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

Jónathan Heras and Vico Pascual

Departamento de Matemáticas y Computación
Universidad de La Rioja
Spain

April 21, 2010

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

Table of Contents

Table of Contents

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

- A new *certified* module for the Kenzo system
  - Simplicial Complexes

From "General" Topology to Homological Algebra

Topological Space

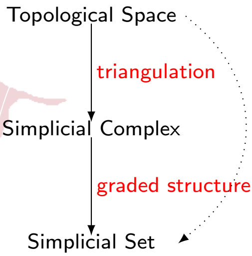
From "General" Topology to Homological Algebra

Topological Space

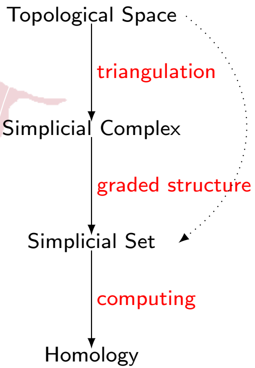
↓ triangulation

Simplicial Complex

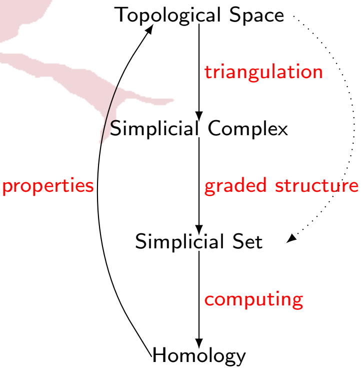
From "General" Topology to Homological Algebra



From “General” Topology to Homological Algebra

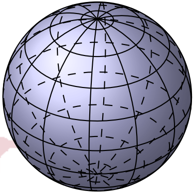


From "General" Topology to Homological Algebra

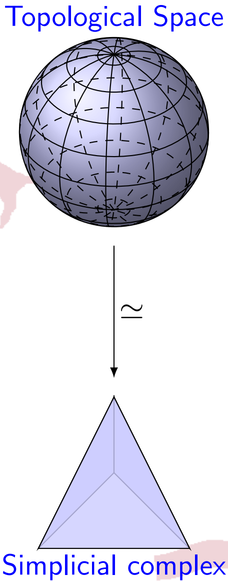


An example

Topological Space

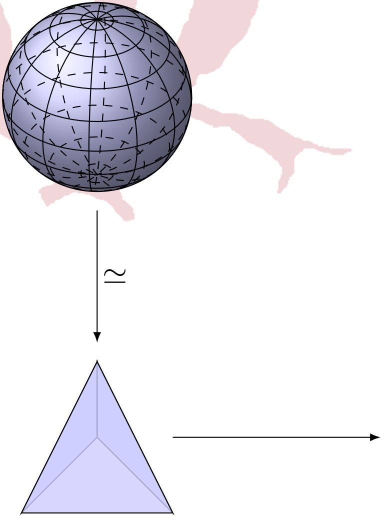


An example



An example

Topological Space



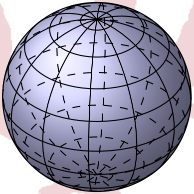
Simplicial complex

$$K_0 = \text{vertices (4 vertices)}$$
$$K_1 = \text{edges (6 edges)}$$
$$K_2 = \text{triangles (4 triangles)}$$

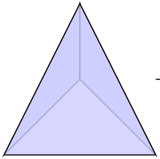
Simplicial set

An example

Topological Space



$\downarrow \simeq$



Simplicial complex

$\longrightarrow$

K_0 = vertices (4 vertices)
K_1 = edges (6 edges)
K_2 = triangles (4 triangles)

Simplicial set

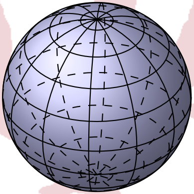
$\uparrow$

Homology groups

$$H_0 = \mathbb{Z}$$
$$H_1 = 0$$
$$H_2 = \mathbb{Z}$$
$$H_3 = 0$$
$$\ldots$$

An example

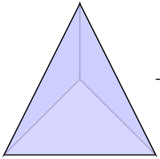
Topological Space



Homology groups

$$H_0 = \mathbb{Z}$$
$$H_1 = 0$$
$$H_2 = \mathbb{Z}$$
$$H_3 = 0$$
$$\dots$$

$\simeq$



Simplicial complex

K_0 = vertices (4 vertices)
K_1 = edges (6 edges)
K_2 = triangles (4 triangles)

