\sigma"] & B_* \arrow[l, shift left, "j"] \arrow[r, shift left, "\rho"] & C_* \arrow[l, shift left, "i"] & 0 \arrow[l]
\end{tikzcd}$$

be an effective short exact sequence of chain complexes. Then three general algorithms are available

$$\begin{array}{l}
SES_1 : (B_{*,EH}, C_{*,EH}) \mapsto A_{*,EH} \\
SES_2 : (A_{*,EH}, C_{*,EH}) \mapsto B_{*,EH} \\
SES_3 : (A_{*,EH}, B_{*,EH}) \mapsto C_{*,EH}
\end{array}$$

producing the effective homology of one chain complex when the effective homology of both others is given.

Sketch of the algorithm

Theorem (SES Theorems)

Let

$$\begin{tikzcd} 0 & A_* \arrow[l, "0"'] \arrow[r, shift left, "\sigma"] & B_* \arrow[l, shift left, "j"] \arrow[r, shift left, "\rho"] & C_* \arrow[l, shift left, "i"] & 0 \arrow[l] \end{tikzcd}$$

be an effective short exact sequence of chain complexes. Then three general algorithms are available

$$\begin{array}{l} SES_1 : (B_{*,EH}, C_{*,EH}) \mapsto A_{*,EH} \\ SES_2 : (A_{*,EH}, C_{*,EH}) \mapsto B_{*,EH} \\ SES_3 : (A_{*,EH}, B_{*,EH}) \mapsto C_{*,EH} \end{array}$$

producing the effective homology of one chain complex when the effective homology of both others is given.

$$\begin{tikzcd} 0 & M_* \arrow[l] \arrow[r, shift left, "\sigma"] & C_*P \arrow[l, shift left, "j"] \arrow[r, shift left, "\rho"] & C_*Y \oplus C_*Z \arrow[l, shift left, "i"] & 0 \arrow[l] \end{tikzcd}$$

M_* is the chain complex associated with $X \times \Delta^1$ but with the simplexes of $X \times (0)$ and $X \times (1)$ cancelled

Sketch of the algorithm continued

Step 1. From $f : X \rightarrow Y$ and $g : X \rightarrow Z$ simplicial morphisms, $P_{(f,g)}$ and its associated chain complex $C_* P$ are constructed

Step 2. The effective homology of M_* is constructed

  - Define

$$\begin{tikzcd} 0 & M_* \arrow[l] \arrow[r, shift left, "\sigma 2"] & C_*(X \times \Delta^1) \arrow[l, shift left, "j2"] \arrow[r, shift left, "\rho 2"] & C_*(X \times (0)) \oplus C_*(X \times (1)) \arrow[l, shift left, "i2"] & 0 \arrow[l] \end{tikzcd}$$

  - Construct the chain complex M_*
  - Build the effective homology of $C_*(X \times \Delta^1)$
  - Construct the effective homology of $C_*(X \times (0)) \oplus C_*(X \times (1))$
  - Construct the effective homology of $Cone(i2)$
  - Construct the reduction $M_* \Lleftarrow Cone(i2)$
  - Obtain the effective homology of M_* applying case SES_1

Step 3. The effective homology of $C_* X \oplus C_* Y$ is constructed

Step 4. The effective homology of the pushout $P_{(f,g)}$ is constructed

  - Define

$$\begin{tikzcd} 0 & M_* \arrow[l] \arrow[r, shift left, "\sigma"] & C_* P \arrow[l, shift left, "j"] \arrow[r, shift left, "\rho"] & C_* Y \oplus C_* Z \arrow[l, shift left, "i"] & 0 \arrow[l] \end{tikzcd}$$

  - Construct the effective homology of $(C_* Y \oplus C_* Z)^{[1]}$
  - Define the morphism *shift* : $C_* Y \oplus C_* Z \rightarrow (C_* Y \oplus C_* Z)^{[1]}$
  - Define the chain complex morphism $\chi : M_* \rightarrow (C_* Y \oplus C_* Z)^{[1]}$
  - Construct the effective homology of $Cone(\chi)$
  - Obtain the effective homology of $P_{(f,g)}$ applying case SES_2

Example: $\pi_*(\Sigma(SL_2(\mathbb{Z})))$

$SL_2(\mathbb{Z})$

- Group of 2×2 matrices with determinant 1 over $\mathbb{Z}$

Example: $\pi_*(\Sigma(SL_2(\mathbb{Z})))$

$SL_2(\mathbb{Z})$

- Group of 2×2 matrices with determinant 1 over $\mathbb{Z}$
- Isomorphic to $\mathbb{Z}_4 *_{\mathbb{Z}_2} \mathbb{Z}_6$

📕 J. P. Serre. *Trees*. Springer-Verlag, 1980

Example: $\pi_*(\Sigma(SL_2(\mathbb{Z})))$

$SL_2(\mathbb{Z})$

- Group of 2×2 matrices with determinant 1 over $\mathbb{Z}$
- Isomorphic to $\mathbb{Z}_4 *_{\mathbb{Z}_2} \mathbb{Z}_6$

  J. P. Serre. *Trees*. Springer-Verlag, 1980

- $$\begin{tikzcd} K(\mathbb{Z}_2,1) \arrow[r, "i_1"] \arrow[d, "i_2"] & K(\mathbb{Z}_4,1) \arrow[d] \\ K(\mathbb{Z}_6,1) \arrow[r] & K(SL_2(\mathbb{Z}),1) \end{tikzcd}$$

  K. S. Brown. *Cohomology of Groups*. Springer-Verlag, 1982

Example: $\pi_*(\Sigma(SL_2(\mathbb{Z})))$

Construction of the object $K(SL_2(\mathbb{Z},1))$

Example: $\pi_*(\Sigma(SL_2(\mathbb{Z})))$

Construction of the object $K(SL_2(\mathbb{Z},1))$

- Firstly, we define $K(\mathbb{Z}_2,1), K(\mathbb{Z}_4,1)$ and $K(\mathbb{Z}_6,1)$

```
> (setf kz2 (k-zp-1 2)) ✠
[K2 Abelian-Simplicial-Group]
> (setf kz4 (k-zp-1 4)) ✠
[K15 Abelian-Simplicial-Group]
> (setf kz6 (k-zp-1 6)) ✠
[K28 Abelian-Simplicial-Group]
```

