politopix

4.0.0

Generated by Doxygen 1.8.7

Thu Nov 24 2016 12:55:01

Contents

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Chapter 4

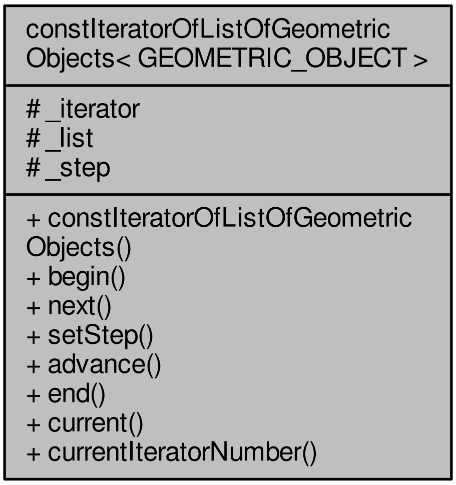
Class Documentation

4.1 constIteratorOfListOfGeometricObjects< GEOMETRIC_OBJECT > Class Template Reference

This class is designed to run the list of all geometric objects representing a polytope.

`#include <GeometricObjectIterator_Rn.h>`

Collaboration diagram for constIteratorOfListOfGeometricObjects< GEOMETRIC_OBJECT >:


constIteratorOfListOfGeometric
Objects< GEOMETRIC_OBJECT >
# _iterator
# _list
# _step
+ constIteratorOfListOfGeometric
Objects()
+ begin()
+ next()
+ setStep()
+ advance()
+ end()
+ current()
+ currentIteratorNumber()

Public Member Functions

- constIteratorOfListOfGeometricObjects (const listOfGeometricObjects< GEOMETRIC_OBJECT > &l)

  Constructor.
- void begin ()

  Move the iterator at the beginning of the list.
- void next ()

  Move the iterator one step forward.

- void setStep (unsigned int n)

  Step forward in the list geometric elements.
- void advance (unsigned int n)

  Step forward in the list geometric elements.
- bool end () const

  Tell whether we have reached the end of the list.
- const GEOMETRIC_OBJECT current ()

  Return the current geometric element.
- int currentIteratorNumber () const

  Return the current position in the list.

Protected Attributes

- unsigned int _iterator

  The current position in the list.
- const listOfGeometricObjects
  < GEOMETRIC_OBJECT > & _list

  The actual list of geometric elements.
- unsigned int _step

  To perform a step.

4.1.1 Detailed Description

template<class GEOMETRIC_OBJECT>class constIteratorOfListOfGeometricObjects< GEOMETRIC_OBJECT >

This class is designed to run the list of all geometric objects representing a polytope.

Definition at line 142 of file GeometricObjectIterator_Rn.h.

4.1.2 Constructor & Destructor Documentation

4.1.2.1 template<class GEOMETRIC_OBJECT > constIteratorOfListOfGeometricObjects< GEOMETRIC_OBJECT >::constIteratorOfListOfGeometricObjects (const listOfGeometricObjects< GEOMETRIC_OBJECT > & *l*) [inline]

Constructor.

Definition at line 146 of file GeometricObjectIterator_Rn.h.

4.1.3 Member Function Documentation

4.1.3.1 template<class GEOMETRIC_OBJECT > void constIteratorOfListOfGeometricObjects< GEOMETRIC_OBJECT >::advance (unsigned int *n*) [inline]

Step forward in the list geometric elements.

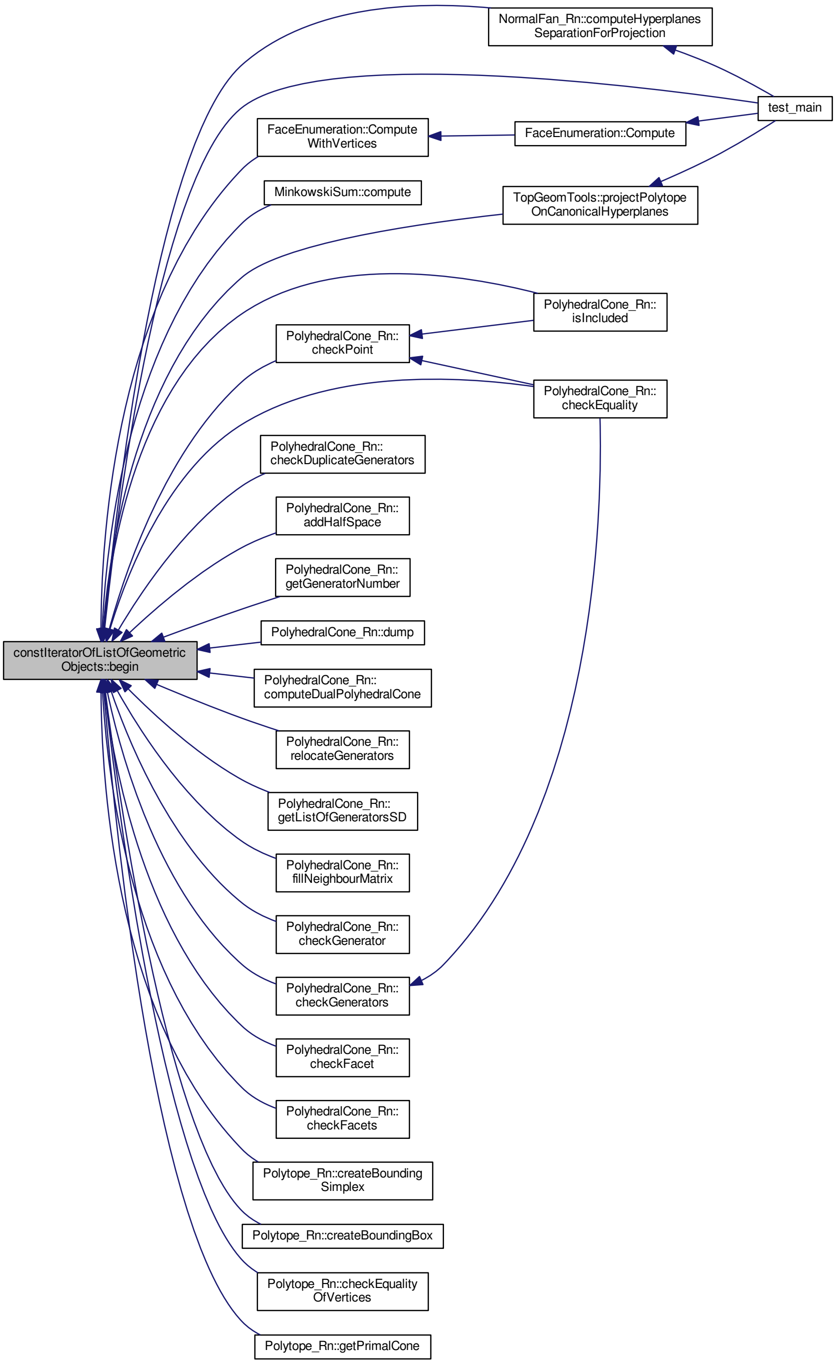
Definition at line 159 of file GeometricObjectIterator_Rn.h.

4.1.3.2 template<class GEOMETRIC_OBJECT > void constIteratorOfListOfGeometricObjects< GEOMETRIC_OBJECT >::begin ( ) [inline]

Move the iterator at the beginning of the list.

Definition at line 150 of file GeometricObjectIterator_Rn.h.

Here is the caller graph for this function:


NormalFan_Rn::computeHyperplanes SeparationForProjection
test_main
FaceEnumeration::Compute WithVertices
FaceEnumeration::Compute
MinkowskiSum::compute
TopGeomTools::projectPolytope OnCanonicalHyperplanes
PolyhedralCone_Rn:: isIncluded
PolyhedralCone_Rn:: checkPoint
PolyhedralCone_Rn:: checkEquality
PolyhedralCone_Rn:: checkDuplicateGenerators
PolyhedralCone_Rn:: addHalfSpace
PolyhedralCone_Rn:: getGeneratorNumber
PolyhedralCone_Rn::dump
constIteratorOfListOfGeometric Objects::begin
PolyhedralCone_Rn:: computeDualPolyhedralCone
PolyhedralCone_Rn:: relocateGenerators
PolyhedralCone_Rn:: getListOfGeneratorsSD
PolyhedralCone_Rn:: fillNeighbourMatrix
PolyhedralCone_Rn:: checkGenerator
PolyhedralCone_Rn:: checkGenerators
PolyhedralCone_Rn:: checkFacet
PolyhedralCone_Rn:: checkFacets
Polytope_Rn::createBounding Simplex
Polytope_Rn::createBoundingBox
Polytope_Rn::checkEquality OfVertices
Polytope_Rn::getPrimalCone

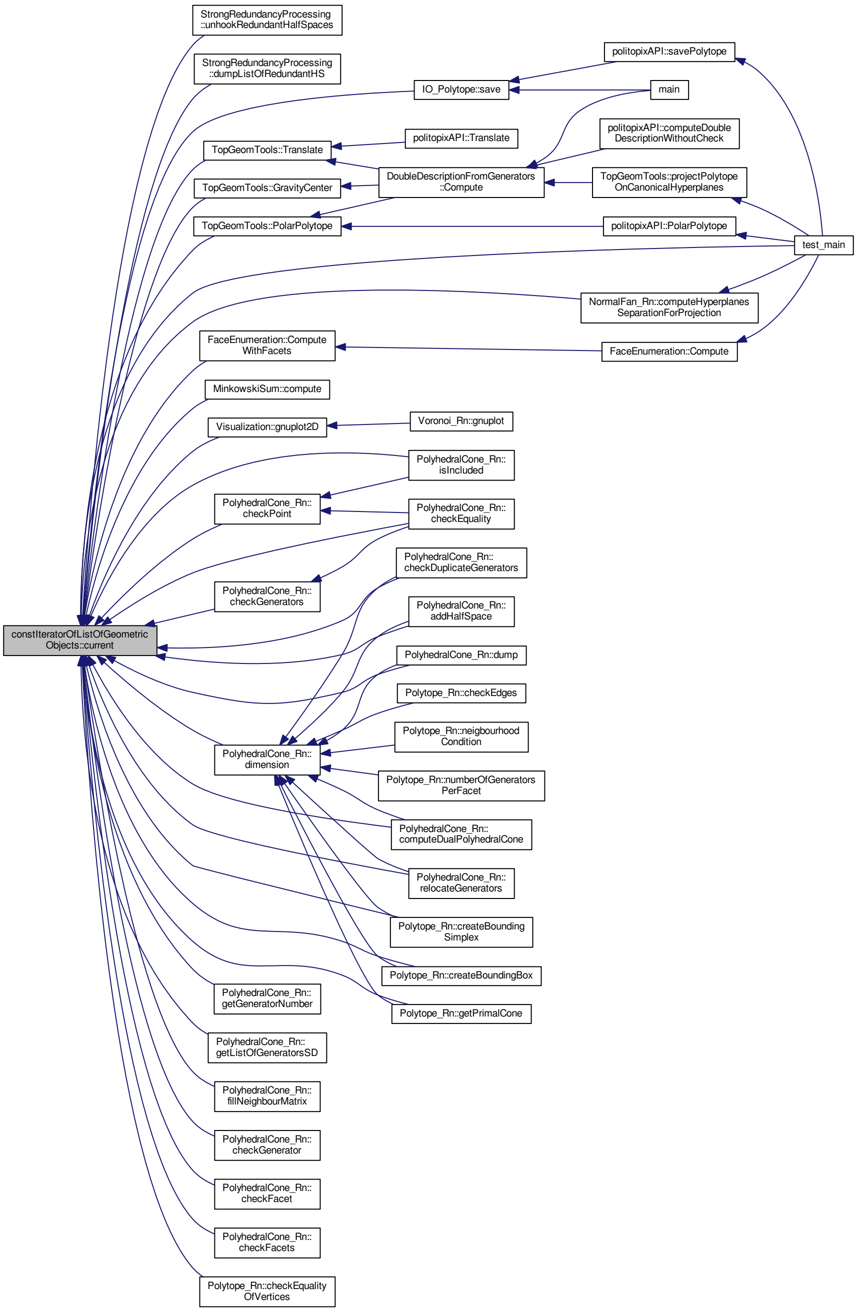
4.1.3.3 template< class GEOMETRIC_OBJECT > const GEOMETRIC_OBJECT constIteratorOfListOfGeometricObjects< GEOMETRIC_OBJECT >::current ( ) [inline]

