ACL2 verification of Simplicial Complexes programs for the Kenzo system[1]

Jónathan Heras, Vico Pascual and Julio Rubio

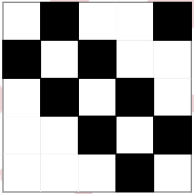
Departamento de Matemáticas y Computación
Universidad de La Rioja
Spain

February 10, 2011

[1] Partially supported by Ministerio de Educación y Ciencia, project MTM2009-13842-C02-01, and by European Commission FP7, STREP project ForMath

Algebraic Topology and Digital Images

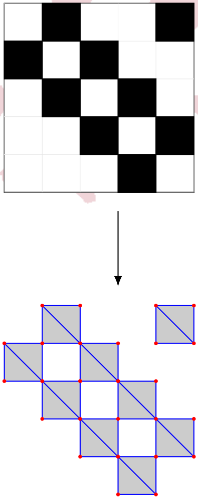
Digital Image



Simplicial set

Algebraic Topology and Digital Images

Digital Image

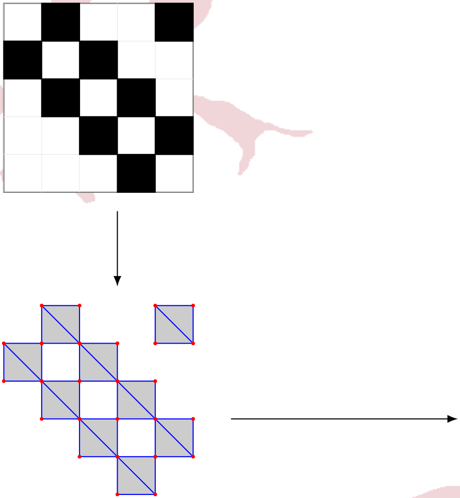


Simplicial complex

Simplicial set

Algebraic Topology and Digital Images

Digital Image



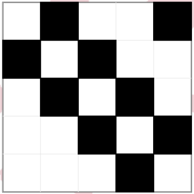
K_0 = vertices
K_1 = edges
K_2 = triangles

