The GAP package simpcomp

A toolbox for simplicial complexes

Munich, July 26th, 2010

Felix Effenberger, Jonathan Spreer

University of Stuttgart
- Purpose: working with (abstract) simplicial complexes in the context of combinatorial topology.
- Purpose: working with (abstract) simplicial complexes in the context of combinatorial topology.

- Goal: easy to use in an interactive way.
Purpose: working with (abstract) simplicial complexes in the context of combinatorial topology.

Goal: easy to use in an interactive way.

Uses the convenient GAP environment including command completion, inline help system and a comprehensive amount of already implemented functionalities.
The GAP package simpcomp

- Purpose: working with (abstract) simplicial complexes in the context of combinatorial topology.

- Goal: easy to use in an interactive way.

- Uses the convenient GAP environment including command completion, inline help system and a comprehensive amount of already implemented functionalities.

- In particular makes use of GAP packets GRAPE (Soicher, McKay) and homology (Dumas et al.) as well as several GAP programs by Frank Lutz, TU Berlin.
The GAP package simpcomp

- Purpose: working with (abstract) simplicial complexes in the context of combinatorial topology.

- Goal: easy to use in an interactive way.

- Uses the convenient GAP environment including command completion, inline help system and a comprehensive amount of already implemented functionalities.

- In particular makes use of GAP packets GRAPE (Soicher, McKay) and homology (Dumas et al.) as well as several GAP programs by Frank Lutz, TU Berlin.

- Submitted to the GAP Group.
The GAP package simpcomp

- Purpose: working with (abstract) simplicial complexes in the context of combinatorial topology.

- Goal: easy to use in an interactive way.

- Uses the convenient GAP environment including command completion, inline help system and a comprehensive amount of already implemented functionalities.

- In particular makes use of GAP packets GRAPE (Soicher, McKay) and homology (Dumas et al.) as well as several GAP programs by Frank Lutz, TU Berlin.

- Submitted to the GAP Group.

- Function prefix: SC...
simpcomp data type

simpcomp is based on a new GAP abstract data type SCsimplicialComplex:
simpcomp data type

simpcomp is based on a new GAP abstract data type SCSimplicialComplex:

The abstract data type is designed
simpcomp data type

simpcomp is based on a new GAP abstract data type SCSimplicialComplex:

The abstract data type is designed

- to take care of data consistency (in particular to ensure consistent vertex labeling),
simpcomp data type

simpcomp is based on a new GAP abstract data type SCSimplicialComplex:

The abstract data type is designed

- to take care of data consistency (in particular to ensure consistent vertex labeling),

- to cache already known properties of a complex, thus, avoiding unnecessary double calculations,
simpcomp data type

simpcomp is based on a new GAP abstract data type SCSimplicialComplex:

The abstract data type is designed

- to take care of data consistency (in particular to ensure consistent vertex labeling),

- to cache already known properties of a complex, thus, avoiding unnecessary double calculations,

- to store, to load and to access simplicial complexes via user libraries and pseudo object orientation in an easy way.
The main features of simpcomp are:

- Construction of complexes from
The main features of simpcomp are:

- Construction of complexes from facet lists,
The main features of simpcomp are:

- **Construction of complexes from**
  - facet lists,
  - existing ones using standard operations (connected sum, (simplicial) Cartesian product, handle additions, etc.),
The main features of simpcomp are:

- **Construction of complexes** from
  - facet lists,
  - existing ones using standard operations (connected sum, (simplicial) Cartesian product, handle additions, etc.),
  - orbit representatives,
**simpcomp – functions by area**

The main features of simpcomp are:

- **Construction of complexes** from
  - facet lists,
  - existing ones using standard operations (connected sum, (simplicial) Cartesian product, handle additions, etc.),
  - orbit representatives,
  - difference cycles,

...and many more (Version 1.3.0 provides 262 functions and 160 pages of documentation).
The main features of simpcomp are:

- **Construction of complexes** from
  - facet lists,
  - existing ones using standard operations (connected sum, (simplicial) Cartesian product, handle additions, etc.),
  - orbit representatives,
  - difference cycles,
- **Library of known triangulations**, user libraries,
The main features of simpcomp are:

- **Construction of complexes** from
  - facet lists,
  - existing ones using standard operations (connected sum, (simplicial) Cartesian product, handle additions, etc.),
  - orbit representatives,
  - difference cycles,
- **Library of known triangulations**, user libraries,
- **Property calculation** of complexes,
The main features of simpcomp are:

- **Construction of complexes** from
  - facet lists,
  - existing ones using standard operations (connected sum, (simplicial) Cartesian product, handle additions, etc.),
  - orbit representatives,
  - difference cycles,

- **Library of known triangulations**, user libraries,

- **Property calculation** of complexes,

- **Bistellar flips**,
The main features of simpcomp are:

- **Construction of complexes** from
  - facet lists,
  - existing ones using standard operations (connected sum, (simplicial) Cartesian product, handle additions, etc.),
  - orbit representatives,
  - difference cycles,

- **Library of known triangulations**, user libraries,

- **Property calculation of complexes**, 

- **Bistellar flips**, 

- **Automorphism groups**, 

...and many more (Version 1.3.0 provides 262 functions and 160 pages of documentation).
simpcomp – functions by area

The main features of simpcomp are:

- **Construction of complexes from**
  - facet lists,
  - existing ones using standard operations (connected sum, (simplicial) Cartesian product, handle additions, etc.),
  - orbit representatives,
  - difference cycles,

- **Library of known triangulations**, user libraries,

- Property calculation of complexes,

- **Bistellar flips**, 

- **Automorphism groups**, 

- **(Co-)Homology basis computation, cup product, intersection form**, 

...and many more (Version 1.3.0 provides 262 functions and 160 pages of documentation).
simpcomp – functions by area

The main features of simpcomp are:

- **Construction of complexes** from
  - facet lists,
  - existing ones using standard operations (connected sum, (simplicial) Cartesian product, handle additions, etc.),
  - orbit representatives,
  - difference cycles,

- **Library of known triangulations**, user libraries,

- Property calculation of complexes,

- Bistellar flips,

- Automorphism groups,

- (Co-)Homology basis computation, cup product, intersection form,

- Fundamental Group computation,
The main features of simpcomp are:

- **Construction of complexes from**
  - facet lists,
  - existing ones using standard operations (connected sum, (simplicial) Cartesian product, handle additions, etc.),
  - orbit representatives,
  - difference cycles,

- **Library of known triangulations**, user libraries,
- **Property calculation of complexes**,
- **Bistellar flips**,
- **Automorphism groups**,
- **(Co-)Homology basis computation, cup product, intersection form**,
- **Fundamental Group computation**,
- **Polyhedral Morse theory**, normal surfaces and slicings,
The main features of simpcomp are:

- **Construction of complexes from**
  - facet lists,
  - existing ones using standard operations (connected sum, (simplicial) Cartesian product, handle additions, etc.),
  - orbit representatives,
  - difference cycles,

- **Library of known triangulations**, user libraries,

- **Property calculation of complexes**,

- **Bistellar flips**,

- **Automorphism groups**,

- **(Co-)Homology basis computation, cup product, intersection form**,

- **Fundamental Group computation**,

- **Polyhedral Morse theory**, normal surfaces and slicings,

- ...and many more (Version 1.3.0 provides 262 functions and 160 pages of documentation).
Example: 4-manifolds