Return the current geometric element.

Return the current geometric element.

Definition at line 174 of file GeometricObjectIterator_Rn.h.

Here is the caller graph for this function:

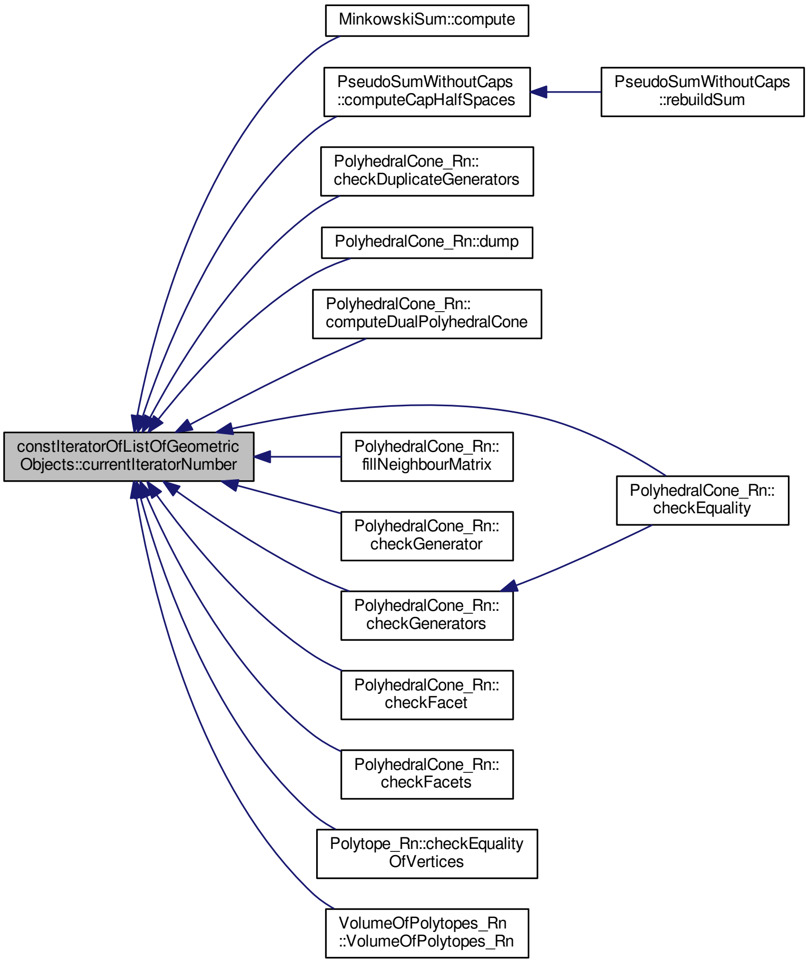

StrongRedundancyProcessing ::unhookRedundantHalfSpaces
politopixAPI::savePolytope
StrongRedundancyProcessing ::dumpListOfRedundantHS
IO_Polytope::save
main
politopixAPI::computeDouble DescriptionWithoutCheck
TopGeomTools::Translate
politopixAPI::Translate
DoubleDescriptionFromGenerators ::Compute
TopGeomTools::projectPolytope OnCanonicalHyperplanes
TopGeomTools::GravityCenter
TopGeomTools::PolarPolytope
politopixAPI::PolarPolytope
test_main
NormalFan_Rn::computeHyperplanes SeparationForProjection
FaceEnumeration::Compute WithFacets
FaceEnumeration::Compute
MinkowskiSum::compute
Visualization::gnuplot2D
Voronoi_Rn::gnuplot
PolyhedralCone_Rn:: isIncluded
PolyhedralCone_Rn:: checkPoint
PolyhedralCone_Rn:: checkEquality
PolyhedralCone_Rn:: checkDuplicateGenerators
PolyhedralCone_Rn:: checkGenerators
PolyhedralCone_Rn:: addHalfSpace
constIteratorOfListOfGeometric Objects::current
PolyhedralCone_Rn::dump
Polytope_Rn::checkEdges
Polytope_Rn::neigbourhood Condition
PolyhedralCone_Rn:: dimension
Polytope_Rn::numberOfGenerators PerFacet
PolyhedralCone_Rn:: computeDualPolyhedralCone
PolyhedralCone_Rn:: relocateGenerators
Polytope_Rn::createBounding Simplex
Polytope_Rn::createBoundingBox
PolyhedralCone_Rn:: getGeneratorNumber
Polytope_Rn::getPrimalCone
PolyhedralCone_Rn:: getListOfGeneratorsSD
PolyhedralCone_Rn:: fillNeighbourMatrix
PolyhedralCone_Rn:: checkGenerator
PolyhedralCone_Rn:: checkFacet
PolyhedralCone_Rn:: checkFacets
Polytope_Rn::checkEquality OfVertices

4.1.3.4 template<class GEOMETRIC_OBJECT > int constIteratorOfListOfGeometricObjects< GEOMETRIC_OBJECT >::currentIteratorNumber ( ) const [inline]

Return the current position in the list.

Definition at line 183 of file GeometricObjectIterator_Rn.h.

Here is the caller graph for this function:

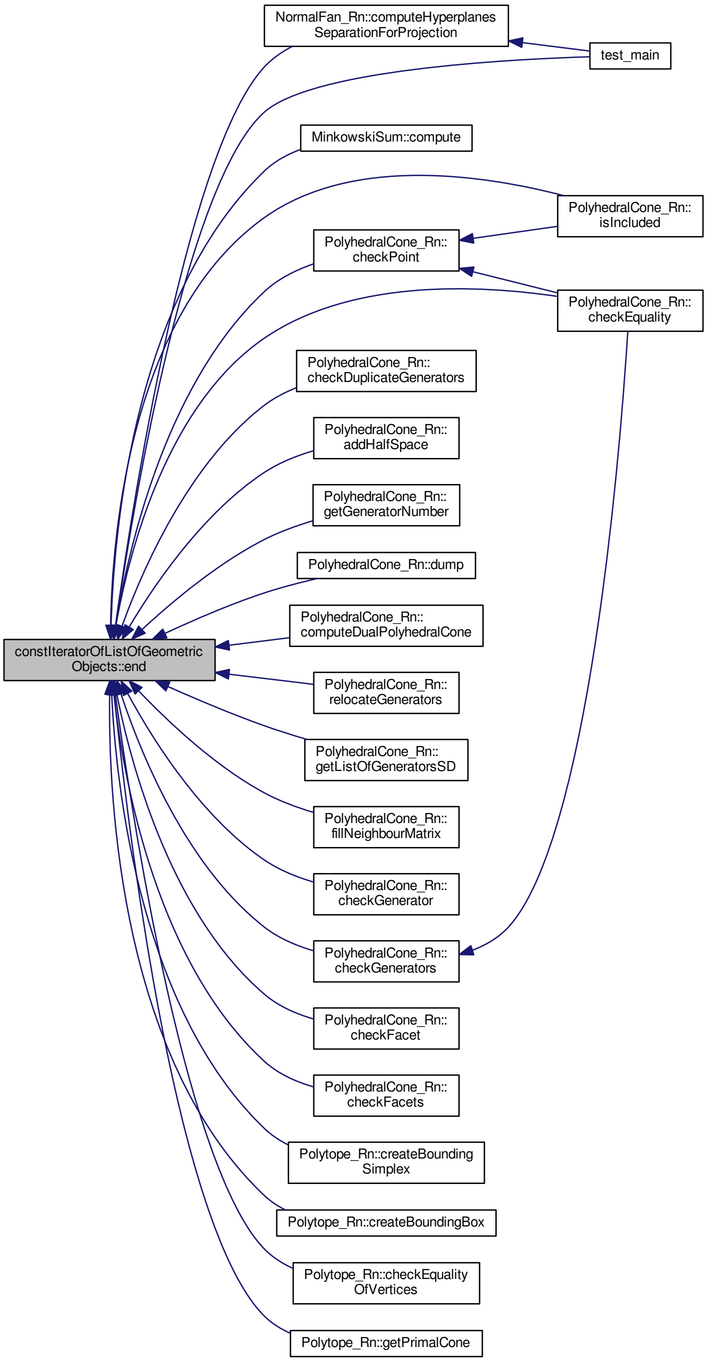

MinkowskiSum::compute
PseudoSumWithoutCaps ::computeCapHalfSpaces
PseudoSumWithoutCaps ::rebuildSum
PolyhedralCone_Rn:: checkDuplicateGenerators
PolyhedralCone_Rn::dump
PolyhedralCone_Rn:: computeDualPolyhedralCone
constIteratorOfListOfGeometric Objects::currentIteratorNumber
PolyhedralCone_Rn:: fillNeighbourMatrix
PolyhedralCone_Rn:: checkEquality
PolyhedralCone_Rn:: checkGenerator
PolyhedralCone_Rn:: checkGenerators
PolyhedralCone_Rn:: checkFacet
PolyhedralCone_Rn:: checkFacets
Polytope_Rn::checkEquality OfVertices
VolumeOfPolytopes_Rn ::VolumeOfPolytopes_Rn

4.1.3.5 template< class GEOMETRIC_OBJECT > bool constIteratorOfListOfGeometricObjects< GEOMETRIC_OBJECT >::end ( ) const [inline]

Tell whether we have reached the end of the list.