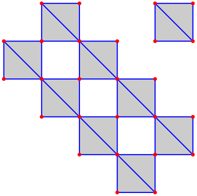
Simplicial complex

Simplicial set

Algebraic Topology and Digital Images

Digital Image





Simplicial complex

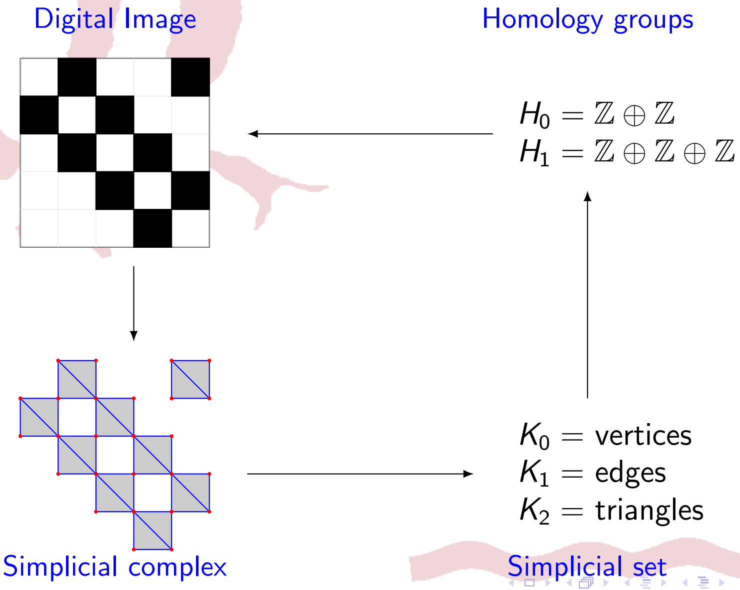
Homology groups

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

$$K_0 = \text{vertices}$$
$$K_1 = \text{edges}$$
$$K_2 = \text{triangles}$$

Simplicial set

Algebraic Topology and Digital Images



Goal

Digital Image

Goal

Digital Image

↓ triangulation

Simplicial Complex

Goal

Digital Image

↓ triangulation

Simplicial Complex

↓ graded structure

Simplicial Set

Goal

Digital Image

↓ triangulation

Simplicial Complex

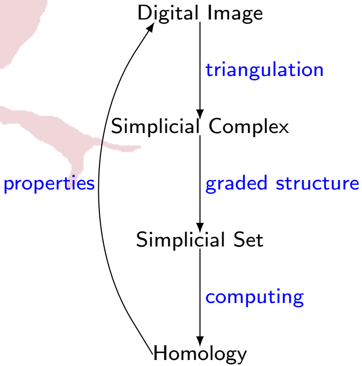
↓ graded structure

Simplicial Set

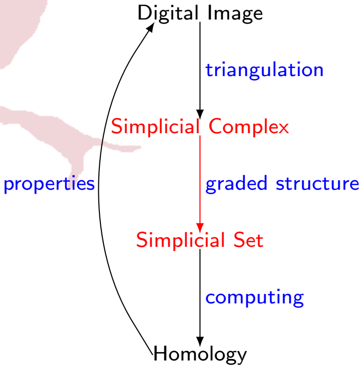
↓ computing

Homology

Goal


Digital Image
triangulation
Simplicial Complex
graded structure
Simplicial Set
computing
Homology
properties

Goal



- Goal:
  - A new *certified* program for Simplicial Complexes

Kenzo

- Kenzo
  - Symbolic Computation System devoted to Algebraic Topology

Kenzo

- Kenzo
  - Symbolic Computation System devoted to Algebraic Topology
  - Homology groups unreachable by any other means

Kenzo

- Kenzo
  - Symbolic Computation System devoted to Algebraic Topology
  - Homology groups unreachable by any other means
  - A Common Lisp package

Kenzo

- Kenzo
  - Symbolic Computation System devoted to Algebraic Topology
  - Homology groups unreachable by any other means
  - A Common Lisp package
  - Works with the main mathematical structures in Simplicial Algebraic Topology

Kenzo

- Kenzo
  - Symbolic Computation System devoted to Algebraic Topology
  - Homology groups unreachable by any other means
  - A Common Lisp package
  - Works with the main mathematical structures in Simplicial Algebraic Topology
- Increasing the reliability of Kenzo by means of Theorem Provers:
  - Isabelle
  - Coq
  - ACL2

Kenzo

- Kenzo
  - Symbolic Computation System devoted to Algebraic Topology
  - Homology groups unreachable by any other means
  - A Common Lisp package
  - Works with the main mathematical structures in Simplicial Algebraic Topology
- Increasing the reliability of Kenzo by means of Theorem Provers:
  - Isabelle
  - Coq
  - ACL2 - simplicial structures

ACL2

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

ACL2

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

ACL2

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

Goal

- Goal:
  - New Kenzo module for Simplicial Complexes certified in ACL2

Table of Contents

Table of Contents

Simplicial Complexes

Definition

Let V be an ordered set, called the vertex set.
A simplex over V is any finite subset of V.

Simplicial Complexes

Definition

Let V be an ordered set, called the vertex set.
A simplex over V is any finite subset of V.

Definition

Let α and β be simplexes over V, we say α is a face of β if α is a subset of β.

Simplicial Complexes

Definition

Let V be an ordered set, called the vertex set.
A simplex over V is any finite subset of V.

Definition

Let α and β be simplexes over V, we say α is a face of β if α is a subset of β.

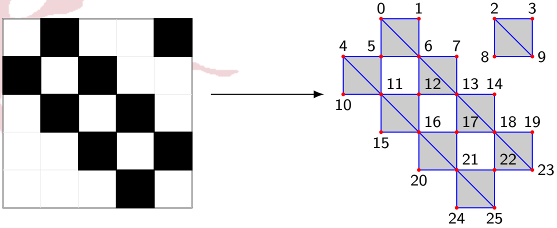
Definition

An ordered (abstract) simplicial complex over V is a set of simplexes $\mathcal{K}$ over V satisfying the property:

$$\forall \alpha \in \mathcal{K}, \text{ if } \beta \subseteq \alpha \Rightarrow \beta \in \mathcal{K}$$

Let $\mathcal{K}$ be a simplicial complex. Then the set $S_n(\mathcal{K})$ of n-simplexes of $\mathcal{K}$ is the set made of the simplexes of cardinality $n+1$.