Example: $\pi_*(\Sigma(SL_2(\mathbb{Z})))$

Construction of the object $K(SL_2(\mathbb{Z},1))$

- Firstly, we define $K(\mathbb{Z}_2,1)$, $K(\mathbb{Z}_4,1)$ and $K(\mathbb{Z}_6,1)$

```
> (setf kz2 (k-zp-1 2)) ✠
[K2 Abelian-Simplicial-Group]
> (setf kz4 (k-zp-1 4)) ✠
[K15 Abelian-Simplicial-Group]
> (setf kz6 (k-zp-1 6)) ✠
[K28 Abelian-Simplicial-Group]
```

- Subsequently, we define the two simplicial morphisms i_1 and i_2

```
> (setf i1 (kzps-incl 2 4)) ✠
[K40 Simplicial-Morphism K2 -> K15]
> (setf i2 (kzps-incl 2 6)) ✠
[K41 Simplicial-Morphism K2 -> K28]
```

Example: $\pi_*(\Sigma(SL_2(\mathbb{Z})))$

Construction of the object $K(SL_2(\mathbb{Z}, 1))$

- Firstly, we define $K(\mathbb{Z}_2, 1)$, $K(\mathbb{Z}_4, 1)$ and $K(\mathbb{Z}_6, 1)$

```
> (setf kz2 (k-zp-1 2)) ✠
[K2 Abelian-Simplicial-Group]
> (setf kz4 (k-zp-1 4)) ✠
[K15 Abelian-Simplicial-Group]
> (setf kz6 (k-zp-1 6)) ✠
[K28 Abelian-Simplicial-Group]
```

- Subsequently, we define the two simplicial morphisms i_1 and i_2

```
> (setf i1 (kzps-incl 2 4)) ✠
[K40 Simplicial-Morphism K2 -> K15]
> (setf i2 (kzps-incl 2 6)) ✠
[K41 Simplicial-Morphism K2 -> K28]
```

- Finally, we construct the pushout of i_1 and i_2 $(K(SL_2(\mathbb{Z}), 1))$

```
> (setf ksl2z (pushout i1 i2)) ✠
[K52 Simplicial-Set]
```

Example: $\pi_*(\Sigma(SL_2(\mathbb{Z})))$

Computation of $\pi_*(\Sigma(SL_2(\mathbb{Z})))$

Example: $\pi_*(\Sigma(SL_2(\mathbb{Z})))$

Computation of $\pi_*(\Sigma(SL_2(\mathbb{Z})))$

- Firstly, we define $\Sigma(K(SL_2(\mathbb{Z}),1))$

```
> (setf sksl2z (suspension ksl2z))
[K62 Simplicial-Set]
```

Example: $\pi_*(\Sigma(SL_2(\mathbb{Z})))$

Computation of $\pi_*(\Sigma(SL_2(\mathbb{Z})))$

- Firstly, we define $\Sigma(K(SL_2(\mathbb{Z}),1))$

```
> (setf sksl2z (suspension ksl2z)) ✠
[K62 Simplicial-Set]
```

- Computing Homotopy groups (Hurewicz theorem)

```
> (homology sksl2z 1 3) ✠
Homology in dimension 1:

Homology in dimension 2:
Component Z/12Z
```

Example: $\pi_*(\Sigma(SL_2(\mathbb{Z})))$

Computation of $\pi_*(\Sigma(SL_2(\mathbb{Z})))$

- Firstly, we define $\Sigma(K(SL_2(\mathbb{Z}),1))$

```
> (setf sksl2z (suspension ksl2z))
[K62 Simplicial-Set]
```

- Computing Homotopy groups (Hurewicz theorem)

```
> (homology sksl2z 1 3)
Homology in dimension 1:

Homology in dimension 2:
Component Z/12Z
```

- $\pi_1(\Sigma(SL_2(\mathbb{Z}))) = 0$, $\pi_2(\Sigma(SL_2(\mathbb{Z}))) = \mathbb{Z}/12\mathbb{Z}$

Example: $\pi_*(\Sigma(SL_2(\mathbb{Z})))$

- Computing Homotopy groups continued (Whitehead tower)

```
> (setf tau (zp-whitehead 12 sksl2z (chml-clss sksl2z 2))) ✠
[K91 Fibration K62 -> K79]
> (setf x (fibration-total tau)) ✠
[K97 Simplicial-Set]
> (homology x 3)
Homology in dimension 3:
Component Z/12Z
> (setf tau2 (zp-whitehead 12 x (chml-clss x 3))) ✠
[K228 Fibration K97 -> K214]
> (setf x2 (fibration-total tau2)) ✠
[K234 Simplicial-Set]
> (homology x2 4)
Homology in dimension 4:
Component Z/12Z
Component Z/2Z
> ...
```

Example: $\pi_*(\Sigma(SL_2(\mathbb{Z})))$

- Computing Homotopy groups continued (Whitehead tower)

```
> (setf tau (zp-whitehead 12 sksl2z (chml-clss sksl2z 2))) ✠
[K91 Fibration K62 -> K79]
> (setf x (fibration-total tau)) ✠
[K97 Simplicial-Set]
> (homology x 3)
Homology in dimension 3:
Component Z/12Z
> (setf tau2 (zp-whitehead 12 x (chml-clss x 3))) ✠
[K228 Fibration K97 -> K214]
> (setf x2 (fibration-total tau2)) ✠
[K234 Simplicial-Set]
> (homology x2 4)
Homology in dimension 4:
Component Z/12Z
Component Z/2Z
> ...
```

- $\pi_3(\Sigma(SL_2(\mathbb{Z}))) = \mathbb{Z}/12\mathbb{Z}$, $\pi_4(\Sigma(SL_2(\mathbb{Z}))) = \mathbb{Z}/12\mathbb{Z} \oplus \mathbb{Z}/2\mathbb{Z}$, ...