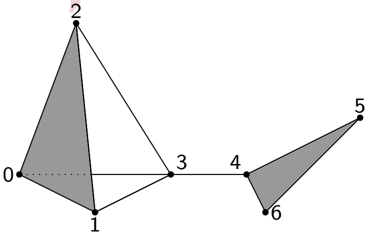
Simplicial set

Simplicial Complexes

Definition

A *simplicial complex* C is a finite set of simplexes satisfying the properties:

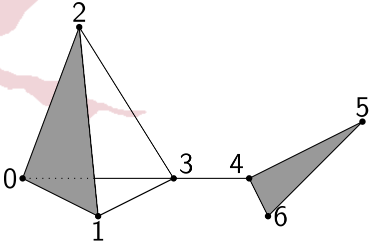
- if σ_n is a simplex of C, and τ_p is a face of σ_n, then τ_p is in C;
- if σ_n and τ_p are simplexes of C, then $\sigma_n \cap \tau_p$ is a common face of σ_n and τ_p.



$C = \{\emptyset, \{0\}, \{1\}, \{2\}, \{3\}, \{4\}, \{5\}, \{6\},$
$\{0,1\}, \{0,2\}, \{0,3\}, \{1,2\}, \{1,3\}, \{2,3\}, \{3,4\}, \{4,5\}, \{4,6\}, \{5,6\},$
$\{0,1,2\}, \{4,5,6\}\}$

Simplicial Complexes

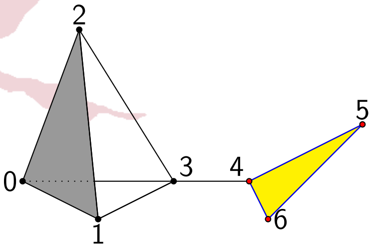
Facets: maximal elements of the simplicial complex



facets of this simplicial complex are: $\{\{1,3\},\{3,4\},\{0,3\},\{2,3\},\{0,1,2\},\{4,5,6\}\}$

Simplicial Complexes

Facets: maximal elements of the simplicial complex

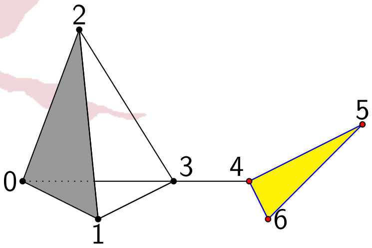


facets of this simplicial complex are: $\{\{1,3\},\{3,4\},\{0,3\},\{2,3\},\{0,1,2\},\{4,5,6\}\}$

$\{4,5,6\} \rightarrow \{\emptyset,\{4\},\{5\},\{6\},\{4,5\},\{5,6\},\{4,6\},\{4,5,6\}\}$

Simplicial Complexes

Facets: maximal elements of the simplicial complex



$C = \{\emptyset, \{0\}, \{1\}, \{2\}, \{3\}, \{4\}, \{5\}, \{6\},$
$\{0,1\}, \{0,2\}, \{0,3\}, \{1,2\}, \{1,3\}, \{2,3\}, \{3,4\}, \{4,5\}, \{4,6\}, \{5,6\},$
$\{0,1,2\}, \{4,5,6\}\}$

Simplicial Complexes to Simplicial Sets

Simplicial Complex —graded structure→ Simplicial Set

Simplicial Complexes to 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:

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

Simplicial Complexes to Simplicial Sets

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

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

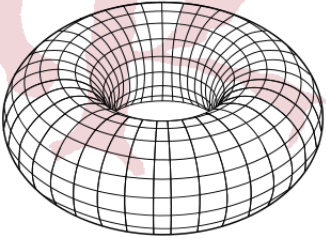
- `simplicial-complex-generator`:
  Input: a list of simplexes
  Output: a simplicial complex
  - `simplicial-complex-generator-with-duplicates`:
    Input: a list of simplexes
    Output: a list of simplexes with the properties of simplicial complexes but with duplicates