Simplicial Complexes

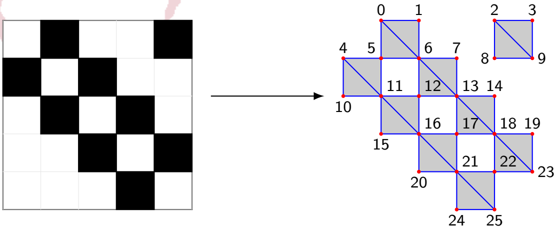


$V = (0, 1, 2, \ldots, 24, 25)$
$\mathcal{K} =$ vertices $\cup$ edges $\cup$ triangles

Simplicial Complexes

Definition

The facets of a simplicial complex $\mathcal{K}$ are the maximal simplexes of the simplicial complex.



The facets are the triangles

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:

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

From Simplicial Complexes to Simplicial Sets

Simplicial Complex $\xrightarrow{\text{graded structure}}$ Simplicial Set

Definition

Let C be a simplicial complex. Then the *simplicial set* $K(C)$ *canonically associated* with C is defined as follows. The set $K^n(C)$ of n-simplexes is the set made of the simplexes of cardinality $n+1$ of C. In addition, let a simplex $\{v_0, \ldots, v_q\}$ the *face* and *degeneracy* operators are defined as follows:

$$\begin{aligned} \partial_i(\{v_0, \ldots, v_i, \ldots, v_q\}) &= \{v_0, \ldots, v_{i-1}, v_{i+1}, \ldots, v_q\} \\ \eta_i(\{v_0, \ldots, v_i, \ldots, v_q\}) &= \{v_0, \ldots, v_i, v_i, \ldots, v_q\} \end{aligned}$$

Goals

- New Kenzo module:

Goals

- New Kenzo module:
  - program1: facets $\longrightarrow$ simplicial complex
  - program2: simplicial complex $\longrightarrow$ simplicial set

Goals

- New Kenzo module:
  - program1: facets $\longrightarrow$ simplicial complex
  - program2: simplicial complex $\longrightarrow$ simplicial set
- Certification of the correctness of the programs in ACL2

Table of Contents

Program1: Simplicial Complex from facets

- `simplicial-complex-generator`:
  Input: a list of simplexes
  Output: a simplicial complex

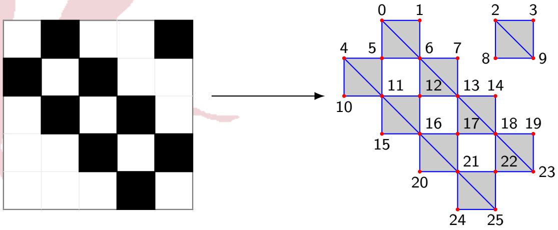
Program1: Simplicial Complex from facets

- `simplicial-complex-generator`:
  Input: a list of simplexes
  Output: a simplicial complex
  - `simplicial-complex-generator-with-duplicates`:
    Input: a list of simplexes
    Output: a list of simplexes with the properties of simplicial complexes but with duplicates

Program1: Simplicial Complex from facets

- `simplicial-complex-generator`:
  Input: a list of simplexes
  Output: a simplicial complex
  - `simplicial-complex-generator-with-duplicates`:
    Input: a list of simplexes
    Output: a list of simplexes with the properties of simplicial complexes but with duplicates
  - `simplicial-complex-generator-from-simplex`:
    Input: a simplex
    Output: a simplicial complex

Example




```
> (setf image-sc (simplicial-complex-generator
'((0 1 6) (0 5 6) (2 3 9) (2 8 9) (4 5 11) (4 10 11)
  (6 7 13) (6 12 13) (11 12 16) (11 15 16) (13 14 18) (13 17 18)
  (16 17 21) (16 20 21) (18 19 23) (18 22 23) (21 22 25) (21 24 25))) ✠
((0 1 6) (0 1) (0 6) (1 6) (0) (1) (6) (0 5 6) (0 5) (5 6) ...)
```