There is a combinatorial K3 surface due to M. Casella and W. Kühnel with the minimum number of 16 vertices. It has been presented in terms of the complex obtained by the automorphism group \( G \cong AGL(1, \mathbb{F}_{16}) \) given by

\[
G = \left\{ (12)(34)(56)(78)(910)(1112)(1314)(1516), \\
(13)(24)(57)(68)(911)(1012)(1315)(1416), \\
(15)(26)(37)(48)(913)(1014)(1115)(1216), \\
(2131511143581674910612) \right\},
\]

acting on the two generating simplices \( \Delta_1 = \{2, 3, 4, 5, 9\} \) and \( \Delta_2 = \{2, 5, 7, 10, 11\} \).
There is a combinatorial K3 surface due to M. Casella and W. Kühnel with the minimum number of 16 vertices. It has been presented in terms of the complex obtained by the automorphism group $G \cong AGL(1, \mathbb{F}_{16})$ given by


acting on the two generating simplices $\Delta_1 = \{2, 3, 4, 5, 9\}$ and $\Delta_2 = \{2, 5, 7, 10, 11\}$.

**Question:** Is this really a K3 surface?
Example: Generating a complex

Let us check:
Let us check:

```gap
G := Group((1,2)(3,4)(5,6)(7,8)(9,10)(11,12)(13,14)(15,16),
(1,3)(2,4)(5,7)(6,8)(9,11)(10,12)(13,15)(14,16),
(1,5)(2,6)(3,7)(4,8)(9,13)(10,14)(11,15)(12,16),
(1,9)(2,10)(3,11)(4,12)(5,13)(6,14)(7,15)(8,16),
(2,13,15,11,14,3,5,8,16,7,4,9,10,6,12));

K3 := SCFromGenerators(G, [[2,3,4,5,9], [2,5,7,10,11]]);
```

Properties known: AutomorphismGroup, AutomorphismGroupSize,
AutomorphismGroupStructure, AutomorphismGroupTransitivity,
Dim, Facets, Generators, Name, VertexLabels.

Name="complex from generators under group ((C2 x C2 x C2 x C2) : C5) : C3"
Dim=4
AutomorphismGroupSize=240
AutomorphismGroupStructure="((C2 x C2 x C2 x C2) : C5) : C3"
AutomorphismGroupTransitivity=2
```

/SimplicialComplex]
Example: $f$-vector, homology and bistellar moves

First compute the $f$-vector, the Euler characteristic and the homology groups of $K3$: 

```plaintext
gap> K3.F; 
[ 16, 120, 560, 720, 288 ]
gap> K3.Chi; 
24

gap> K3.Homology; 
[ [ 0, [ ] ], [ 0, [ ] ], [ 22, [ ] ], [ 0, [ ] ], [ 1, [ ] ] ]
```

Now verify that $K3$ is a combinatorial manifold using a heuristic algorithm based on bistellar moves:

```plaintext
gap> K3.IsManifold; 
true
```
Example: $f$-vector, homology and bistellar moves

First compute the $f$-vector, the Euler characteristic and the homology groups of $K3$:

```
gap> K3.F;
[ 16, 120, 560, 720, 288 ]
gap> K3.Chi;
24

gap> K3.Homology;
[ [ 0, [ ] ], [ 0, [ ] ], [ 22, [ ] ], [ 0, [ ] ], [ 1, [ ] ] ]
```

Now verify that $K3$ is a combinatorial manifold using a heuristic algorithm based on bistellar moves:
Example: \( f \)-vector, homology and bistellar moves

First compute the \( f \)-vector, the Euler characteristic and the homology groups of \( K3 \):

\[
\text{gap> K3.F;}
\]
\[
[ 16, 120, 560, 720, 288 ]
\]

\[
\text{gap> K3.Chi;}
\]
\[
24
\]

\[
\text{gap> K3.Homology;}
\]
\[
[ [ 0, [ ] ], [ 0, [ ] ], [ 22, [ ] ], [ 0, [ ] ], [ 1, [ ] ] ]
\]

Now verify that \( K3 \) is a combinatorial manifold using a heuristic algorithm based on bistellar moves:

\[
\text{gap> K3.IsManifold;}
\]
\[
\text{true}
\]
Example: Intersection form and neighborliness

K3 is simply connected (compute fundamental group or check that the complex is 3-neighborly):
Example: Intersection form and neighborliness

K3 is simply connected (compute fundamental group or check that the complex is 3-neighborly):