Table of Contents

ACL2

- ACL2 (A Computational Logic for an Applicative Common Lisp)

ACL2

- ACL2 (A Computational Logic for an Applicative Common Lisp)
  - Programming Language
  - First-Order Logic
  - Theorem Prover

ACL2

- ACL2 (A Computational Logic for an Applicative Common Lisp)
  - Programming Language
  - First-Order Logic
  - Theorem Prover
- Proof techniques
  - Simplification
  - Induction
  - *"The Method"*

ACL2

- ACL2 (A Computational Logic for an Applicative Common Lisp)
  - Programming Language
  - First-Order Logic
  - Theorem Prover
- Proof techniques
  - Simplification
  - Induction
  - *"The Method"*
- Encapsulate principle
  - Simulation of Higher-Order Logic

Goal

Simplified view of Kenzo way of working

1. Construction of initial spaces
2. Construction of new spaces from other ones
3. Computation of homology groups

Goal

Simplified view of Kenzo way of working

1. Construction of initial spaces
    - Certification of the correctness of Kenzo constructors of simplicial sets
2. Construction of new spaces from other ones
    - Easy Perturbation Lemma
    - Composition of reductions
    - Cone Equivalence Theorem
    - ...
3. Computation of homology groups

```
> (sphere 3) ✠
[K1 Simplicial-Set]
```

Mathematical context: Simplicial Sets

Definition

A simplicial set K*, is a union* $K = \bigcup_{q \geq 0} K^q$*, where the* K^q *are disjoints sets, together with functions*

$$\begin{array}{lll} \partial_i^q : K^q \rightarrow K^{q-1}, & q > 0, & i = 0, \dots, q, \\ \eta_i^q : K^q \rightarrow K^{q+1}, & q \geq 0, & i = 0, \dots, q, \end{array}$$

subject to the relations

$$\begin{array}{llllll} (1) & \partial_i^{q-1}\partial_j^q & = & \partial_{j-1}^{q-1}\partial_i^q & \textit{if} & i < j, \\ (2) & \eta_i^{q+1}\eta_j^q & = & \eta_{j+1}^{q+1}\eta_i^q & \textit{if} & i \leq j, \\ (3) & \partial_i^{q+1}\eta_j^q & = & \eta_{j-1}^{q-1}\partial_i^q & \textit{if} & i < j, \\ (4) & \partial_i^{q+1}\eta_i^q & = & \textit{identity} & = & \partial_{i+1}^{q+1}\eta_i^q, \\ (5) & \partial_i^{q+1}\eta_j^q & = & \eta_j^{q-1}\partial_{i-1}^q & \textit{if} & i > j+1, \end{array}$$

Mathematical context: Simplicial Sets

Definition

A simplicial set K*, is a union* $K = \bigcup_{q \geq 0} K^q$*, where the* K^q *are disjoints sets, together with functions*

$$\begin{array}{lll} \partial_i^q : K^q \to K^{q-1}, & q > 0, & i = 0, \ldots, q, \\ \eta_i^q : K^q \to K^{q+1}, & q \geq 0, & i = 0, \ldots, q, \end{array}$$

subject to the relations

$$\begin{array}{llllll} (1) & \partial_i^{q-1}\partial_j^q & = & \partial_{j-1}^{q-1}\partial_i^q & if & i < j, \\ (2) & \eta_i^{q+1}\eta_j^q & = & \eta_{j+1}^{q+1}\eta_i^q & if & i \leq j, \\ (3) & \partial_i^{q+1}\eta_j^q & = & \eta_{j-1}^{q-1}\partial_i^q & if & i < j, \\ (4) & \partial_i^{q+1}\eta_i^q & = & identity & = & \partial_{i+1}^{q+1}\eta_i^q, \\ (5) & \partial_i^{q+1}\eta_j^q & = & \eta_j^{q-1}\partial_{i-1}^q & if & i > j+1, \end{array}$$

- The elements of K^q are called q-simplexes

Mathematical context: Simplicial Sets

Definition

A simplicial set K, is a union $K = \bigcup_{q \geq 0} K^q$, where the K^q are disjoints sets, together with functions

$$\begin{array}{lll} \partial_i^q : K^q \rightarrow K^{q-1}, & q > 0, & i = 0, \ldots, q, \\ \eta_i^q : K^q \rightarrow K^{q+1}, & q \geq 0, & i = 0, \ldots, q, \end{array}$$

subject to the relations

$$\begin{array}{llllll} (1) & \partial_i^{q-1}\partial_j^q & = & \partial_{j-1}^{q-1}\partial_i^q & \textit{if} & i < j, \\ (2) & \eta_i^{q+1}\eta_j^q & = & \eta_{j+1}^{q+1}\eta_i^q & \textit{if} & i \leq j, \\ (3) & \partial_i^{q+1}\eta_j^q & = & \eta_{j-1}^{q-1}\partial_i^q & \textit{if} & i < j, \\ (4) & \partial_i^{q+1}\eta_i^q & = & \textit{identity} & = & \partial_{i+1}^{q+1}\eta_i^q, \\ (5) & \partial_i^{q+1}\eta_j^q & = & \eta_j^{q-1}\partial_{i-1}^q & \textit{if} & i > j+1, \end{array}$$

- The elements of K^q are called q-simplexes
- A q-simplex x is degenerate if $x = \eta_i^{q-1} y$ for some simplex $y \in K^{q-1}$

Mathematical context: Simplicial Sets

Definition

A simplicial set K, is a union $K = \bigcup_{q \geq 0} K^q$, where the K^q are disjoints sets, together with functions