Program2: Simplicial Set from Simplicial Complex

- `ss-from-sc`:
  Input: a simplicial complex
  Output: a simplicial set

Program2: Simplicial Set from Simplicial Complex

- `ss-from-sc`:
  Input: a simplicial complex
  Output: a simplicial set
  - Kenzo function `build-smst`:

```
(build-smst
  :basis basis
  :face face
  ...)
```

    - `basis`: a function returning the list of simplexes in a dimension
    - `face`: a function for face operation
    - `degeneracy`: not included

Example

Simplicial set canonically associated to `image-sc`:

```
> (setf image-ss (ss-from-sc image-sc)) ✠
[K1 Simplicial-Set]
```

Example

Simplicial set canonically associated to `image-sc`:

```
> (setf image-ss (ss-from-sc image-sc)) ✠
[K1 Simplicial-Set]
```

```
> (basis image-ss 0) ✠
((0) (1) (2) (3) (4) (5) (6) (7) (8) (9) ...)
```

Example

Simplicial set canonically associated to `image-sc`:

```
> (setf image-ss (ss-from-sc image-sc)) ✠
[K1 Simplicial-Set]
```

```
> (basis image-ss 0) ✠
((0) (1) (2) (3) (4) (5) (6) (7) (8) (9) ...)
```

```
> (homology image-ss 0 2) ✠
Homology in dimension 0:
Component Z
Component Z
Homology in dimension 1:
Component Z
Component Z
Component Z
```

$H_0(image) = \mathbb{Z} \oplus \mathbb{Z}$ and $H_1(image) = \mathbb{Z} \oplus \mathbb{Z} \oplus \mathbb{Z}$

Table of Contents

Program1

`simplicial-complex-generator` program:

Program1

`simplicial-complex-generator` program:

- Follows simple inductive schemas

Program1

`simplicial-complex-generator` program:

- Follows simple inductive schemas
- Inefficient

Input of a list of 11613 simplexes:

```
> (simplicial-complex-generator ...) ✠
Error: Stack overflow (signal 1000)
[condition type: SYNCHRONOUS-OPERATING-SYSTEM-SIGNAL]
```

Program1

`simplicial-complex-generator` program:

- Follows simple inductive schemas
- Inefficient

Input of a list of 11613 simplexes:

```
> (simplicial-complex-generator ...) ✠
Error: Stack overflow (signal 1000)
[condition type: SYNCHRONOUS-OPERATING-SYSTEM-SIGNAL]
```

`optimized-simplicial-complex-generator`:

Program1

`simplicial-complex-generator` program:

- Follows simple inductive schemas
- Inefficient

Input of a list of 11613 simplexes:

```
> (simplicial-complex-generator ...) ✠
Error: Stack overflow (signal 1000)
[condition type: SYNCHRONOUS-OPERATING-SYSTEM-SIGNAL]
```

`optimized-simplicial-complex-generator`:

- Equivalent efficient program

Program1

`simplicial-complex-generator` program:

- Follows simple inductive schemas
- Inefficient

Input of a list of 11613 simplexes:

```
> (simplicial-complex-generator ...) ✠
Error: Stack overflow (signal 1000)
[condition type: SYNCHRONOUS-OPERATING-SYSTEM-SIGNAL]
```

`optimized-simplicial-complex-generator`:

- Equivalent efficient program
- Memoization technique

Two equivalent algorithms for program1

Situation:

- `simplicial-complex-generator` program is
- `optimized-simplicial-complex-generator` program is

Two equivalent algorithms for program1

Situation:

- `simplicial-complex-generator` program is
  - specially designed to be proved;
- `optimized-simplicial-complex-generator` program is
  - designed to be efficient;

Two equivalent algorithms for program1

Situation:

- `simplicial-complex-generator` program is
  - specially designed to be proved;
  - programmed in ACL2 (and, of course, Common Lisp);
- `optimized-simplicial-complex-generator` program is
  - designed to be efficient;
  - written in Common Lisp;