```gap
gap> K3.FundamentalGroup;
<fp group with 105 generators>
gap> Size(last);
1
gap> K3.Neighborliness;
3
```

Compute the parity and the signature of the intersection form of K3:
Example: Intersection form and neighborliness

K3 is simply connected (compute fundamental group or check that the complex is 3-neighborly):

```
gap> K3.FundamentalGroup;
<fp group with 105 generators>
gap> Size(last);
1
gap> K3.Neighborliness;
3
```

Compute the parity and the signature of the intersection form of K3:

```
gap> K3.IntersectionFormParity;
0
gap> K3.IntersectionFormSignature;
[ 22, 3, 19 ]
```

This means that the intersection form of K3 is even, has dimension 22 and signature $19 - 3 = 16$. 
It now follows from a theorem of M. Freedman that the complex is in fact homeomorphic to a K3 surface. Furthermore, K3 is a tight triangulation as can be verified as follows:
It now follows from a theorem of M. Freedman that the complex is in fact homeomorphic to a K3 surface. Furthermore, K3 is a tight triangulation as can be verified as follows:

\[
gap> \text{K3.IsTight};
\]

#I SCIsTight: complex is (k+1)-neighborly 2k-manifold and thus tight.  
true

Of course K3 is also in simpcomp's built-in library:
Example: Tightness and library

It now follows from a theorem of M. Freedman that the complex is in fact homeomorphic to a K3 surface. Furthermore, K3 is a tight triangulation as can be verified as follows:

```gap
gap> K3.IsTight;
#I  SCIsTight: complex is (k+1)-neighborly 2k-manifold and thus tight.
true

Of course K3 is also in simpcomp's built-in library:

```gap
gap> SCLib.SearchByName("K3");
[ [ 7489, "K3 surface" ] ]
gap> c:=SCLib.Load(7489);;
gap> SCIsIsomorphic(c,K3);
true
gap>
```
Example: Polyhedral Morse theory

We can also have a look at the multiplicity vectors of the perfect polyhedral Morse function that orders the vertices $v_1, \ldots, v_{16}$ linearly:
We can also have a look at the multiplicity vectors of the perfect polyhedral Morse function that orders the vertices $v_1, \ldots, v_{16}$ linearly:

```gap
gap> SCMorseIsPerfect(K3,[1..16]);
true
gap> SCMorseMultiplicityVector(K3,[1..16]);
[[[1, 0, 0, 0, 0], [0, 0, 0, 0, 0], [0, 0, 0, 0, 0], [0, 0, 1, 0, 0], [0, 0, 2, 0, 0],
  [0, 0, 1, 0, 0], [0, 0, 4, 0, 0], [0, 0, 3, 0, 0], [0, 0, 3, 0, 0], [0, 0, 4, 0, 0],
  [0, 0, 1, 0, 0], [0, 0, 0, 0, 0], [0, 0, 0, 0, 0], [0, 0, 0, 0, 1]]
```

Finally, we could check if there are other known examples of, say, 3-neighborly 4-manifolds:
We can also have a look at the multiplicity vectors of the perfect polyhedral Morse function that orders the vertices $v_1, \ldots, v_{16}$ linearly:

```gap
gap> SCMorseIsPerfect(K3,[1..16]);
true
gap> SCMorseMultiplicityVector(K3,[1..16]);
[ [ 1, 0, 0, 0, 0 ], [ 0, 0, 0, 0, 0 ], [ 0, 0, 0, 0, 0 ], [ 0, 0, 1, 0, 0 ],
[ 0, 0, 2, 0, 0 ], [ 0, 0, 1, 0, 0 ], [ 0, 0, 4, 0, 0 ], [ 0, 0, 3, 0, 0 ],
[ 0, 0, 3, 0, 0 ], [ 0, 0, 4, 0, 0 ], [ 0, 0, 1, 0, 0 ], [ 0, 0, 2, 0, 0 ],
[ 0, 0, 1, 0, 0 ], [ 0, 0, 0, 0, 0 ], [ 0, 0, 0, 0, 0 ], [ 0, 0, 0, 0, 1 ] ]
```