Definition at line 171 of file GeometricObjectIterator_Rn.h.

Here is the caller graph for this function:

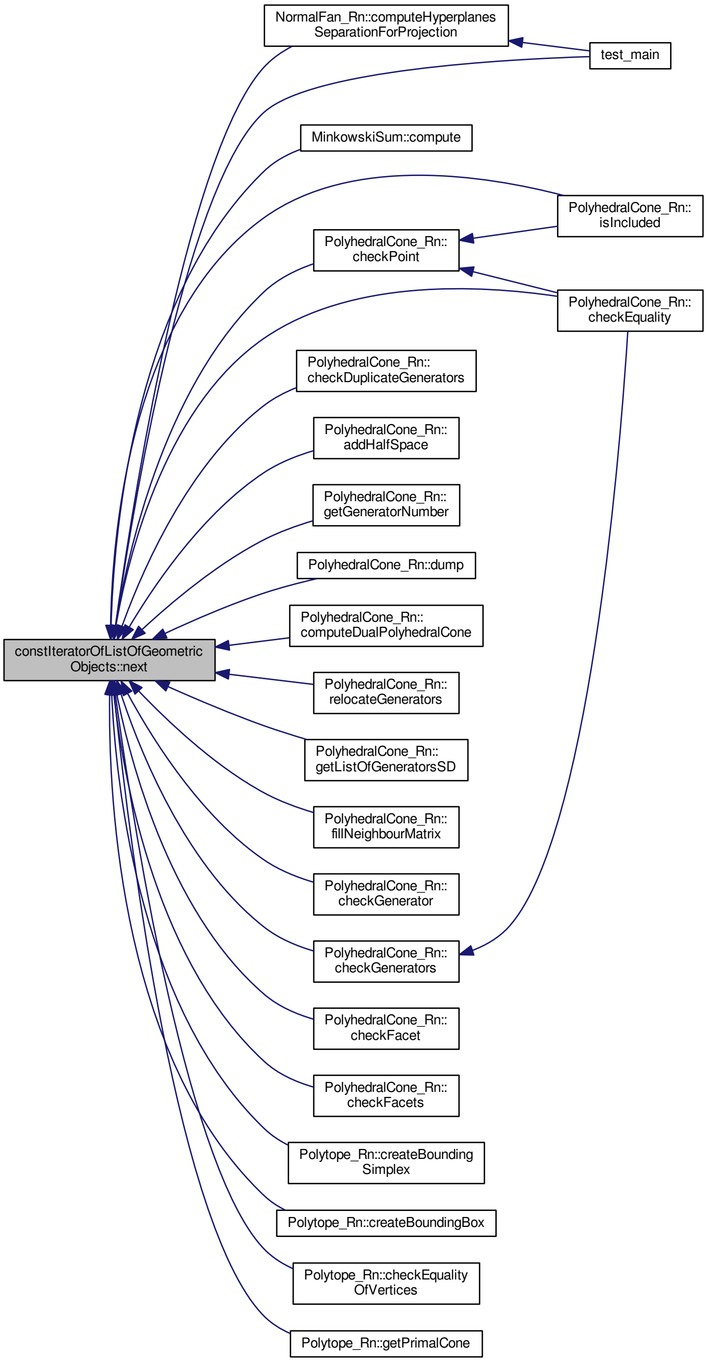

NormalFan_Rn::computeHyperplanes SeparationForProjection
test_main
MinkowskiSum::compute
PolyhedralCone_Rn:: isIncluded
PolyhedralCone_Rn:: checkPoint
PolyhedralCone_Rn:: checkEquality
PolyhedralCone_Rn:: checkDuplicateGenerators
PolyhedralCone_Rn:: addHalfSpace
PolyhedralCone_Rn:: getGeneratorNumber
PolyhedralCone_Rn::dump
PolyhedralCone_Rn:: computeDualPolyhedralCone
constIteratorOfListOfGeometric Objects::end
PolyhedralCone_Rn:: relocateGenerators
PolyhedralCone_Rn:: getListOfGeneratorsSD
PolyhedralCone_Rn:: fillNeighbourMatrix
PolyhedralCone_Rn:: checkGenerator
PolyhedralCone_Rn:: checkGenerators
PolyhedralCone_Rn:: checkFacet
PolyhedralCone_Rn:: checkFacets
Polytope_Rn::createBounding Simplex
Polytope_Rn::createBoundingBox
Polytope_Rn::checkEquality OfVertices
Polytope_Rn::getPrimalCone

4.1.3.6 template<class GEOMETRIC_OBJECT > void constIteratorOfListOfGeometricObjects< GEOMETRIC_OBJECT >::next ( ) [inline]

Move the iterator one step forward.

Definition at line 153 of file GeometricObjectIterator_Rn.h.

Here is the caller graph for this function:


NormalFan_Rn::computeHyperplanes SeparationForProjection
test_main
MinkowskiSum::compute
PolyhedralCone_Rn:: isIncluded
PolyhedralCone_Rn:: checkPoint
PolyhedralCone_Rn:: checkEquality
PolyhedralCone_Rn:: checkDuplicateGenerators
PolyhedralCone_Rn:: addHalfSpace
PolyhedralCone_Rn:: getGeneratorNumber
PolyhedralCone_Rn::dump
PolyhedralCone_Rn:: computeDualPolyhedralCone
constIteratorOfListOfGeometric Objects::next
PolyhedralCone_Rn:: relocateGenerators
PolyhedralCone_Rn:: getListOfGeneratorsSD
PolyhedralCone_Rn:: fillNeighbourMatrix
PolyhedralCone_Rn:: checkGenerator
PolyhedralCone_Rn:: checkGenerators
PolyhedralCone_Rn:: checkFacet
PolyhedralCone_Rn:: checkFacets
Polytope_Rn::createBounding Simplex
Polytope_Rn::createBoundingBox
Polytope_Rn::checkEquality OfVertices
Polytope_Rn::getPrimalCone

4.1.3.7 template< class GEOMETRIC_OBJECT > void constIteratorOfListOfGeometricObjects< GEOMETRIC_OBJECT >::setStep (unsigned int *n*) [inline]

Step forward in the list geometric elements.

Definition at line 156 of file GeometricObjectIterator_Rn.h.

4.1.4 Member Data Documentation

4.1.4.1 template< class GEOMETRIC_OBJECT > unsigned int constIteratorOfListOfGeometricObjects< GEOMETRIC_OBJECT >::_iterator [protected]

The current position in the list.

Definition at line 187 of file GeometricObjectIterator_Rn.h.

4.1.4.2 template< class GEOMETRIC_OBJECT > const listOfGeometricObjects<GEOMETRIC_OBJECT>& constIteratorOfListOfGeometricObjects< GEOMETRIC_OBJECT >::_list [protected]

The actual list of geometric elements.

Definition at line 189 of file GeometricObjectIterator_Rn.h.

4.1.4.3 template< class GEOMETRIC_OBJECT > unsigned int constIteratorOfListOfGeometricObjects< GEOMETRIC_OBJECT >::_step [protected]

To perform a step.

Definition at line 191 of file GeometricObjectIterator_Rn.h.

The documentation for this class was generated from the following file:

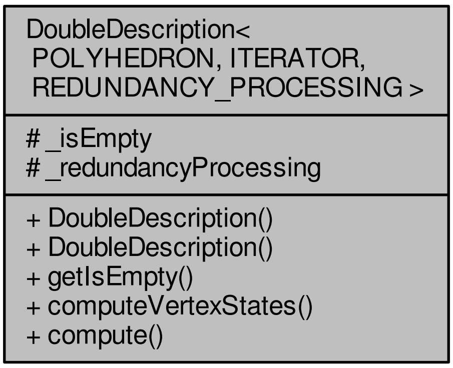
- /home/vindelos/CPP/I2M/politopix/trunk/GeometricObjectIterator_Rn.h

4.2 DoubleDescription< POLYHEDRON, ITERATOR, REDUNDANCY_PROCESSING > Class Template Reference

The algorithm implemented here is an incremental algorithm as mentioned in *How Good are Convex Hull Algorithms?* (1997) by **David Avis** and **David Bremner**. Specific and efficient implementations can be found in *The double description method revisited* (1996) written by **Komei Fukuda** and **Alain Prodon** .
Incremental algorithms for the vertex enumeration problem compute the vertex description by intersecting the defining half-spaces sequentially. An initial simplex is constructed from a subset of *n+1* half-spaces and its vertices and 1-skeleton are computed. Additional half-spaces are introduced sequentially and the vertex description and 1-skeleton are updated at each stage. Essentially such an update amounts to identifying and removing all vertices that are not contained in the new half-space, introducing new vertices for all intersections between edges and the bounding hyperplane of the new half-space, and generating the new edges between these new vertices.
This algorithm can be instantiated by polytopes or polyhedral cones, and as a second argument can be instantiated by iterators such as minindex, lexmin, lexmax.

```
#include <DoubleDescription_Rn.h>
```

Collaboration diagram for DoubleDescription< POLYHEDRON, ITERATOR, REDUNDANCY_PROCESSING >:



Public Member Functions

- DoubleDescription (POLYHEDRON poly, ITERATOR ite, REDUNDANCY_PROCESSING redproc, int truncationStep)
- DoubleDescription (POLYHEDRON poly, ITERATOR ite, REDUNDANCY_PROCESSING redproc, int truncationStep, TrackingOperatorToResult &trackerVdesc, TrackingOperatorToResult &trackerHdesc)
- bool getIsEmpty () const
- void computeVertexStates (std::vector< boost::shared_ptr< Generator_Rn_SD > > &GN_list, const boost::shared_ptr< HalfSpace_Rn > &currentHalfSpace, std::vector< double > &GN_IN_sp, std::vector< double > &GN_OUT_sp, std::vector< boost::shared_ptr< Generator_Rn_SD > > &GN_IN, std::vector< boost::shared_ptr< Generator_Rn_SD > > &GN_OUT, std::vector< boost::shared_ptr< Generator_Rn_SD > > &GN_ON)

  For each generator compute its state according to the current half-space.
- bool compute (POLYHEDRON poly, ITERATOR iteHS, int truncationStep, std::vector< boost::shared_ptr< Generator_Rn_SD > > &listOfGenSD)

  The main function splitting the polyhedron cone or polytope 1-skeleton with a list of half-spaces.

Protected Attributes

- bool _isEmpty

  Store the current state of the intersection.
- REDUNDANCY_PROCESSING _redundancyProcessing

  This class is dedicated to dealing with redundant half-spaces with the desired policy.