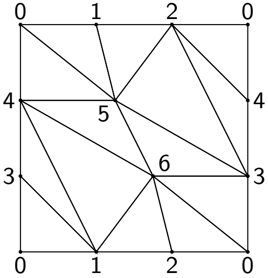
Program1: Simplicial Complex from facets

- `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

Torus simplicial complex $S^1 \times S^1$:






```
> (setf torus-sc (simplicial-complex-generator
'((0 1 3) (0 1 5) (0 2 4)
  (0 2 6) (0 3 6) (0 4 5)
  (1 2 5) (1 2 6) (1 3 4)
  (1 4 6) (2 3 4) (2 3 5)
  (3 5 6) (4 5 6))))
((0 1 3) (0 1) (0 1 5) (0 2 4) (0 2) ...)
```

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 `torus-sc`:

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

Example

Simplicial set canonically associated to `torus-sc`:

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

```
> (basis torus-ss 1)
((0 1) (0 2) (0 6) (0 3) (0 5) (0 4) (1 5) (2 6) (1 2) (1 3) ...)
```

Example

Simplicial set canonically associated to `torus-sc`:

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

```
> (basis torus-ss 1) ✠
((0 1) (0 2) (0 6) (0 3) (0 5) (0 4) (1 5) (2 6) (1 2) (1 3) ...)
```

```
> (homology torus-ss 0 3) ✠
Homology in dimension 0:
Component Z
Homology in dimension 1:
Component Z
Component Z
Homology in dimension 2:
Component Z
```

$H_0(torus) = \mathbb{Z}$, $H_1(torus) = \mathbb{Z} \oplus \mathbb{Z}$ and $H_2(torus) = \mathbb{Z}$

Table of Contents

Problem with program1

`simplicial-complex-generator` program:

Problem with program1

`simplicial-complex-generator` program:

- Follows simple inductive schemas

Problem with 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]
```

Problem with 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`:

Problem with 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

Problem with 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 (Simplicial Complex)

A *simplicial complex* C is a finite set of simplexes satisfying the properties:

- if σ_n is a simplex of C, and τ_p is a face of σ_n, then τ_p is in C;
- if σ_n and τ_p are simplexes of C, then $\sigma_n \cap \tau_p$ is a common face of σ_n and τ_p.

ACL2 definitions for Simplicial Complexes

Definition (Simplicial Complex)

A *simplicial complex* C is a finite set of simplexes satisfying the properties:

- if σ_n is a simplex of C, and τ_p is a face of σ_n, then τ_p is in C;
- if σ_n and τ_p are simplexes of C, then $\sigma_n \cap \tau_p$ is a common face of σ_n and τ_p.

- simplex: `simplex-p`

ACL2 definitions for Simplicial Complexes

Definition (Simplicial Complex)

A *simplicial complex* C is a finite set of simplexes satisfying the properties:

- if σ_n is a simplex of C, and τ_p is a face of σ_n, then τ_p is in C;
- if σ_n and τ_p are simplexes of C, then $\sigma_n \cap \tau_p$ is a common face of σ_n and τ_p.

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

ACL2 definitions for Simplicial Complexes

Definition (Simplicial Complex)

A *simplicial complex* C is a finite set of simplexes satisfying the properties:

- if σ_n is a simplex of C, and τ_p is a face of σ_n, then τ_p is in C;
- if σ_n and τ_p are simplexes of C, then $\sigma_n \cap \tau_p$ is a common face of σ_n and τ_p.

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

ACL2 definitions for Simplicial Complexes

Definition (Simplicial Complex)

A *simplicial complex* C is a finite set of simplexes satisfying the properties:

- if σ_n is a simplex of C, and τ_p is a face of σ_n, then τ_p is in C;
- if σ_n and τ_p are simplexes of C, then $\sigma_n \cap \tau_p$ is a common face of σ_n and τ_p.

- simplex: `simplex-p`
- list of simplexes: `list-of-simplexes-p`
- without duplicates: `without-duplicates-p`
- face: `subset-p`
- member: `member-equal`

ACL2 definitions for Simplicial Complexes

Definition (Simplicial Complex)

A *simplicial complex* C is a finite set of simplexes satisfying the properties:

- if σ_n is a simplex of C, and τ_p is a face of σ_n, then τ_p is in C;
- if σ_n and τ_p are simplexes of C, then $\sigma_n \cap \tau_p$ is a common face of σ_n and τ_p.

- simplex: `simplex-p`
- list of simplexes: `list-of-simplexes-p`
- without duplicates: `without-duplicates-p`
- face: `subset-p`
- member: `member-equal`
- intersection: `intersect`

ACL2 theorems for Simplicial Complexes

Definition (Simplicial Complex)

A *simplicial complex* C is a finite set of simplexes satisfying the properties:

- if σ_n is a simplex of C, and τ_p is a face of σ_n, then τ_p is in C;
- if σ_n and τ_p are simplexes of C, then $\sigma_n \cap \tau_p$ is a common face of σ_n and τ_p.

Lemma

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