Two equivalent algorithms for program1

Situation:

- `simplicial-complex-generator` program is
  - specially designed to be proved;
  - programmed in ACL2 (and, of course, Common Lisp);
  - not efficient;

- `optimized-simplicial-complex-generator` program is
  - designed to be efficient;
  - written in Common Lisp;
  - efficient;

Two equivalent algorithms for program1

Situation:

- `simplicial-complex-generator` program is
  - specially designed to be proved;
  - programmed in ACL2 (and, of course, Common Lisp);
  - not efficient;
  - tested;
- `optimized-simplicial-complex-generator` program is
  - designed to be efficient;
  - written in Common Lisp;
  - efficient;
  - tested;

Two equivalent algorithms for program1

Situation:

- `simplicial-complex-generator` program is
  - specially designed to be proved;
  - programmed in ACL2 (and, of course, Common Lisp);
  - not efficient;
  - tested;
  - proved in ACL2
- `optimized-simplicial-complex-generator` program is
  - designed to be efficient;
  - written in Common Lisp;
  - efficient;
  - tested;
  - unproved

Two equivalent algorithms for program1

- `optimized-simplicial-complex-generator` "equivalent to" `simplicial-complex-generator`

Two equivalent algorithms for program1

- `optimized-simplicial-complex-generator` "equivalent to" `simplicial-complex-generator`
- Not a proof of the equivalence

Two equivalent algorithms for program1

- `optimized-simplicial-complex-generator` "equivalent to" `simplicial-complex-generator`
- Not a proof of the equivalence
- Automated testing

```
(defun automated-testing ()
  (let ((cases (generate-test-cases 100000)))
    (dolist (case cases)
      (if (not (equal-as-sc (simplicial-complex-generator case)
                            (optimized-simplicial-complex-generator case)))
          (report-on-failure case)))
  )
```

A Common Lisp (but not ACL2) program

ACL2 definitions for Simplicial Complexes

Definition

An ordered (abstract) simplicial complex over V is a set of simplexes $\mathcal{K}$ over V satisfying the property:

$$\forall \alpha \in \mathcal{K},\ \textit{if}\ \beta \subseteq \alpha \Rightarrow \beta \in \mathcal{K}$$

ACL2 definitions for Simplicial Complexes

Definition

An ordered (abstract) simplicial complex over V is a set of simplexes $\mathcal{K}$ over V satisfying the property:

$$\forall \alpha \in \mathcal{K}, \textit{ if } \beta \subseteq \alpha \Rightarrow \beta \in \mathcal{K}$$

- simplex: `simplex-p`

ACL2 definitions for Simplicial Complexes

Definition

An ordered (abstract) simplicial complex over V is a set of simplexes $\mathcal{K}$ over V satisfying the property:

$$\forall \alpha \in \mathcal{K}, \text{ if } \beta \subseteq \alpha \Rightarrow \beta \in \mathcal{K}$$

- simplex: `simplex-p`
- list of simplexes: `list-of-simplexes-p`
- without duplicates: `without-duplicates-p`

ACL2 definitions for Simplicial Complexes

Definition

An ordered (abstract) simplicial complex over V is a set of simplexes $\mathcal{K}$ over V satisfying the property:

$$\forall \alpha \in \mathcal{K}, \text{ if } \beta \subseteq \alpha \Rightarrow \beta \in \mathcal{K}$$

- simplex: `simplex-p`
- list of simplexes: `list-of-simplexes-p`
- without duplicates: `without-duplicates-p`
- face: `subsetp-equal` (ACL2)

ACL2 definitions for Simplicial Complexes

Definition

An ordered (abstract) simplicial complex over V is a set of simplexes $\mathcal{K}$ over V satisfying the property:

$$\forall \alpha \in \mathcal{K}, \text{ if } \beta \subseteq \alpha \Rightarrow \beta \in \mathcal{K}$$

- simplex: `simplex-p`
- list of simplexes: `list-of-simplexes-p`
- without duplicates: `without-duplicates-p`
- face: `subsetp-equal` (ACL2)
- member: `member-equal` (ACL2)