4.2.1 Detailed Description

template<class POLYHEDRON, class ITERATOR, class REDUNDANCY_PROCESSING>class DoubleDescription< POLYHEDRON, ITERATOR, REDUNDANCY_PROCESSING >

The algorithm implemented here is an incremental algorithm as mentioned in *How Good are Convex Hull Algorithms?* (1997) by **David Avis** and **David Bremner**. Specific and efficient implementations can be found in *The double description method revisited* (1996) written by **Komei Fukuda** and **Alain Prodon** .

Incremental algorithms for the vertex enumeration problem compute the vertex description by intersecting the defining half-spaces sequentially. An initial simplex is constructed from a subset of *n+1* half-spaces and its vertices and 1-skeleton are computed. Additional half-spaces are introduced sequentially and the vertex description and 1-skeleton are updated at each stage. Essentially such an update amounts to identifying and removing all vertices that are not contained in the new half-space, introducing new vertices for all intersections between edges and the bounding hyperplane of the new half-space, and generating the new edges between these new vertices.
This algorithm can be instantiated by polytopes or polyhedral cones, and as a second argument can be instantiated by iterators such as minindex, lexmin, lexmax.

- minindex : Insert the half-spaces in the order given by the input.

- lexmin : Insert the half-spaces in the the lexicographic increasing order of coefficient vectors.

- lexmax : Insert the half-spaces in the the lexicographic decreasing order of coefficient vectors.

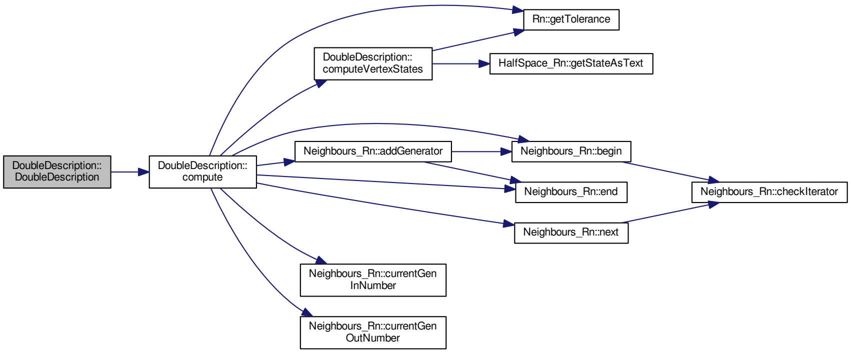
Definition at line 49 of file DoubleDescription_Rn.h.

4.2.2 Constructor & Destructor Documentation

4.2.2.1 template<class POLYHEDRON , class ITERATOR , class REDUNDANCY_PROCESSING > DoubleDescription< POLYHEDRON, ITERATOR, REDUNDANCY_PROCESSING >::DoubleDescription (POLYHEDRON *poly,* ITERATOR *ite,* REDUNDANCY_PROCESSING *redproc,* int *truncationStep*) [inline]

Definition at line 52 of file DoubleDescription_Rn.h.

Here is the call graph for this function:

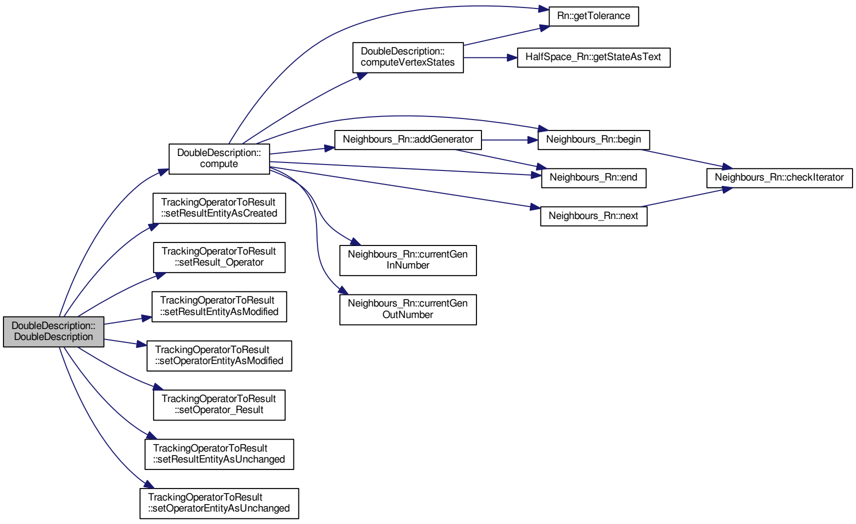


4.2.2.2 template<class POLYHEDRON , class ITERATOR , class REDUNDANCY_PROCESSING > DoubleDescription< POLYHEDRON, ITERATOR, REDUNDANCY_PROCESSING >::DoubleDescription (POLYHEDRON *poly,* ITERATOR *ite,* REDUNDANCY_PROCESSING *redproc,* int *truncationStep,* TrackingOperatorToResult & *trackerVdesc,* TrackingOperatorToResult & *trackerHdesc*) [inline]

Compute the double description tracking all entities and considering the operator1 as the V-description and operator2 as the H-description.

Definition at line 77 of file DoubleDescription_Rn.h.

Here is the call graph for this function:



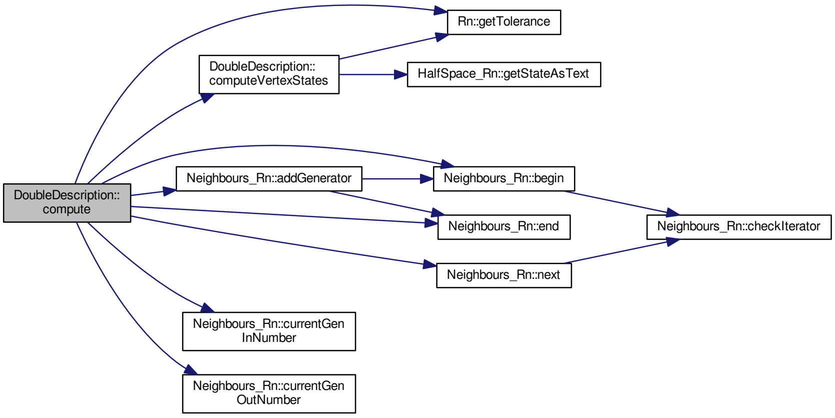
4.2.3 Member Function Documentation

4.2.3.1 template< class POLYHEDRON , class ITERATOR , class REDUNDANCY_PROCESSING > bool DoubleDescription< POLYHEDRON, ITERATOR, REDUNDANCY_PROCESSING >::compute (POLYHEDRON *poly*, ITERATOR *iteHS*, int *truncationStep*, std::vector< boost::shared_ptr< Generator_Rn_SD > > & *listOfGenSD*) [inline]

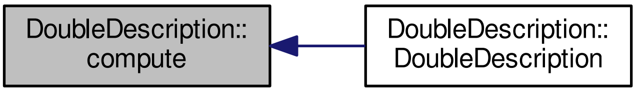
The main function splitting the polyhedron cone or polytope 1-skeleton with a list of half-spaces.

Definition at line 203 of file DoubleDescription_Rn.h.

Here is the call graph for this function:



Here is the caller graph for this function:

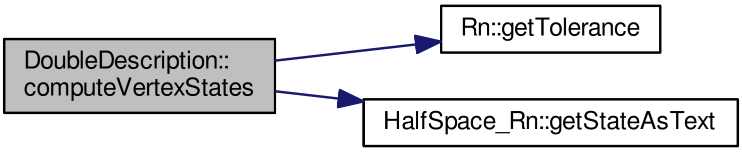


4.2.3.2 template< class POLYHEDRON , class ITERATOR , class REDUNDANCY_PROCESSING > void **DoubleDescription**< POLYHEDRON, ITERATOR, REDUNDANCY_PROCESSING >::computeVertexStates (std::vector< boost::shared_ptr< **Generator_Rn_SD** > > & *GN_list,* const boost::shared_ptr< **HalfSpace_Rn** > & *currentHalfSpace,* std::vector< double > & *GN_IN_sp,* std::vector< double > & *GN_OUT_sp,* std::vector< boost::shared_ptr< **Generator_Rn_SD** > > & *GN_IN,* std::vector< boost::shared_ptr< **Generator_Rn_SD** > > & *GN_OUT,* std::vector< boost::shared_ptr< **Generator_Rn_SD** > > & *GN_ON*) `[inline]`

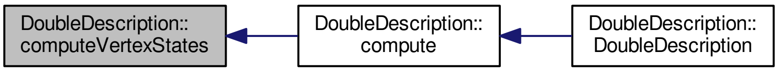
For each generator compute its state according to the current half-space.

Definition at line 138 of file DoubleDescription_Rn.h.

Here is the call graph for this function:



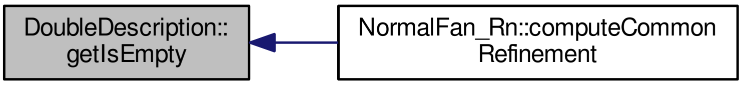
Here is the caller graph for this function:



4.2.3.3 template< class POLYHEDRON , class ITERATOR , class REDUNDANCY_PROCESSING > bool **DoubleDescription**< POLYHEDRON, ITERATOR, REDUNDANCY_PROCESSING >::getIsEmpty ( ) const `[inline]`

Definition at line 135 of file DoubleDescription_Rn.h.

Here is the caller graph for this function:



4.2.4 Member Data Documentation

4.2.4.1 template<class POLYHEDRON , class ITERATOR , class REDUNDANCY_PROCESSING > bool **DoubleDescription**< POLYHEDRON, ITERATOR, REDUNDANCY_PROCESSING >::_isEmpty [protected]

Store the current state of the intersection.

Definition at line 460 of file DoubleDescription_Rn.h.

4.2.4.2 template<class POLYHEDRON , class ITERATOR , class REDUNDANCY_PROCESSING > REDUNDANCY_PROCESSING **DoubleDescription**< POLYHEDRON, ITERATOR, REDUNDANCY_PROCESSING >::_redundancyProcessing [protected]

This class is dedicated to dealing with redundant half-spaces with the desired policy.

Definition at line 462 of file DoubleDescription_Rn.h.

The documentation for this class was generated from the following file:

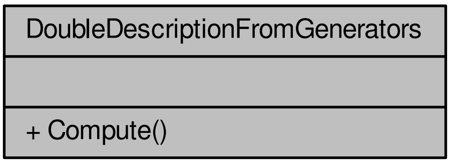
- /home/vindelos/CPP/I2M/politopix/trunk/DoubleDescription_Rn.h

4.3 DoubleDescriptionFromGenerators Class Reference

Compute the V-description from the H-description.