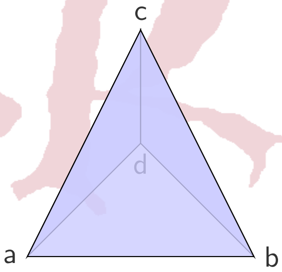
$$\begin{array}{llll} \partial_i^q : K^q \rightarrow K^{q-1}, & q > 0, & i = 0, \ldots, q, \\ \eta_i^q : K^q \rightarrow K^{q+1}, & q \geq 0, & i = 0, \ldots, q, \end{array}$$

subject to the relations

$$\begin{array}{llllll} (1) & \partial_i^{q-1}\partial_j^q & = & \partial_{j-1}^{q-1}\partial_i^q & \mathit{if} & i < j, \\ (2) & \eta_i^{q+1}\eta_j^q & = & \eta_{j+1}^{q+1}\eta_i^q & \mathit{if} & i \leq j, \\ (3) & \partial_i^{q+1}\eta_j^q & = & \eta_{j-1}^{q-1}\partial_i^q & \mathit{if} & i < j, \\ (4) & \partial_i^{q+1}\eta_i^q & = & \mathit{identity} & = & \partial_{i+1}^{q+1}\eta_i^q, \\ (5) & \partial_i^{q+1}\eta_j^q & = & \eta_j^{q-1}\partial_{i-1}^q & \mathit{if} & i > j+1, \end{array}$$

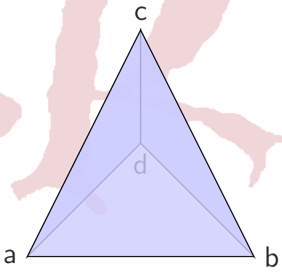
- The elements of K^q are called q-simplexes
- A q-simplex x is degenerate if $x = \eta_i^{q-1} y$ for some simplex $y \in K^{q-1}$
- Otherwise x is called non-degenerate (or geometric)

Mathematical context: Example



- 0-simplexes: vertices: $(a), (b), (c), (d)$
- non-degenerate 1-simplexes: edges: $(a\ b)$,$(a\ c)$,$(a\ d)$,$(b\ c)$,$(b\ d)$,$(c\ d)$
- non-degenerate 2-simplexes: (filled) triangles: $(a\ b\ c)$,$(a\ b\ d)$,$(a\ c\ d)$,$(b\ c\ d)$
- non-degenerate 3-simplexes: (filled) tetrahedron: $(a\ b\ c\ d)$

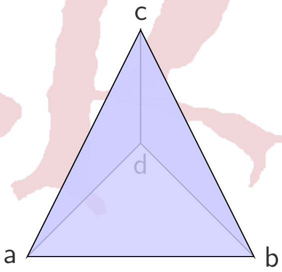
Mathematical context: Example



- 0-simplexes: vertices: $(a), (b), (c), (d)$
- non-degenerate 1-simplexes: edges: $(a\ b)$,$(a\ c)$,$(a\ d)$,$(b\ c)$,$(b\ d)$,$(c\ d)$
- non-degenerate 2-simplexes: (filled) triangles: $(a\ b\ c)$,$(a\ b\ d)$,$(a\ c\ d)$,$(b\ c\ d)$
- non-degenerate 3-simplexes: (filled) tetrahedron: $(a\ b\ c\ d)$

$$\text{face: } \partial_i(a\ b\ c) = \begin{cases} (b\ c) & \text{if } i = 0 \\ (a\ c) & \text{if } i = 1 \\ (a\ b) & \text{if } i = 2 \end{cases}$$

Mathematical context: Example



- 0-simplexes: vertices: $(a), (b), (c), (d)$
- non-degenerate 1-simplexes: edges: $(a\ b)$,$(a\ c)$,$(a\ d)$,$(b\ c)$,$(b\ d)$,$(c\ d)$
- non-degenerate 2-simplexes: (filled) triangles: $(a\ b\ c)$,$(a\ b\ d)$,$(a\ c\ d)$,$(b\ c\ d)$
- non-degenerate 3-simplexes: (filled) tetrahedron: $(a\ b\ c\ d)$

$$\text{face: } \partial_i(a\ b\ c) = \begin{cases} (b\ c) & \text{if } i = 0 \\ (a\ c) & \text{if } i = 1 \\ (a\ b) & \text{if } i = 2 \end{cases}$$

$$\text{degeneracy: } \eta_i(a\ b\ c) = \left\{ \begin{array}{ll} (a\ a\ b\ c) & \text{if } i = 0 \\ (a\ b\ b\ c) & \text{if } i = 1 \\ (a\ b\ c\ c) & \text{if } i = 2 \end{array} \right\} \text{ non-geometrical meaning}$$

Mathematical context: abstract simplexes

Proposition

Let K be a simplicial set. Any n-simplex $x \in K^n$ can be expressed in a unique way as a (possibly) iterated degeneracy of a non-degenerate simplex y in the following way

$$x = \eta_{j_k} \dots \eta_{j_1} y$$

with $y \in K^r$, $k = n - r \geq 0$, and $0 \leq j_1 < \cdots < j_k < n$.

Mathematical context: abstract simplexes

Proposition

Let K be a simplicial set. Any n-simplex $x \in K^n$ can be expressed in a unique way as a (possibly) iterated degeneracy of a non-degenerate simplex y in the following way

$$x = \eta_{j_k} \dots \eta_{j_1} y$$

with $y \in K^r$, $k = n - r \geq 0$, *and* $0 \leq j_1 < \dots < j_k < n$.

- *abstract simplex*
  - $(dgop \ \ gmsm) := \begin{cases} \text{dgop is a strictly decreasing sequence of degeneracy maps} \\ \text{gmsm is a geometric simplex} \end{cases}$

Mathematical context: abstract simplexes

Proposition

Let K be a simplicial set. Any n-simplex $x \in K^n$ can be expressed in a unique way as a (possibly) iterated degeneracy of a non-degenerate simplex y in the following way

$$x = \eta_{j_k} \dots \eta_{j_1} y$$

with $y \in K^r$, $k = n - r \geq 0$, and $0 \leq j_1 < \dots < j_k < n$.