ACL2 theorems for Simplicial Complexes

Definition

An ordered (abstract) simplicial complex over V is a set of simplexes $\mathcal{K}$ over V satisfying the property:

$$\forall \alpha \in \mathcal{K},\ \textit{if}\ \beta \subseteq \alpha \Rightarrow \beta \in \mathcal{K}$$

ACL2 Lemma

Let ls be a list of simplexes, then (`simplicial-complex-generator` ls) builds a set of simplexes.

```
(defun set-of-simplexes-p (ls)
  (and (list-of-simplexes-p ls) (without-duplicates-p ls)))
```

```
(defthm simplicial-complex-generator-constructs-simplicial-complex-1
  (implies (list-of-simplexes-p ls)
           (set-of-simplexes-p (simplicial-complex-generator ls))))
```

ACL2 theorems for Simplicial Complexes

Definition

An ordered (abstract) simplicial complex over V is a set of simplexes $\mathcal{K}$ over V satisfying the property:

$$\forall \alpha \in \mathcal{K}, \text{ if } \beta \subseteq \alpha \Rightarrow \beta \in \mathcal{K}$$

ACL2 Lemma

Let x be a simplex and ls be a list of simplexes, if x is in (simplicial-complex-generator ls) and y is a face of x, then y is in (simplicial-complex-generator ls).

```
(defthm simplicial-complex-generator-constructs-simplicial-complex-2
  (implies (and (simplex-p s1)
                (simplex-p s2)
                (list-of-simplexes-p ls)
                (member-equal s1 (simplicial-complex-generator ls))
                (subsetp-equal s2 s1))
           (member-equal s2 (simplicial-complex-generator ls))))
```

ACL2 theorems for Simplicial Complexes

Definition

An ordered (abstract) simplicial complex over V is a set of simplexes $\mathcal{K}$ over V satisfying the property:

$$\forall \alpha \in \mathcal{K}, \text{ if } \beta \subseteq \alpha \Rightarrow \beta \in \mathcal{K}$$

ACL2 Lemma

Let `ls` be a list of simplexes and let `s` be an element of the simplicial complex constructed with the `simplicial-complex-generator` function taking as argument `ls`; then, `s` is a face of some of the simplexes of `ls`.

```
(defthm simplicial-complex-generator-correctness
  (implies (and (list-of-simplexes-p ls)
                (member-equal s (simplicial-complex-generator ls))
           (face-of-some-p s ls)))
```

ACL2 theorems for Simplicial Complexes

ACL2 Theorem

Let ls be a list of simplexes, then (`simplicial-complex-generator ls`) constructs the simplicial complex associated with ls.

Proof

Apply the three previous lemmas

Theorem for Simplicial Sets from Simplicial Complexes

Proving truthfulness of Kenzo statements like:

```
> (setf image-ss (ss-from-sc image-sc)) ✠
[K1 Simplicial-Set]
```

where `image-sc` is a simplicial complex

Theorem for Simplicial Sets from Simplicial Complexes

Proving truthfulness of Kenzo statements like:

```
> (setf image-ss (ss-from-sc image-sc)) ✠
[K1 Simplicial-Set]
```

where `image-sc` is a simplicial complex

ACL2 Theorem

Let sc be a simplicial complex, then (`ss-from-sc` sc) constructs a simplicial set.

Main Tools

ACL2 Theorem

Let $\mathcal{K}$ be a Kenzo object implementing a simplicial set. If for every natural number $q \geq 2$ and for every geometric simplex gmsm in dimension q the following properties hold:

1. $\forall i, j \in \mathbb{N} : i < j \leq q \rightarrow \partial_i^{q-1} \circ (\partial_j^q gmsm) = \partial_{j-1}^{q-1} \circ (\partial_i^q gmsm)$,
2. $\forall i \in \mathbb{N}$, $i \leq q$: $\partial_i^q gmsm$ *is a simplex of* $\mathcal{K}$ *in dimension* $q-1$,

then:

$$\mathcal{K} \textit{ is a simplicial set.}$$

J. Heras, V. Pascual and J. Rubio, *Proving with ACL2 the correctness of simplicial sets in the Kenzo system*. In LOPSTR 2010, Lecture Notes in Computer Science. Springer-Verlag.