`#include <PolyhedralAlgorithms_Rn.h>`

Collaboration diagram for DoubleDescriptionFromGenerators:



Static Public Member Functions

- static int Compute (boost::shared_ptr< Polytope_Rn > &pol, double bb_size=1000.) throw (invalid_argument, out_of_range, ios_base::failure, logic_error)

  Use the polarity to get the facets from the generators.

4.3.1 Detailed Description

Compute the V-description from the H-description.

Definition at line 340 of file PolyhedralAlgorithms_Rn.h.

4.3.2 Member Function Documentation

4.3.2.1 int DoubleDescriptionFromGenerators::Compute (boost::shared_ptr< **Polytope_Rn** > & *pol,* double *bb_size =* 1000.) throw invalid_argument, out_of_range, ios_base::failure, logic_error) [static]

Use the polarity to get the facets from the generators.

Parameters

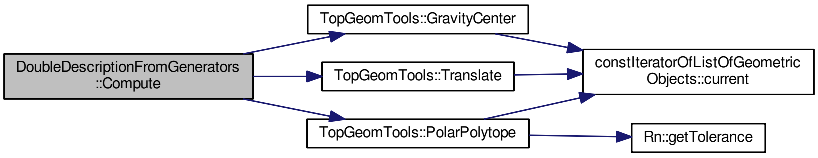
pol	The input polytope

Returns

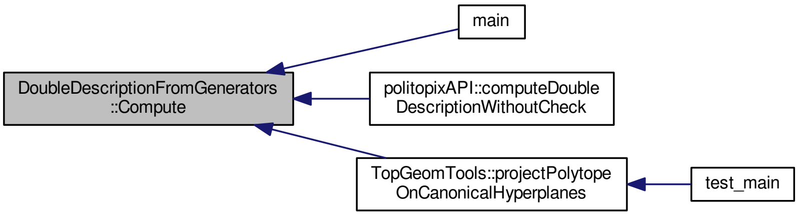
TEST_OK or 0 if the process was successful, TEST_KO or -1 if something went wrong.

Definition at line 1317 of file PolyhedralAlgorithms_Rn.cpp.

Here is the call graph for this function:


DoubleDescriptionFromGenerators ::Compute
TopGeomTools::GravityCenter
TopGeomTools::Translate
TopGeomTools::PolarPolytope
constIteratorOfListOfGeometric Objects::current
Rn::getTolerance

Here is the caller graph for this function:


main
DoubleDescriptionFromGenerators ::Compute
politopixAPI::computeDouble DescriptionWithoutCheck
TopGeomTools::projectPolytope OnCanonicalHyperplanes
test_main

The documentation for this class was generated from the following files:

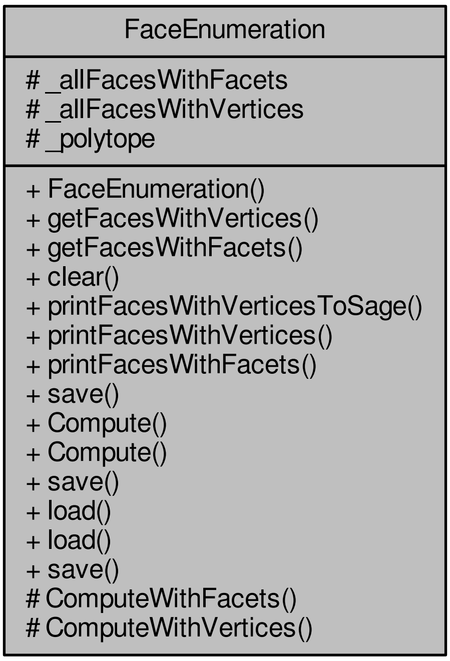
- /home/vindelos/CPP/I2M/politopix/trunk/PolyhedralAlgorithms_Rn.h
- /home/vindelos/CPP/I2M/politopix/trunk/PolyhedralAlgorithms_Rn.cpp

4.4 FaceEnumeration Class Reference

Combinatorial face enumeration for polytopes.

```
#include <PolyhedralAlgorithms_Rn.h>
```

Collaboration diagram for FaceEnumeration:



Public Member Functions

- FaceEnumeration (const boost::shared_ptr< Polytope_Rn > &A)

  General Face Enumeration Algorithm.
- const std::vector< std::vector
  < ListOfFaces > > & getFacesWithVertices () throw (std::domain_error)
- const std::vector< std::vector
  < ListOfFaces > > & getFacesWithFacets () throw (std::domain_error)
- void clear ()
- void printFacesWithVerticesToSage (std::ostream &this_ostream) const
- void printFacesWithVertices (std::ostream &this_ostream) const
- void printFacesWithFacets (std::ostream &this_ostream) const
- void save (std::ostream &this_stream) const

Static Public Member Functions

- static void Compute (const boost::shared_ptr< Polytope_Rn > &A)
- static void Compute (const boost::shared_ptr< Polytope_Rn > &A, FaceEnumeration &FE)
- static void save (const std::string &filename, const std::vector< std::vector< ListOfFaces > > &latt)

  Save the polytope lattice.
- static void load (const std::string &filename, std::vector< std::vector< ListOfFaces > > &latt) throw (std↩ ::ios_base::failure)

  Load the polytope lattice.
- static void load (std::istream &this_stream, std::vector< std::vector< ListOfFaces > > &latt) throw (std::out↩ _of_range)

  Load the polytope lattice 1st line : comments = "# SpaceDimension NumberOfHalfspaces"
  2nd line : SpaceDimension TotalNumberOfFaces
  3rd line : FiDimension k V1 ... Vk
  4th line : FjDimension l Vu ... V(u+l)
  k-th line : ...
- static void save (std::ostream &this_stream, const std::vector< std::vector< ListOfFaces > > &latt)

  Save the polytope lattice 1st line : comments = "# SpaceDimension NumberOfHalfspaces"
  2nd line : SpaceDimension TotalNumberOfFaces
  3rd line : FiDimension k V1 ... Vk
  4th line : FjDimension l Vu ... V(u+l)
  k-th line : ...

Static Protected Member Functions

- static void ComputeWithFacets (const boost::shared_ptr< Polytope_Rn > &A, FaceEnumeration &Face↩ Enum)
- static void ComputeWithVertices (const boost::shared_ptr< Polytope_Rn > &A, FaceEnumeration &Face↩ Enum)

Protected Attributes

- std::vector< std::vector
  < ListOfFaces > > _allFacesWithFacets
- std::vector< std::vector
  < ListOfFaces > > _allFacesWithVertices
- const boost::shared_ptr
  < Polytope_Rn > & _polytope

4.4.1 Detailed Description

Combinatorial face enumeration for polytopes.

Definition at line 44 of file PolyhedralAlgorithms_Rn.h.

4.4.2 Constructor & Destructor Documentation

4.4.2.1 FaceEnumeration::FaceEnumeration (const boost::shared_ptr< Polytope_Rn > & *A*) [inline]

General Face Enumeration Algorithm.

Definition at line 49 of file PolyhedralAlgorithms_Rn.h.

4.4.3 Member Function Documentation

4.4.3.1 void FaceEnumeration::clear ( ) [inline]

Definition at line 89 of file PolyhedralAlgorithms_Rn.h.

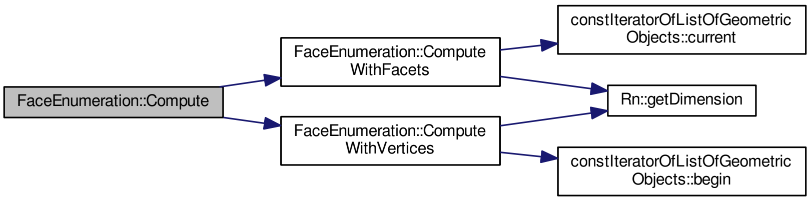
4.4.3.2 void FaceEnumeration::Compute (const boost::shared_ptr< **Polytope_Rn** > & *A*) [static]

General Face Enumeration Algorithm from *Combinatorial face enumeration in convex polytopes* (1994) by **Komei Fukuda** and **Vera Rosta**.
Input: the set $\mathscr{P}_0$ of vertices of a polytope *P*.
Output: the set $\mathscr{P}$ of vertices of a polytope *P*.

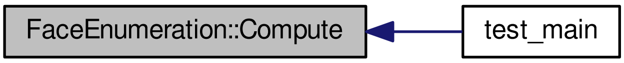
```
procedure FaceEnumeration( $\mathscr{P}_0$: vertices)
Create a binary tree T with set of leaves $\mathscr{P}_0$
k=0; $f_0$ = $|\mathscr{P}_0|$; $\mathscr{P}_{(0)} = \mathscr{P}_0$
WHILE $f_k >= 2$ DO
    $f_{k+1} = 0$ ; $\mathscr{P}_{(k+1)} = \emptyset$
    FOREACH pair (F, F') in $\mathscr{P}_{(k)}$ DO
        $F'' = F \cap F'$
        IF $F'' \notin T$ THEN
            IF $F'' == F$ or $F'' == F'$ THEN
                Delete F" from $\mathscr{P}_{(k)}$;
                $f_k = f_k - 1$
            ENDIF
            Add F" to T and to $\mathscr{P}_{(k+1)}$
            $f_{k+1} = f_{k+1} + 1$
        ENDIF
    ENDFOR
    Output $\mathscr{P}_{(k)}$; k=k+1
ENDWHILE
```

Definition at line 40 of file PolyhedralAlgorithms_Rn.cpp.

Here is the call graph for this function:


FaceEnumeration::Compute
FaceEnumeration::Compute WithFacets
FaceEnumeration::Compute WithVertices
constIteratorOfListOfGeometric Objects::current
Rn::getDimension
constIteratorOfListOfGeometric Objects::begin

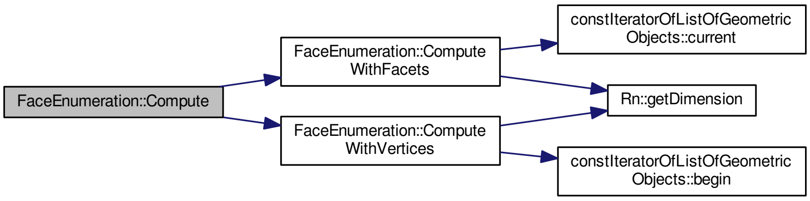
Here is the caller graph for this function:



4.4.3.3 void FaceEnumeration::Compute (const boost::shared_ptr< **Polytope_Rn** > & *A*, **FaceEnumeration** & *FE*) [static]

Definition at line 46 of file PolyhedralAlgorithms_Rn.cpp.