- *abstract simplex*
  - $(dgop \;\; gmsm) := \begin{cases} \text{dgop is a strictly decreasing sequence of degeneracy maps} \\ \text{gmsm is a geometric simplex} \end{cases}$
  - Examples

	simplex	abstract simplex
non-degenerate	$(a\ b)$	$(\emptyset\ (a\ b))$
degenerate	$(a\ a\ b\ c)$	$(\eta_0\ (a\ b\ c))$

Mathematical context

- *degeneracy operator*

$$\eta_i^q(dgop \quad gmsm) := (\eta_i^q \circ dgop \quad gmsm)$$

Mathematical context

- *degeneracy operator*

$$\eta_i^q(dgop \quad gmsm) := (\eta_i^q \circ dgop \quad gmsm)$$

- *face operator*

$$\partial_i^q(dgop \quad gmsm) := \begin{cases} (\partial_i^q \circ dgop \quad gmsm) & \text{if } \eta_i \in dgop \vee \eta_{i-1} \in dgop \\ (\partial_i^q \circ dgop \quad \partial_k^r gmsm) & \text{otherwise;} \end{cases}$$

where

$$r = q - \{\text{number of degeneracies in } dgop\} \text{ and}$$

$$k = i - \{\text{number of degeneracies in } dgop \text{ with index lower than } i\}$$

Mathematical context

- *degeneracy operator*

$$\eta_i^q(dgop \quad gmsm) := (\eta_i^q \circ dgop \quad gmsm)$$

- *face operator*

$$\partial_i^q(dgop \quad gmsm) := \begin{cases} (\partial_i^q \circ dgop \quad gmsm) & \text{if } \eta_i \in dgop \vee \eta_{i-1} \in dgop \\ (\partial_i^q \circ dgop \quad \partial_k^r gmsm) & \text{otherwise;} \end{cases}$$

where

$$r = q - \{\text{number of degeneracies in } dgop\} \text{ and}$$
$$k = i - \{\text{number of degeneracies in } dgop \text{ with index lower than } i\}$$

- *invariant operator*

$$(dgop \quad gmsm) \in K^q$$

  - (length $dgop$) $< q$
  - $gmsm \in K^r$ where $r = q -$ (length $dgop$)
  - index of the first degeneracy in $dgop$ is lower than q

Mathematical context

- *degeneracy operator*

$$\eta_i^q(dgop \quad gmsm) := (\eta_i^q \circ dgop \quad gmsm)$$

- *face operator*

$$\partial_i^q(dgop \quad gmsm) := \begin{cases} (\partial_i^q \circ dgop \quad gmsm) & \text{if } \eta_i \in dgop \vee \eta_{i-1} \in dgop \\ (\partial_i^q \circ dgop \quad \partial_k^r gmsm) & \text{otherwise;} \end{cases}$$

where

$r = q - \{$number of degeneracies in $dgop\}$ and

$k = i - \{$number of degeneracies in $dgop$ with index lower than $i\}$

- *invariant operator*

$$(dgop \quad gmsm) \in K^q$$

  - (length $dgop$) $< q$
  - $gmsm \in K^r$ where $r = q -$ (length $dgop$)
  - index of the first degeneracy in $dgop$ is lower than q

Dependent part from the chosen simplicial set → Affect geometric part

Independent parts from the chosen simplicial set → Not affect geometric part

Representation of a Simplicial Set

- Functions
    - face-absm
    - degeneracy-absm
    - invariant-absm

Representation of a Simplicial Set

- Functions
  - face-absm
  - degeneracy-absm
  - invariant-absm

Dependent parts

- face-gmsm
- invariant-gmsm

Representation of a Simplicial Set

- Functions
  - face-absm
  - degeneracy-absm
  - invariant-absm

Dependent parts

- face-gmsm
- invariant-gmsm

Independent parts

- degeneracy
- face-independent
- invariant-independent

Representation of a Simplicial Set

- Functions
  - face-absm
  - degeneracy-absm
  - invariant-absm
- Properties
  1. $\partial_i^{q-1}\partial_j^q = \partial_{j-1}^{q-1}\partial_i^q$ if $i < j$
  2. $\eta_i^{q+1}\eta_j^q = \eta_{j+1}^{q+1}\eta_i^q$ if $i \leq j$
  3. $\partial_i^{q+1}\eta_j^q = \eta_{j-1}^{q-1}\partial_i^q$ if $i < j$
  4. $\partial_i^{q+1}\eta_i^q = \textit{identity} = \partial_{i+1}^{q+1}\eta_i^q$
  5. $\partial_i^{q+1}\eta_j^q = \eta_j^{q-1}\partial_{i-1}^q$ if $i > j+1$
  6. $x \in K^q \Rightarrow \eta_i^q x \in K^{q+1}$
  7. $x \in K^q \Rightarrow \partial_i^q x \in K^{q-1}$

Dependent parts

- face-gmsm
- invariant-gmsm

Independent parts

- degeneracy
- face-independent
- invariant-independent

Representation of a Simplicial Set

- Functions
  - face-absm
  - degeneracy-absm
  - invariant-absm
- Properties
  1. $\partial_i^{q-1}\partial_j^q = \partial_{j-1}^{q-1}\partial_i^q$ if $i < j$
  2. $\eta_i^{q+1}\eta_j^q = \eta_{j+1}^{q+1}\eta_i^q$ if $i \leq j$
  3. $\partial_i^{q+1}\eta_j^q = \eta_{j-1}^{q-1}\partial_i^q$ if $i < j$
  4. $\partial_i^{q+1}\eta_i^q = \; identity \; = \partial_{i+1}^{q+1}\eta_i^q$
  5. $\partial_i^{q+1}\eta_j^q = \eta_j^{q-1}\partial_{i-1}^q$ if $i > j+1$
  6. $x \in K^q \Rightarrow \eta_i^q x \in K^{q+1}$
  7. $x \in K^q \Rightarrow \partial_i^q x \in K^{q-1}$

Dependent parts

- face-gmsm
- invariant-gmsm
- $\partial_i^{q-1}\partial_j^q gmsm = \partial_{j-1}^{q-1}\partial_i^q gmsm, i < j$
- $gmsm \in K^q \Rightarrow \partial_i^q gmsm \in K^{q-1}$