```
(defthm simplicial-complex-generator-theorem-1
  (implies (list-of-simplexes-p ls)
           (and (list-of-simplexes-p (simplicial-complex-generator ls))
                (without-duplicates-p (simplicial-complex-generator ls)))))
```

ACL2 theorems for Simplicial Complexes

Definition (Simplicial Complex)

A *simplicial complex* C is a finite set of simplexes satisfying the properties:

- if σ_n is a simplex of C, and τ_p is a face of σ_n, then τ_p is in C;
- if σ_n and τ_p are simplexes of C, then $\sigma_n \cap \tau_p$ is a common face of σ_n and τ_p.

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-theorem-2
  (implies (and (simplex-p s1)
                (simplex-p s3)
                (list-of-simplexes-p ls)
                (member-equal s1 (simplicial-complex-generator ls))
                (subset-p s3 s1))
           (member-equal s3 (simplicial-complex-generator ls))))
```

ACL2 theorems for Simplicial Complexes

Definition (Simplicial Complex)

A *simplicial complex* C is a finite set of simplexes satisfying the properties:

- if σ_n is a simplex of C, and τ_p is a face of σ_n, then τ_p is in C;
- if σ_n and τ_p are simplexes of C, then $\sigma_n \cap \tau_p$ is a common face of σ_n and τ_p.

Lemma

Let x, y be simplexes and ls be a list of simplexes, if x and y are in ($\mathit{simplicial\text{-}complex\text{-}generator}$ ls), then $x \cap y$ is a common face of x and y.

```
(defthm simplicial-complex-generator-theorem-3
  (implies (and (list-of-simplexes-p ls)
                (member-equal s1 (simplicial-complex-generator ls))
                (member-equal s2 (simplicial-complex-generator ls)))
           (and (subset-p (intersect s1 s2) s1)
                (subset-p (intersect s1 s2) s2))))
```

ACL2 theorems for Simplicial Complexes

Theorem

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

Proof.

Apply the three previous lemmas □

Theorem for Simplicial Sets from Simplicial Complexes

Proving truthfulness of Kenzo statements like:

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

where `torus-sc` is a simplicial complex

Theorem for Simplicial Sets from Simplicial Complexes

Proving truthfulness of Kenzo statements like:

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

where `torus-sc` is a simplicial complex

Theorem

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

Main Tools

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*. Preprint

Main Tools

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*. Preprint

- 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 15 definitions and 375 theorems (+ 77 definitions and 601 theorems)

Generic simplicial set theory

- Generic simplicial set theory for simplicial complexes:
  - From 4 definitions and 4 theorems
  - Instantiates 15 definitions and 375 theorems (+ 77 definitions and 601 theorems)

Theorem

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

Table of Contents

Conclusions and Further Work

- Conclusions:

Conclusions and Further Work

- Conclusions:
  - New module for the Kenzo system

Conclusions and Further Work

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

Conclusions and Further Work

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

Conclusions and Further Work

- 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

- 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

- 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
  - New modules for Kenzo and certification of their correctness

ACL2 verification of Simplicial Complexes programs for the Kenzo system

Jónathan Heras and Vico Pascual

Departamento de Matemáticas y Computación
Universidad de La Rioja
Spain

April 21, 2010