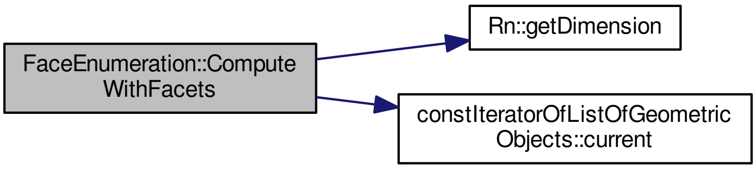
Here is the call graph for this function:


FaceEnumeration::Compute
FaceEnumeration::Compute WithFacets
FaceEnumeration::Compute WithVertices
constIteratorOfListOfGeometric Objects::current
Rn::getDimension
constIteratorOfListOfGeometric Objects::begin

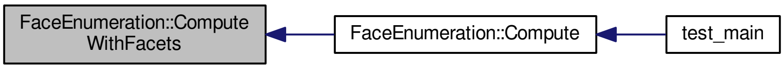
4.4.3.4 void FaceEnumeration::ComputeWithFacets (const boost::shared_ptr< **Polytope_Rn** > & *A*, **FaceEnumeration** & *FaceEnum*) [static], [protected]

Definition at line 51 of file PolyhedralAlgorithms_Rn.cpp.

Here is the call graph for this function:


FaceEnumeration::Compute WithFacets
Rn::getDimension
constIteratorOfListOfGeometric Objects::current

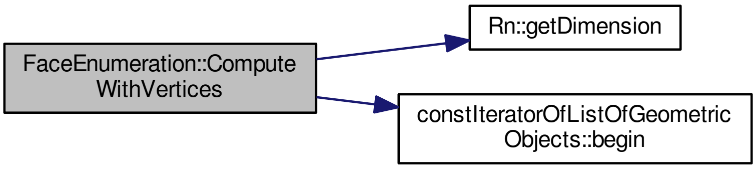
Here is the caller graph for this function:



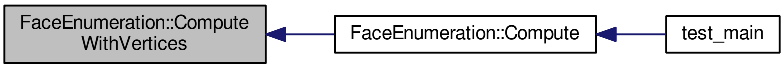
4.4.3.5 void FaceEnumeration::ComputeWithVertices (const boost::shared_ptr< **Polytope_Rn** > & *A*, **FaceEnumeration** & *FaceEnum*) [static], [protected]

Definition at line 124 of file PolyhedralAlgorithms_Rn.cpp.

Here is the call graph for this function:


FaceEnumeration::Compute WithVertices
Rn::getDimension
constIteratorOfListOfGeometric Objects::begin

Here is the caller graph for this function:



4.4.3.6 const std::vector< std::vector< **ListOfFaces** > >& FaceEnumeration::getFacesWithFacets ( ) throw std::domain_error) [inline]

Definition at line 83 of file PolyhedralAlgorithms_Rn.h.

4.4.3.7 const std::vector< std::vector< **ListOfFaces** > >& FaceEnumeration::getFacesWithVertices ( ) throw std::domain_error) [inline]

Definition at line 77 of file PolyhedralAlgorithms_Rn.h.

Here is the caller graph for this function:

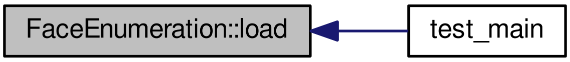
FaceEnumeration::getFaces WithVertices
test_main

4.4.3.8 void FaceEnumeration::load (const std::string & *filename*, std::vector< std::vector< **ListOfFaces** > > & *latt*) throw std::ios_base::failure) [static]

Load the polytope lattice.

Definition at line 262 of file PolyhedralAlgorithms_Rn.cpp.

Here is the caller graph for this function:



4.4.3.9 void FaceEnumeration::load (std::istream & *this_stream*, std::vector< std::vector< **ListOfFaces** > > & *latt*) throw std::out_of_range) [static]

Load the polytope lattice 1st line : comments = "# SpaceDimension NumberOfHalfspaces"
2nd line : SpaceDimension TotalNumberOfFaces
3rd line : FiDimension k V1 ... Vk
4th line : FjDimension l Vu ... V(u+l)
k-th line : ...

Definition at line 287 of file PolyhedralAlgorithms_Rn.cpp.

4.4.3.10 void FaceEnumeration::printFacesWithFacets (std::ostream & *this_ostream*) const

Definition at line 173 of file PolyhedralAlgorithms_Rn.cpp.

4.4.3.11 void FaceEnumeration::printFacesWithVertices (std::ostream & *this_ostream*) const

Definition at line 198 of file PolyhedralAlgorithms_Rn.cpp.

4.4.3.12 void FaceEnumeration::printFacesWithVerticesToSage (std::ostream & *this_ostream*) const

Definition at line 223 of file PolyhedralAlgorithms_Rn.cpp.

4.4.3.13 void FaceEnumeration::save (const std::string & *filename,* const std::vector< std::vector< ListOfFaces > > & *latt*) [static]

Save the polytope lattice.

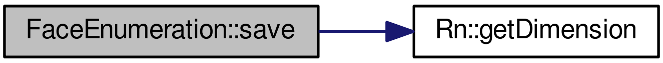
Definition at line 275 of file PolyhedralAlgorithms_Rn.cpp.

4.4.3.14 void FaceEnumeration::save (std::ostream & *this_stream,* const std::vector< std::vector< ListOfFaces > > & *latt*) [static]

Save the polytope lattice 1st line : comments = "# SpaceDimension NumberOfHalfspaces"
2nd line : SpaceDimension TotalNumberOfFaces
3rd line : FiDimension k V1 ... Vk
4th line : FjDimension l Vu ... V(u+l)
k-th line : ...

Definition at line 321 of file PolyhedralAlgorithms_Rn.cpp.

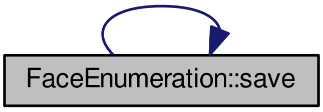
Here is the call graph for this function:



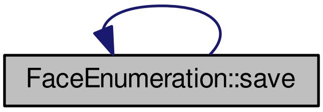
4.4.3.15 void FaceEnumeration::save (std::ostream & *this_stream*) const [inline]

Definition at line 121 of file PolyhedralAlgorithms_Rn.h.

Here is the call graph for this function:



Here is the caller graph for this function:



4.4.4 Member Data Documentation

4.4.4.1 std::vector< std::vector< ListOfFaces > > FaceEnumeration::_allFacesWithFacets [protected]

Definition at line 129 of file PolyhedralAlgorithms_Rn.h.

4.4.4.2 std::vector< std::vector< ListOfFaces > > FaceEnumeration::_allFacesWithVertices [protected]

Definition at line 131 of file PolyhedralAlgorithms_Rn.h.

4.4.4.3 const boost::shared_ptr<Polytope_Rn>& FaceEnumeration::_polytope [protected]

Definition at line 133 of file PolyhedralAlgorithms_Rn.h.

The documentation for this class was generated from the following files:

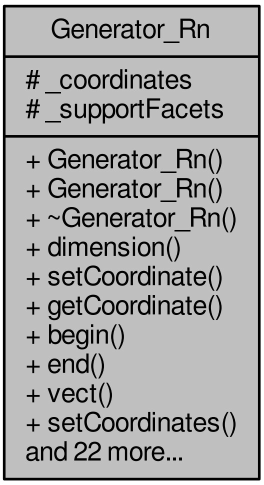
- /home/vindelos/CPP/I2M/politopix/trunk/PolyhedralAlgorithms_Rn.h
- /home/vindelos/CPP/I2M/politopix/trunk/PolyhedralAlgorithms_Rn.cpp

4.5 Generator_Rn Class Reference

A n-coordinates generator, which can be a vertex or an edge whether it is contained by a polytope or a polyhedral cone. It contains all of its support facets.

```
#include <Generator_Rn.h>
```

Collaboration diagram for Generator_Rn:



Public Member Functions

- Generator_Rn (unsigned int n)

  Creates a n-coordinates generator.
- Generator_Rn (const Generator_Rn &gn)

  Copy constructor.
- ∼Generator_Rn ()

  Destructor.
- int dimension () const
- void setCoordinate (unsigned int i, double val)
- double getCoordinate (unsigned int i) const
- vector< double >::const_iterator begin () const
- vector< double >::const_iterator end () const
- const vector< double > & vect () const
- void setCoordinates (const vector< double > &vec)
- void negate ()
- bool isEqual1 (const boost::shared_ptr< Generator_Rn > &gn, unsigned int RnDIM, double TOL2)
- bool isEqual2 (const boost::shared_ptr< Generator_Rn > &gn, unsigned int RnDIM, double TOL2)
- void clearFacets ()

  Clear the list of facets.
- void switchFacets (const std::vector< boost::shared_ptr< HalfSpace_Rn > > &tab)

  Clear the list of facets.
- void setFacet (boost::shared_ptr< HalfSpace_Rn > F)

  Insert a new support facet for the current generator.
- void importFacets (const std::set< boost::shared_ptr< HalfSpace_Rn > > &setOfFacets)

  Insert all facets stored in the argument.
- void exportFacets (std::set< boost::shared_ptr< HalfSpace_Rn > > &setOfFacets) const

  Store all facets in a set.
- void removeFacet (unsigned int i) throw (std::out_of_range,std::domain_error)

Remove the i-th facet in list.
- boost::shared_ptr< HalfSpace_Rn > getFacet (unsigned int i) const throw (std::out_of_range)
- HalfSpace_Rn ∗ getRawFacet (unsigned int i)

  Return the i-th facet as a pointer for very fast comparisons. No check is performed!
- bool isFacetInside (boost::shared_ptr< HalfSpace_Rn > F) const

  Check whether the given half-space is inside the generator's list.
- unsigned int numberOfFacets () const

  Return the total number of support faces.
- void makeDiff (const boost::shared_ptr< Generator_Rn > &gn1, const boost::shared_ptr< Generator_Rn > &gn2)
- void makeSum (const boost::shared_ptr< Generator_Rn > &gn1, const boost::shared_ptr< Generator_Rn > &gn2)
- void makeCoefSum (const boost::shared_ptr< Generator_Rn > &gn1, const boost::shared_ptr< Generator_Rn > &gn2, double coef1, double coef2)
- double getNormalDistance (const boost::shared_ptr< Generator_Rn > &gn1, double coef, unsigned int Rn↩ DIM)
- double normalize ()
- double distanceFrom (const Generator_Rn &P)
- void dump (std::ostream &this_ostream) const
- void load (std::istream &this_istream)
- void save (std::ostream &this_ostream) const

Protected Attributes

- vector< double > _coordinates

  The set of coordinates.
- std::vector< boost::shared_ptr
  < HalfSpace_Rn > > _supportFacets

  Contain the list of all support facets.

Friends

- class Generator_Rn_SD

4.5.1 Detailed Description

A n-coordinates generator, which can be a vertex or an edge whether it is contained by a polytope or a polyhedral cone. It contains all of its support facets.

Definition at line 38 of file Generator_Rn.h.