Main Tools

ACL2 Theorem

Let $\mathcal{K}$ be a Kenzo object implementing a simplicial set. If for every natural number $q \geq 2$ and for every geometric simplex gmsm in dimension q the following properties hold:

1. $\forall i, j \in \mathbb{N} : i < j \leq q \rightarrow \partial_i^{q-1} \circ (\partial_j^q gmsm) = \partial_{j-1}^{q-1} \circ (\partial_i^q gmsm)$,
2. $\forall i \in \mathbb{N}$, $i \leq q$: $\partial_i^q gmsm$ *is a simplex of* $\mathcal{K}$ *in dimension* $q-1$,

then:

$$\mathcal{K} \text{ is a simplicial set.}$$

J. Heras, V. Pascual and J. Rubio, *Proving with ACL2 the correctness of simplicial sets in the Kenzo system*. In LOPSTR 2010, Lecture Notes in Computer Science. Springer-Verlag.

- Generic instantiation tool:
  - Development of a generic theory
  - Instantiation of definitions and theorems for different implementations

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.

Generic simplicial set theory

- Generic simplicial set theory for simplicial complexes:
  - From 4 definitions and 4 theorems

Generic simplicial set theory

- Generic simplicial set theory for simplicial complexes:
  - From 4 definitions and 4 theorems
  - Instantiates 3 definitions and 7 theorems (+ 89 definitions and 969 theorems)

Generic simplicial set theory

- Generic simplicial set theory for simplicial complexes:
  - From 4 definitions and 4 theorems
  - Instantiates 3 definitions and 7 theorems (+ 89 definitions and 969 theorems)

ACL2 Theorem

Let sc be a simplicial complex, then (`ss-from-sc` sc) constructs a simplicial set.

Table of Contents

Conclusions and Further Work

Digital Image → Simplicial Complex → Simplicial Sets → Homology

- Conclusions:

Conclusions and Further Work

Digital Image → Simplicial Complex → Simplicial Sets → Homology

- Conclusions:
  - New module for the Kenzo system

Conclusions and Further Work

Digital Image → Simplicial Complex → Simplicial Sets → Homology

- Conclusions:
  - New module for the Kenzo system
  - Certification of the correctness of the new programs

Conclusions and Further Work

Digital Image → Simplicial Complex → Simplicial Sets → Homology

- Conclusions:
  - New module for the Kenzo system
  - Certification of the correctness of the new programs
- Further Work:

Conclusions and Further Work

Digital Image → Simplicial Complex → Simplicial Sets → Homology

- Conclusions:
  - New module for the Kenzo system
  - Certification of the correctness of the new programs
- Further Work:
  - Efficient algorithm in the ACL2 system

Conclusions and Further Work

Digital Image → Simplicial Complex → Simplicial Sets → Homology

- Conclusions:
  - New module for the Kenzo system
  - Certification of the correctness of the new programs
- Further Work:
  - Efficient algorithm in the ACL2 system
  - Equivalence between the new algorithm and the previous one

Conclusions and Further Work

Digital Image → Simplicial Complex → Simplicial Sets → Homology

- Conclusions:
  - New module for the Kenzo system
  - Certification of the correctness of the new programs
- Further Work:
  - Efficient algorithm in the ACL2 system
  - Equivalence between the new algorithm and the previous one
  - Digital Images → Simplicial Complexes

Conclusions and Further Work

Digital Image ⟶ Simplicial Complex ⟶ Simplicial Sets ⟶ Homology

- Conclusions:
  - New module for the Kenzo system
  - Certification of the correctness of the new programs
- Further Work:
  - Efficient algorithm in the ACL2 system
  - Equivalence between the new algorithm and the previous one
  - Digital Images $\longrightarrow$ Simplicial Complexes
  - Simplicial Sets $\longrightarrow$ Homology

ACL2 verification of Simplicial Complexes programs for the Kenzo system

Jónathan Heras, Vico Pascual and Julio Rubio

Departamento de Matemáticas y Computación
Universidad de La Rioja
Spain

February 10, 2011