Independent parts

- degeneracy
- face-independent
- invariant-independent
- Independent parts of Properties (1) and (7)

Representation of a Simplicial Set

- Functions
  - face-absm
  - degeneracy-absm
  - invariant-absm
- Properties
  1. $\partial_i^{q-1}\partial_j^q = \partial_{j-1}^{q-1}\partial_i^q$ if $i < j$
  2. $\eta_i^{q+1}\eta_j^q = \eta_{j+1}^{q+1}\eta_i^q$ if $i \leq j$
  3. $\partial_i^{q+1}\eta_j^q = \eta_{j-1}^{q-1}\partial_i^q$ if $i < j$
  4. $\partial_i^{q+1}\eta_i^q = \ identity \ = \partial_{i+1}^{q+1}\eta_i^q$
  5. $\partial_i^{q+1}\eta_j^q = \eta_j^{q-1}\partial_{i-1}^q$ if $i > j+1$
  6. $x \in K^q \Rightarrow \eta_i^q x \in K^{q+1}$
  7. $x \in K^q \Rightarrow \partial_i^q x \in K^{q-1}$

Dependent parts

- face-gmsm
- invariant-gmsm
- $\partial_i^{q-1}\partial_j^q gmsm = \partial_{j-1}^{q-1}\partial_i^q gmsm, i < j$
- $gmsm \in K^q \Rightarrow \partial_i^q gmsm \in K^{q-1}$

Independent parts

- degeneracy
- face-independent
- invariant-independent
- Independent parts of Properties (1) and (7)
- Properties (2) to (6)

ACL2 representation of a Simplicial Set

- Functions
  - face-absm
  - degeneracy-absm
  - invariant-absm
- Properties
  1. $\partial_i^{q-1}\partial_j^q = \partial_{j-1}^{q-1}\partial_i^q$ if $i < j$
  2. $\eta_i^{q+1}\eta_j^q = \eta_{j+1}^{q+1}\eta_i^q$ if $i \leq j$
  3. $\partial_i^{q+1}\eta_j^q = \eta_{j-1}^{q-1}\partial_i^q$ if $i < j$
  4. $\partial_i^{q+1}\eta_i^q = \ identity \ = \partial_{i+1}^{q+1}\eta_i^q$
  5. $\partial_i^{q+1}\eta_j^q = \eta_j^{q-1}\partial_{i-1}^q$ if $i > j+1$
  6. $x \in K^q \Rightarrow \eta_i^q x \in K^{q+1}$
  7. $x \in K^q \Rightarrow \partial_i^q x \in K^{q-1}$

```
(encapsulate
 ; Signatures
 (((face-absm * * *)  => *)
  ((degeneracy-absm * * *)  => *)
  ((invariant-absm * *) => *))
 ; Theorems
 (defthm property1 ...)
 (defthm property2 ...)
 (defthm property3 ...)
 (defthm property4 ...)
 (defthm property5 ...)
 (defthm property6 ...)
 (defthm property7 ...))
```

ACL2 representation of a Simplicial Set

- Functions
  - face-absm
  - degeneracy-absm
  - invariant-absm
- Properties
  1. $\partial_i^{q-1}\partial_j^q = \partial_{j-1}^{q-1}\partial_i^q$ if $i < j$
  2. $\eta_i^{q+1}\eta_j^q = \eta_{j+1}^{q+1}\eta_i^q$ if $i \leq j$
  3. $\partial_i^{q+1}\eta_j^q = \eta_{j-1}^{q-1}\partial_i^q$ if $i < j$
  4. $\partial_i^{q+1}\eta_i^q = identity = \partial_{i+1}^{q+1}\eta_i^q$
  5. $\partial_i^{q+1}\eta_j^q = \eta_j^{q-1}\partial_{i-1}^q$ if $i > j+1$
  6. $x \in K^q \Rightarrow \eta_i^q x \in K^{q+1}$
  7. $x \in K^q \Rightarrow \partial_i^q x \in K^{q-1}$

```
(encapsulate
 ; Signatures
 (((face-absm * * *)  => *)
  ((degeneracy-absm * * *)  => *)
  ((invariant-absm * *) => *))
 ; Theorems
 (defthm property1 ...)
 (defthm property2 ...)
 (defthm property3 ...)
 (defthm property4 ...)
 (defthm property5 ...)
 (defthm property6 ...)
 (defthm property7 ...))
```

Concrete Simplicial Set

- 3 definitions + 7 theorems
- Proofs are not reusable for other cases

ACL2 representation of a Simplicial Set

Dependent parts

- face-gmsm
- invariant-gmsm
- $\partial_i^{q-1} \partial_j^q gmsm = \partial_{j-1}^{q-1} \partial_i^q gmsm, i < j$
- $gmsm \in K^q \Rightarrow \partial_i^q gmsm \in K^{q-1}$

Independent parts

- degeneracy
- face-independent
- invariant-independent
- Independent parts of Properties (1) and (7)
- Properties (2) to (6)

```
(encapsulate
 ; Signatures
 (((face-gmsm * * *)  => *)
  ((inv-gmsm * *) => *))
; Theorems
(defthm faceoface ;; ($\partial_i^{q-1}\partial_j^q gmsm = \partial_{j-1}^{q-1}\partial_i^q gmsm$ if $i < j$)
   ...)
(defthm invariant-prop ;; ($gmsm \in K^q \Rightarrow \partial_i^q gmsm \in K^{q-1}$)
   ...))
```

ACL2 representation of a Simplicial Set

Dependent parts

- face-gmsm
- invariant-gmsm
- $\partial_i^{q-1}\partial_j^q gmsm = \partial_{j-1}^{q-1}\partial_i^q gmsm, i < j$
- $gmsm \in K^q \Rightarrow \partial_i^q gmsm \in K^{q-1}$