4.5.2 Constructor & Destructor Documentation

4.5.2.1 Generator_Rn::Generator_Rn (unsigned int *n*)

Creates a n-coordinates generator.

Definition at line 28 of file Generator_Rn.cpp.

4.5.2.2 Generator_Rn::Generator_Rn (const Generator_Rn & *gn*) [inline]

Copy constructor.

Definition at line 46 of file Generator_Rn.h.

4.5.2.3 Generator_Rn::∼Generator_Rn ( )

Destructor.

Definition at line 32 of file Generator_Rn.cpp.

4.5.3 Member Function Documentation

4.5.3.1 vector<double>::const_iterator Generator_Rn::begin ( ) const [inline]

Definition at line 60 of file Generator_Rn.h.

4.5.3.2 void Generator_Rn::clearFacets ( ) [inline]

Clear the list of facets.

Definition at line 95 of file Generator_Rn.h.

4.5.3.3 int Generator_Rn::dimension ( ) const [inline]

Definition at line 54 of file Generator_Rn.h.

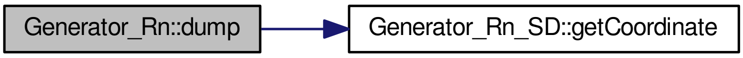
4.5.3.4 double Generator_Rn::distanceFrom (const Generator_Rn & *P*) [inline]

Definition at line 180 of file Generator_Rn.h.

4.5.3.5 void Generator_Rn::dump (std::ostream & *this_ostream*) const [inline]

Definition at line 187 of file Generator_Rn.h.

Here is the call graph for this function:



4.5.3.6 vector<double>::const_iterator Generator_Rn::end ( ) const [inline]

Definition at line 62 of file Generator_Rn.h.

4.5.3.7 void Generator_Rn::exportFacets (std::set< boost::shared_ptr< HalfSpace_Rn > > & *setOfFacets*) const [inline]

Store all facets in a set.

Definition at line 116 of file Generator_Rn.h.

4.5.3.8 double Generator_Rn::getCoordinate (unsigned int *i*) const [inline]

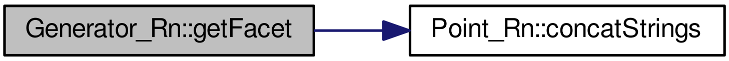
Definition at line 58 of file Generator_Rn.h.

4.5.3.9 boost::shared_ptr< HalfSpace_Rn > Generator_Rn::getFacet (unsigned int *i*) const throw std::out_of_range) [inline]

Return the i-th facet. The user has to check validity of the returned smart pointer.

Definition at line 128 of file Generator_Rn.h.

Here is the call graph for this function:



4.5.3.10 double Generator_Rn::getNormalDistance (const boost::shared_ptr< Generator_Rn > & *gn1,* double *coef,* unsigned int *RnDIM*) [inline]

Return the square distance of the generator gn1 to the straight line defined by _coordinates and passing through the origin.

Definition at line 165 of file Generator_Rn.h.

4.5.3.11 HalfSpace_Rn* Generator_Rn::getRawFacet (unsigned int *i*) [inline]

Return the i-th facet as a pointer for very fast comparisons. No check is performed!

Definition at line 139 of file Generator_Rn.h.

4.5.3.12 void Generator_Rn::importFacets (const std::set< boost::shared_ptr< HalfSpace_Rn > > & *setOfFacets*) [inline]

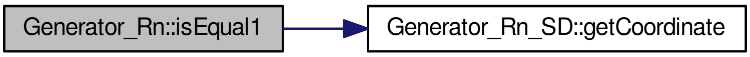
Insert all facets stored in the argument.

Definition at line 107 of file Generator_Rn.h.

4.5.3.13 bool Generator_Rn::isEqual1 (const boost::shared_ptr< Generator_Rn > & *gn,* unsigned int *RnDIM,* double *TOL2*) [inline]

Definition at line 70 of file Generator_Rn.h.

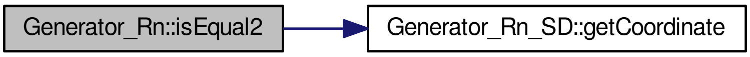
Here is the call graph for this function:



4.5.3.14 bool Generator_Rn::isEqual2 (const boost::shared_ptr< **Generator_Rn** > & *gn,* unsigned int *RnDIM,* double *TOL2*) [inline]

Definition at line 82 of file Generator_Rn.h.

Here is the call graph for this function:



4.5.3.15 bool Generator_Rn::isFacetInside (boost::shared_ptr< **HalfSpace_Rn** > *F*) const

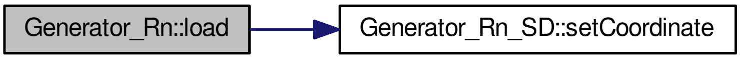
Check whether the given half-space is inside the generator's list.

Definition at line 50 of file Generator_Rn.cpp.

4.5.3.16 void Generator_Rn::load (std::istream & *this_istream*) [inline]

Definition at line 198 of file Generator_Rn.h.

Here is the call graph for this function:



4.5.3.17 void Generator_Rn::makeCoefSum (const boost::shared_ptr< **Generator_Rn** > & *gn1,* const boost::shared_ptr< **Generator_Rn** > & *gn2,* double *coef1,* double *coef2*) [inline]

Definition at line 155 of file Generator_Rn.h.

4.5.3.18 void Generator_Rn::makeDiff (const boost::shared_ptr< **Generator_Rn** > & *gn1*, const boost::shared_ptr< **Generator_Rn** > & *gn2*) [inline]

Definition at line 147 of file Generator_Rn.h.

4.5.3.19 void Generator_Rn::makeSum (const boost::shared_ptr< **Generator_Rn** > & *gn1*, const boost::shared_ptr< **Generator_Rn** > & *gn2*) [inline]

Definition at line 151 of file Generator_Rn.h.

4.5.3.20 void Generator_Rn::negate ( ) [inline]

Definition at line 68 of file Generator_Rn.h.

4.5.3.21 double Generator_Rn::normalize ( ) [inline]

Definition at line 174 of file Generator_Rn.h.

4.5.3.22 unsigned int Generator_Rn::numberOfFacets ( ) const [inline]

Return the total number of support faces.

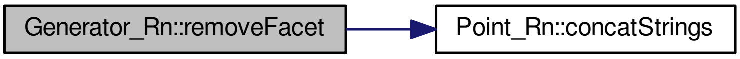
Definition at line 145 of file Generator_Rn.h.

4.5.3.23 void Generator_Rn::removeFacet (unsigned int *i*) throw std::out_of_range, std::domain_error)

Remove the i-th facet in list.

Definition at line 35 of file Generator_Rn.cpp.

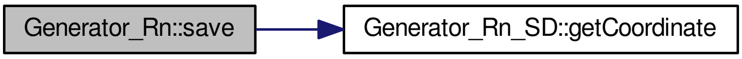
Here is the call graph for this function:



4.5.3.24 void Generator_Rn::save (std::ostream & *this_ostream*) const [inline]

Definition at line 206 of file Generator_Rn.h.

Here is the call graph for this function:



4.5.3.25 void Generator_Rn::setCoordinate (unsigned int *i*, double *val*) [inline]

Definition at line 56 of file Generator_Rn.h.

4.5.3.26 void Generator_Rn::setCoordinates (const vector< double > & *vec*) [inline]

Definition at line 66 of file Generator_Rn.h.

4.5.3.27 void Generator_Rn::setFacet (boost::shared_ptr< **HalfSpace_Rn** > *F*) [inline]

Insert a new support facet for the current generator.

Definition at line 104 of file Generator_Rn.h.

4.5.3.28 void Generator_Rn::switchFacets (const std::vector< boost::shared_ptr< **HalfSpace_Rn** > > & *tab*) [inline]

Clear the list of facets.

Definition at line 98 of file Generator_Rn.h.

4.5.3.29 const vector<double>& Generator_Rn::vect ( ) const [inline]

Definition at line 64 of file Generator_Rn.h.

4.5.4 Friends And Related Function Documentation

4.5.4.1 friend class **Generator_Rn_SD** [friend]

Definition at line 39 of file Generator_Rn.h.

4.5.5 Member Data Documentation

4.5.5.1 vector<double> Generator_Rn::_coordinates [protected]

The set of coordinates.

Definition at line 216 of file Generator_Rn.h.

4.5.5.2 std::vector< boost::shared_ptr<HalfSpace_Rn> > Generator_Rn::_supportFacets [protected]

Contain the list of all support facets.

Definition at line 218 of file Generator_Rn.h.

The documentation for this class was generated from the following files:

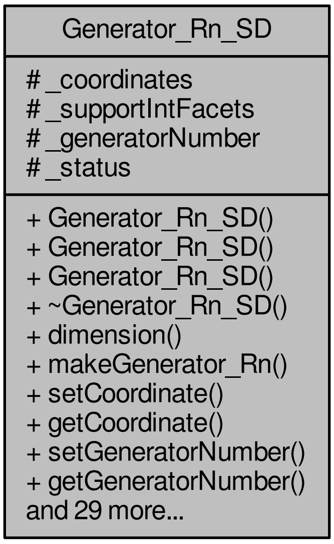
- /home/vindelos/CPP/I2M/politopix/trunk/Generator_Rn.h
- /home/vindelos/CPP/I2M/politopix/trunk/Generator_Rn.cpp

4.6 Generator_Rn_SD Class Reference

A n-coordinates generator for internal data structure. It can be a vertex or an edge whether it is embedded in a polytope or a polyhedral cone. It contains all of its support facets.

`#include <Generator_Rn.h>`

Collaboration diagram for Generator_Rn_SD:



Public Types

- enum Status {
  UNCHANGED, MODIFIED, CREATED, CREATED_AND_MODIFIED,
  DELETED, UNKNOWN }

Public Member Functions

- Generator_Rn_SD (unsigned int n, unsigned int nb, Status st)

  Creates a n-coordinates generator.
- Generator_Rn_SD (const Generator_Rn_SD &gn)

Copy constructor.