Finally, we could check if there are other known examples of, say, 3-neighborly 4-manifolds:

```gap
gap> SCLib.SearchByAttribute("Binomial(F[1],3) = F[3] and Dim=4");
[ [ 16, "CP^2 (VT)" ], [ 7489, "K3 surface" ] ]
gap> CP2:=SCLib.Load(16);;
gap> SCPropertyByName(CP2,"Reference");
"manifold_4_9_13_1 in F.H.Lutz: 'The Manifold Page',
```

As a matter of fact there is a 3-neighborly triangulation of the complex projective plane which can be found in the work mentioned above.
Example: Handlebodies and collapsing

There is a series of bounded $d$-manifolds on $2d + 2$ vertices given as the difference cycle $\{1 : 1 : \ldots : 1 : d + 3\}$. In dimension 7 we have
Example: Handlebodies and collapsing

There is a series of bounded $d$-manifolds on $2d + 2$ vertices given as the difference cycle \{(1 : 1 : \ldots : 1 : d + 3)\}. In dimension 7 we have

gap> c := SCFromDifferenceCycles([[1, 1, 1, 1, 1, 1, 1, 9]]);;
gap> SCHomology(c);
[[0, []], [1, []], [0, []], [0, []], [0, []], [0, []], [0, []], [0, []]]
gap> coll := SCCollapseGreedy(c);;
gap> SCFacets(coll);
[ [ 7, 14 ], [ 7, 15 ], [ 14, 15 ] ]
gap>

and, thus, a handle body. Its boundary is the sphere product $S^5 \times S^1$: 
Example: Handlebodies and collapsing

There is a series of bounded $d$-manifolds on $2d + 2$ vertices given as the difference cycle $\{(1:1: \ldots :1:d+3)\}$. In dimension 7 we have

```gap
gap> c := SCFromDifferenceCycles([[1, 1, 1, 1, 1, 1, 1, 9]]);

gap> SCHomology(c);
[[0, []], [1, []], [0, []], [0, []], [0, []], [0, []], [0, []], [0, []]]

gap> coll := SCCollapseGreedy(c);

gap> SCFacets(coll);
[[7, 14], [7, 15], [14, 15]]

and, thus, a handle body. Its boundary is the sphere product $S^5 \times S^1$:

```gap
gap> bd := SCBoundary(c);

gap> bd.Homology;
[[0, []], [1, []], [0, []], [0, []], [0, []], [1, []], [1, []]]

which follows by the simplicial homology groups and a theorem due to Kreck.
Example: Bistellar moves

If we quickly want to produce example manifolds of a certain topological type, simpcomp offers a variety of functionalities to ease the construction process. Let us say we want to construct the 3-manifold \((S^2 \times S^1)^\#^3\):

```
gap> s2s1:=SCCartesianProduct(SCBdSimplex(2),SCBdSimplex(3));;
gap> s2s13:=SCConnectedProduct(s2s1,3);;
gap> s2s13.F;
[ 28, 132, 208, 104 ]
gap> res:=SCReduceComplex(s2s13);

```

This means we constructed a version of \((S^2 \times S^1)^\#^3\) with \(\hat{3}^\hat{4}\) vertices using standard techniques and reduced it via bistellar moves to a version with only \(16\) vertices (although this cannot be expected to be the minimal number of vertices needed).
Example: Bistellar moves

If we quickly want to produce example manifolds of a certain topological type `simpcomp` offers a variety of functionalities to ease the construction process. Let us say we want to construct the 3-manifold \((\mathbb{S}^2 \times \mathbb{S}^1)^\#^3\):

```
gap> s2s1:=SCCartesianProduct(SCBdSimplex(2),SCBdSimplex(3));;
gap> s2s13:=SCConnectedProduct(s2s1,3);;
gap> s2s13.F;
[ 28, 132, 208, 104 ]
gap> res:=SCRReduceComplex(s2s13);;
gap> res[1];
false
gap> res[2].F;
[ 16, 84, 136, 68 ]
gap>
```