Independent parts

- degeneracy
- face-independent
- invariant-independent
- Independent parts of Properties (1) and (7)
- Properties (2) to (6)

```
(encapsulate
 ; Signatures
 (((face-gmsm * * *)  => *)
  ((inv-gmsm * *) => *))
; Theorems
(defthm faceoface ;; ($\partial_i^{q-1}\partial_j^q gmsm = \partial_{j-1}^{q-1}\partial_i^q gmsm$ if $i < j$)
   ...)
(defthm invariant-prop ;; ($gmsm \in K^q \Rightarrow \partial_i^q gmsm \in K^{q-1}$)
   ...))
```

Definitions of independent parts

Proof of the independent theorems

ACL2 representation of a Simplicial Set

Dependent parts

- face-gmsm
- invariant-gmsm
- $\partial_i^{q-1}\partial_j^q gmsm = \partial_{j-1}^{q-1}\partial_i^q gmsm, i < j$
- $gmsm \in K^q \Rightarrow \partial_i^q gmsm \in K^{q-1}$

Independent parts

- degeneracy
- face-independent
- invariant-independent
- Independent parts of Properties (1) and (7)
- Properties (2) to (6)

`(encapsulate`

` ; Signatures`

` (((face-gmsm * * *)  => *)`

`  ((inv-gmsm * *) => *))`

`; Theorems`

`(defthm faceoface ;;` $(\partial_i^{q-1}\partial_j^q gmsm = \partial_{j-1}^{q-1}\partial_i^q gmsm$ if $i < j)$

`   ...)`

`(defthm invariant-prop ;;` $(gmsm \in K^q \Rightarrow \partial_i^q gmsm \in K^{q-1})$

`   ...))`

Definitions of independent parts

Proof of the independent theorems

Construction of a simplicial set instance

ACL2 representation of a Simplicial Set

Dependent parts

- face-gmsm
- invariant-gmsm
- $\partial_i^{q-1} \partial_j^q gmsm = \partial_{j-1}^{q-1} \partial_i^q gmsm, i < j$
- $gmsm \in K^q \Rightarrow \partial_i^q gmsm \in K^{q-1}$

Independent parts

- degeneracy
- face-independent
- invariant-independent
- Independent parts of Properties (1) and (7)
- Properties (2) to (6)

Concrete Simplicial Set

- 2 definitions + 2 theorems
- Proofs are reusable

```
(encapsulate
 ; Signatures
 (((face-gmsm * * *)  => *)
  ((inv-gmsm * *) => *))
; Theorems
(defthm faceoface ;; ($\partial_i^{q-1} \partial_j^q gmsm = \partial_{j-1}^{q-1} \partial_i^q gmsm$ if $i < j$)
   ...)
(defthm invariant-prop ;; ($gmsm \in K^q \Rightarrow \partial_i^q gmsm \in K^{q-1}$)
   ...))
```

Definitions of independent parts

Proof of the independent theorems

Construction of a simplicial set instance

ACL2 representation of a Simplicial Set

```
(encapsulate
 ; Signatures
 (((face-gmsm * * *)  => *)
  ((inv-gmsm * *) => *))
; Theorems
(defthm faceoface ;; $(\partial_i^{q-1}\partial_j^q gmsm = \partial_{j-1}^{q-1}\partial_i^q gmsm$ if $i < j)$
   ...)
(defthm invariant-prop ;; $(gmsm \in K^q \Rightarrow \partial_i^q gmsm \in K^{q-1})$
   ...))
```

Definitions of independent parts

Proof of the independent theorems

Construction of a simplicial set instance

Concrete Simplicial Set

- 2 definitions + 2 theorems
- Proofs are reusable

```
(encapsulate
 ; Signatures
 (((face-absm * * *)  => *)
  ((degeneracy-absm * * *)  => *)
  ((invariant-absm * *) => *))
 ; Theorems
 (defthm property1 ...)
 (defthm property2 ...)
 (defthm property3 ...)
 (defthm property4 ...)
 (defthm property5 ...)
 (defthm property6 ...)
 (defthm property7 ...))
```

Concrete Simplicial Set

- 3 definitions + 7 theorems
- Proofs are not reusable for other cases

Summary of our methodology

(reduced) encapsulate + independent functions

Summary of our methodology

(reduced) encapsulate + independent functions $\xrightarrow{proof}$ Generic Simplicial Set

Summary of our methodology

(reduced) encapsulate + independent functions $\xrightarrow{\textit{proof}}$ Generic Simplicial Set

↑

encapsulate instance

Summary of our methodology

$$\begin{tikzcd}
\text{(reduced) encapsulate} & + & \text{independent functions} \arrow[r, "\textit{proof}"] & \text{Generic Simplicial Set} \arrow[d] \\
& \text{encapsulate instance} \arrow[u] & & \text{Concrete Simplicial Set} \\
& & & + \\
& & & \text{Proof correctness}
\end{tikzcd}$$

Summary of our methodology

(reduced) encapsulate + independent functions $\xrightarrow{\textit{proof}}$ Generic Simplicial Set

↑ encapsulate instance

↓ Concrete Simplicial Set + Proof correctness

- From 2 definitions and 2 theorems
- Instantiates 3 definitions and 7 theorems
- The proof of the 7 theorems involves: 92 definitions and 969 theorems
- The proof effort is considerably reduced

Summary of our methodology

$$\begin{tikzcd}
\text{(reduced) encapsulate} & + & \text{independent functions} \arrow[r, "\textit{proof}"] & \text{Generic Simplicial Set} \arrow[d] \\
& \text{encapsulate instance} \arrow[u] & & \text{Concrete Simplicial Set} \\
& & & + \\
& & & \text{Proof correctness}
\end{tikzcd}$$

- From 2 definitions and 2 theorems
- Instantiates 3 definitions and 7 theorems
- The proof of the 7 theorems involves: 92 definitions and 969 theorems
- The proof effort is considerably reduced
- Generic Instantiation tool

  F. J. Martín-Mateos, J. A. Alonso, M. J. Hidalgo, and J. L. Ruiz-Reina. A Generic Instantiation Tool and a Case Study: A Generic Multiset Theory. Proceedings of the Third ACL2 workshop. Grenoble, Francia, pp. 188–203, 2002

Summary of our methodology

(reduced) encapsulate + independent functions $\xrightarrow{proof}$ Generic Simplicial Set

↑ encapsulate instance

↓ Concrete Simplicial Set + Proof correctness

- From 2 definitions and 2 theorems
- Instantiates 3 definitions and 7 theorems
- The proof of the 7 theorems involves: 92 definitions and 969 theorems
- The proof effort is considerably reduced
- Generic Instantiation tool
  - F. J. Martín-Mateos, J. A. Alonso, M. J. Hidalgo, and J. L. Ruiz-Reina. A Generic Instantiation Tool and a Case Study: A Generic Multiset Theory. Proceedings of the Third ACL2 workshop. Grenoble, Francia, pp. 188–203, 2002
- This methodology can be extrapolated to other cases

Generic Theory for families of Simplicial Sets

A simplicial set

```
(encapsulate
 ; Signatures
 (((face-absm * * *)  => *)
  ((degeneracy-absm * * *)  => *)
  ((invariant-absm * *) => *))
 ; Theorems
 (defthm property1 ...)
 (defthm property2 ...)
 (defthm property3 ...)
 (defthm property4 ...)
 (defthm property5 ...)
 (defthm property6 ...)
 (defthm property7 ...))
```

A family of simplicial sets indexed by $\mathcal{K}$

```
(encapsulate
 ; Signatures
 (((imp-face-absm *  * * *)  => *)
  ((imp-degeneracy-absm *  * * *)  => *)
  ((imp-invariant-absm *  * *) => *))
 ; Theorems
 (defthm imp-property1 ...)
 (defthm imp-property2 ...)
 (defthm imp-property3 ...)
 (defthm imp-property4 ...)
 (defthm imp-property5 ...)
 (defthm imp-property6 ...)
 (defthm imp-property7 ...))
```

Generic Theory for families of Simplicial Sets

A simplicial set

```
(encapsulate
 ; Signatures
 (((face-gmsm * * *)  => *)
  ((inv-gmsm * *) => *))
; Theorems
(defthm faceoface
;; ($\partial_i^{q-1}\partial_j^q gmsm = \partial_{j-1}^{q-1}\partial_i^q gmsm$ if $i < j$)
   ...)
(defthm invariant-prop
;; ($gmsm \in K^q \Rightarrow \partial_i^q gmsm \in K^{q-1}$)
   ...))
```

A family of simplicial sets indexed by $\mathcal{K}$

```
(encapsulate
 ; Signatures
 (((face-gmsm * * * *)  => *)
  ((inv-gmsm *  * *) => *)
  ((indexp *) => *))
; Theorems
(defthm imp-faceoface
;; ($\partial_i^{q-1}\partial_j^q gmsm = \partial_{j-1}^{q-1}\partial_i^q gmsm$ if $i < j$)
   ...)
(defthm imp-invariant-prop
;; ($gmsm \in K^q \Rightarrow \partial_i^q gmsm \in K^{q-1}$)
   ...))
```

Generic Theory for families of Simplicial Sets

A simplicial set

```
(encapsulate
 ; Signatures
 (((face-gmsm * * *)  => *)
  ((inv-gmsm * *) => *))
; Theorems
(defthm faceoface
```
;; $(\partial_i^{q-1}\partial_j^q gmsm = \partial_{j-1}^{q-1}\partial_i^q gmsm$ if $i < j)$
```
   ...)
(defthm invariant-prop
```
;; $(gmsm \in K^q \Rightarrow \partial_i^q gmsm \in K^{q-1})$
```
   ...))
```

A family of simplicial sets indexed by $\mathcal{K}$

```
(encapsulate
 ; Signatures
 (((face-gmsm * * * *)  => *)
  ((inv-gmsm *  * *) => *)
  ((indexp *) => *))
; Theorems
(defthm imp-faceoface
```
;; $(\partial_i^{q-1}\partial_j^q gmsm = \partial_{j-1}^{q-1}\partial_i^q gmsm$ if $i < j)$
```
   ...)
(defthm imp-invariant-prop
```
;; $(gmsm \in K^q \Rightarrow \partial_i^q gmsm \in K^{q-1})$
```
   ...))
```

Generic Simplicial Set Theory

- Spheres family
- Standard Simplicial sets family
- Simplicial Complexes family

Table of Contents

Conclusions

Conclusions

- Communication ($fKenzo$)
  - Integral assistant for Algebraic Topology
  - Provides a mediated access to Kenzo
  - Friendly front-end

Conclusions

Conclusions

- Communication (*fKenzo*)
  - Integral assistant for Algebraic Topology
  - Provides a mediated access to Kenzo
  - Friendly front-end
- Computation
  - Implementation of new Kenzo modules
  - Integration with other Computer Algebra systems

Conclusions

Conclusions

- Communication ($fKenzo$)
  - Integral assistant for Algebraic Topology
  - Provides a mediated access to Kenzo
  - Friendly front-end
- Computation
  - Implementation of new Kenzo modules
  - Integration with other Computer Algebra systems
- Deduction
  - Formalization of correctness of Kenzo programs using ACL2

Further work

Further work

- Formalization of Homological Algebra and Algebraic Topology libraries
    - ACL2
    - Coq/SSReflect

Further work

Further work

- Formalization of Homological Algebra and Algebraic Topology libraries
  - ACL2
  - Coq/SSReflect
- Integration of Theorem Prover tools
  - Compare developments in different theorem provers
  - ACL2 as Coq's oracle

Further work

Further work

- Formalization of Homological Algebra and Algebraic Topology libraries
    - ACL2
    - Coq/SSReflect
- Integration of Theorem Prover tools
    - Compare developments in different theorem provers
    - ACL2 as Coq's oracle
- Applications to biomedical images

Mathematical Knowledge Management in Algebraic Topology

Jónathan Heras Vicente

Supervisors: Dr. Vico Pascual Martínez-Losa
Dr. Julio Rubio García

Department of Mathematics and Computer Science
University of La Rioja
Spain

May 31, 2011