This means we constructed a version of \((\mathbb{S}^2 \times \mathbb{S}^1)^\#^3\) with \(((3 \cdot 4) \cdot 3) – 2 \cdot 4 = 28\) vertices using standard techniques and reduced it via bistellar moves to a version with only 16 vertices (although this can not be expected to be the minimal number of vertices needed).
simpcomp supports slicings, i.e. the pre-image of a regular point of a polyhedral Morse function. In the case of 3-manifolds slicings coincide with normal surfaces and are returned in the form of another abstract data type SCNSNormalSurface.
simpcomp supports slicings, i.e. the pre-image of a regular point of a polyhedral Morse function. In the case of 3-manifolds slicings coincide with normal surfaces and are returned in the form of another abstract data type SCNSNormalSurface.

Slicing of a lens space of type $L(3, 1)$ inducing a genus 1 handle body decomposition of $L(3, 1)$. 
We can use simpcomp to construct non-combinatorial triangulations of manifolds as well: The double suspension of Poincaré’s homology sphere, also called “Edward’s sphere”, is a well-known example for such a complex:
Example: Fundamental Group and double suspension

We can use simpcomp to construct non-combinatorial triangulations of manifolds as well: The double suspension of Poincaré’s homology sphere, also called “Edward’s sphere”, is a well-known example for such a complex:

```
gap> SCLib.SearchByName("Poincare_sphere");
[ [ 7465, "Poincare_sphere" ] ]
gap> c:=SCLib.Load(7465);;
gap> SCHomology(c);
[ [ 0, [  ] ], [ 0, [  ] ], [ 0, [  ] ], [ 1, [  ] ] ]
gap> SCFundamentalGroup(c);
<fp group with 91 generators>
gap> Size(last);
120
gap> es:=SCSuspension(SCSuspension(c));; # is homeomorphic to S^5
gap> lk:=SCLink(es,[[1,17],[2,1]]);; # should be a sphere...
gap> SCIsIsomorphic(lk,c); # ...but is a Poincare-sphere
true
gap>
```
But how do we learn about all the commands?

Let us assume we want to learn more about the library functionalities. By the command completion (tab) of GAP we get:

```
gap> SCLib
SCLib
SCLibAdd
...
SCLibSearchByAttribute
...
```

Let us say, we are interested in the function `SCLibSearchByAttribute`, so we type in:

```
gap> ?SCLibSearchByAttribute
```

Help: Showing `simpcomp: SCLibSearchByAttribute'

```
9.1-10 SCLibSearchByAttribute
> SCLibSearchByAttribute( repository, expr ) ...
...and get directly to the documentation of the package.
But how do we learn about all the commands?
Let us assume we want to learn more about the library functionalities. By the command completion (tab) of GAP we get
But how do we learn about all the commands?
Let us assume we want to learn more about the library functionalities.
By the command completion (tab) of GAP we get

gap> SCLib
    SCLib
    SCLibAdd
    ...
    SCLibSearchByAttribute
    ...

Let us say, we are interested in the function SCLibSearchByAttribute, so we type in
But how do we learn about all the commands? Let us assume we want to learn more about the library functionalities. By the command completion (tab) of GAP we get

```
gap> SCLib
   SCLib
   SCLibAdd
   ... 
   SCLibSearchByAttribute
   ...
```

Let us say, we are interested in the function `SCLibSearchByAttribute`, so we type in

```
gap> ?SCLibSearchByAttribute
   Help: Showing ‘simpcomp: SCLibSearchByAttribute’
9.1-10 SCLibSearchByAttribute
```

```
> SCLibSearchByAttribute( repository, expr ) ... 
```

...and get directly to the documentation of the package.
F. Effenberger, J. Spreer *simpcomp* - A GAP package for simplicial complexes, Version 1.3.0, 2010,
http://www.igt.uni-stuttgart.de/LstDiffgeo/simpcomp.