- Generator_Rn_SD (const Generator_Rn &gn, unsigned int nb, Status st)

  Constructor with a Generator_Rn.
- ∼Generator_Rn_SD ()

  Destructor.
- int dimension () const
- boost::shared_ptr< Generator_Rn > makeGenerator_Rn () const

  To make a Generator_Rn out of a Generator_Rn_SD.
- void setCoordinate (unsigned int i, double val)
- double getCoordinate (unsigned int i) const
- void setGeneratorNumber (unsigned int gn)
- unsigned int getGeneratorNumber () const
- void setStatus (Status st)
- Status getStatus () const
- vector< double >::const_iterator begin () const
- vector< double >::const_iterator end () const
- const vector< double > & vect () const
- void negate ()
- bool isEqual1 (const boost::shared_ptr< Generator_Rn_SD > &gn, unsigned int RnDIM, double TOL2)
- bool isEqual2 (const boost::shared_ptr< Generator_Rn_SD > &gn, unsigned int RnDIM, double TOL2)
- void setFacet (unsigned int F)

  Insert a new support facet for the current generator.
- void setAllFacets (const std::vector< unsigned int > &AF)

  Insert a new support facet for the current generator.
- void importFacets (const std::set< unsigned int > &setOfFacets)

  Insert all facets stored in the argument.
- void exportFacets (std::set< unsigned int > &setOfFacets) const

  Store all facets in a set.
- void removeFacet (unsigned int i) throw (std::out_of_range,std::domain_error)

  Remove the i-th facet in list.
- unsigned int getFacet (unsigned int i) const throw (std::out_of_range)

  Return the i-th facet number.
- unsigned int getRawFacet (unsigned int i)

  Return the i-th facet. No check is performed!
- bool isFacetInside (unsigned int F) const

  Check whether the given half-space is inside the generator's list.
- void orderFacets ()
- std::vector< unsigned int >
  ::const_iterator facetsBegin () const
- std::vector< unsigned int >
  ::const_iterator facetsEnd () const
- unsigned int numberOfFacets () const

  Return the total number of support faces.
- void makeDiff (const boost::shared_ptr< Generator_Rn_SD > &gn1, const boost::shared_ptr< Generator_Rn_SD > &gn2)
- void makeSum (const boost::shared_ptr< Generator_Rn_SD > &gn1, const boost::shared_ptr< Generator_Rn_SD > &gn2)
- void makeCoefSum (const boost::shared_ptr< Generator_Rn_SD > &gn1, const boost::shared_ptr< Generator_Rn_SD > &gn2, double coef1, double coef2)
- double getNormalDistance (const boost::shared_ptr< Generator_Rn_SD > &gn1, double coef, unsigned int RnDIM)
- double normalize ()
- double distanceFrom (const Generator_Rn_SD &P)
- void dump (std::ostream &this_ostream) const
- void load (std::istream &this_istream)
- void save (std::ostream &this_ostream) const

Protected Attributes

- vector< double > _coordinates

  The set of coordinates.
- std::vector< unsigned int > _supportIntFacets

  Contain the list of all support facets.
- unsigned int _generatorNumber

  The SD generator embeds its own number.
- Status _status

  The SD generator embeds its status to trace the operations.

Friends

- class Generator_Rn

4.6.1 Detailed Description

A n-coordinates generator for internal data structure. It can be a vertex or an edge whether it is embedded in a polytope or a polyhedral cone. It contains all of its support facets.

Definition at line 225 of file Generator_Rn.h.

4.6.2 Member Enumeration Documentation

4.6.2.1 enum Generator_Rn_SD::Status

Enumerator

UNCHANGED

MODIFIED

CREATED

CREATED_AND_MODIFIED

DELETED

UNKNOWN

Definition at line 230 of file Generator_Rn.h.

4.6.3 Constructor & Destructor Documentation

4.6.3.1 Generator_Rn_SD::Generator_Rn_SD (unsigned int *n*, unsigned int *nb*, Status *st*) `[inline]`

Creates a n-coordinates generator.

Definition at line 245 of file Generator_Rn.h.

4.6.3.2 Generator_Rn_SD::Generator_Rn_SD (const Generator_Rn_SD & *gn*) `[inline]`

Copy constructor.

Definition at line 248 of file Generator_Rn.h.

4.6.3.3 Generator_Rn_SD::Generator_Rn_SD (const Generator_Rn & *gn*, unsigned int *nb*, Status *st*) [inline]

Constructor with a Generator_Rn.

Definition at line 254 of file Generator_Rn.h.

4.6.3.4 Generator_Rn_SD::∼Generator_Rn_SD ( ) [inline]

Destructor.

Definition at line 259 of file Generator_Rn.h.

4.6.4 Member Function Documentation

4.6.4.1 vector<double>::const_iterator Generator_Rn_SD::begin ( ) const [inline]

Definition at line 282 of file Generator_Rn.h.

4.6.4.2 int Generator_Rn_SD::dimension ( ) const [inline]

Definition at line 261 of file Generator_Rn.h.

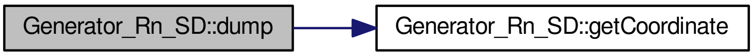
4.6.4.3 double Generator_Rn_SD::distanceFrom (const Generator_Rn_SD & *P*) [inline]

Definition at line 410 of file Generator_Rn.h.

4.6.4.4 void Generator_Rn_SD::dump (std::ostream & *this_ostream*) const [inline]

Definition at line 417 of file Generator_Rn.h.

Here is the call graph for this function:



4.6.4.5 vector<double>::const_iterator Generator_Rn_SD::end ( ) const [inline]

Definition at line 284 of file Generator_Rn.h.

4.6.4.6 void Generator_Rn_SD::exportFacets (std::set< unsigned int > & *setOfFacets*) const [inline]

Store all facets in a set.

Definition at line 330 of file Generator_Rn.h.

4.6.4.7 std::vector<unsigned int>::const_iterator Generator_Rn_SD::facetsBegin ( ) const [inline]

Definition at line 370 of file Generator_Rn.h.

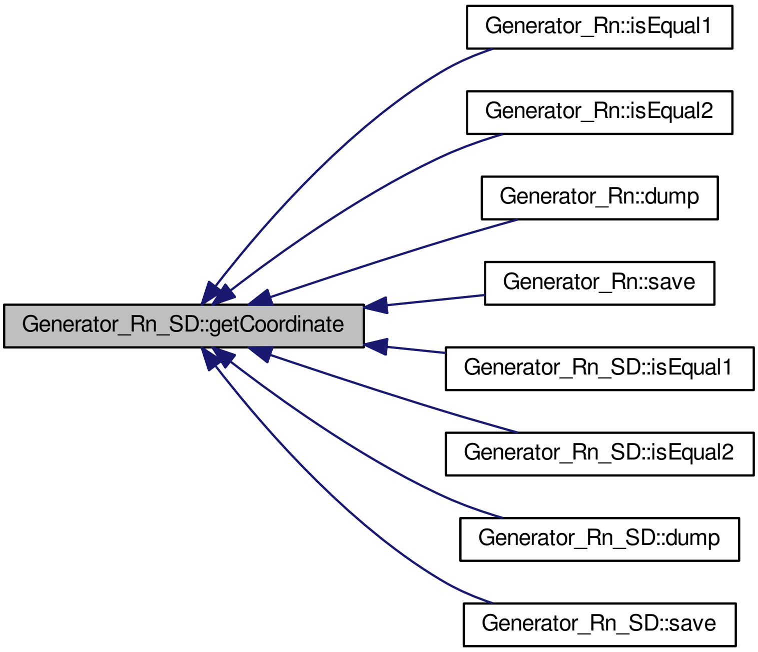
4.6.4.8 std::vector<unsigned int>::const_iterator Generator_Rn_SD::facetsEnd ( ) const [inline]

Definition at line 372 of file Generator_Rn.h.

4.6.4.9 double Generator_Rn_SD::getCoordinate (unsigned int *i*) const [inline]

Definition at line 272 of file Generator_Rn.h.

Here is the caller graph for this function:

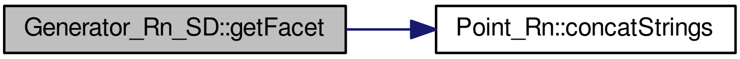


4.6.4.10 unsigned int Generator_Rn_SD::getFacet (unsigned int *i*) const throw std::out_of_range) [inline]

Return the i-th facet number.

Definition at line 345 of file Generator_Rn.h.

Here is the call graph for this function:



4.6.4.11 unsigned int Generator_Rn_SD::getGeneratorNumber ( ) const [inline]

Definition at line 276 of file Generator_Rn.h.

4.6.4.12 double Generator_Rn_SD::getNormalDistance (const boost::shared_ptr< Generator_Rn_SD > & *gn1*, double *coef*, unsigned int *RnDIM*) [inline]

Return the square distance of the generator gn1 to the straight line defined by _coordinates and passing through the origin.

Definition at line 395 of file Generator_Rn.h.

4.6.4.13 unsigned int Generator_Rn_SD::getRawFacet (unsigned int *i*) [inline]

Return the i-th facet. No check is performed!

Definition at line 356 of file Generator_Rn.h.

4.6.4.14 Status Generator_Rn_SD::getStatus ( ) const [inline]

Definition at line 280 of file Generator_Rn.h.

4.6.4.15 void Generator_Rn_SD::importFacets (const std::set< unsigned int > & *setOfFacets*) [inline]

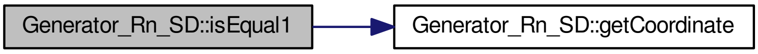
Insert all facets stored in the argument.

Definition at line 321 of file Generator_Rn.h.

4.6.4.16 bool Generator_Rn_SD::isEqual1 (const boost::shared_ptr< Generator_Rn_SD > & *gn*, unsigned int *RnDIM*, double *TOL2*) [inline]

Definition at line 290 of file Generator_Rn.h.

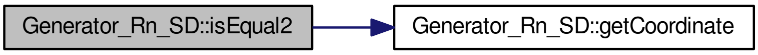
Here is the call graph for this function:



4.6.4.17 bool Generator_Rn_SD::isEqual2 (const boost::shared_ptr< **Generator_Rn_SD** > & *gn*, unsigned int *RnDIM*, double *TOL2*) [inline]

Definition at line 302 of file Generator_Rn.h.

Here is the call graph for this function:



4.6.4.18 bool Generator_Rn_SD::isFacetInside (unsigned int *F*) const [inline]

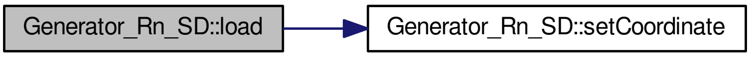
Check whether the given half-space is inside the generator's list.

Definition at line 359 of file Generator_Rn.h.

4.6.4.19 void Generator_Rn_SD::load (std::istream & *this_istream*) [inline]

Definition at line 428 of file Generator_Rn.h.

Here is the call graph for this function:



4.6.4.20 void Generator_Rn_SD::makeCoefSum (const boost::shared_ptr< **Generator_Rn_SD** > & *gn1*, const boost::shared_ptr< **Generator_Rn_SD** > & *gn2*, double *coef1*, double *coef2*) [inline]

Definition at line 385 of file Generator_Rn.h.

4.6.4.21 void Generator_Rn_SD::makeDiff (const boost::shared_ptr< **Generator_Rn_SD** > & *gn1*, const boost::shared_ptr< **Generator_Rn_SD** > & *gn2*) [inline]

Definition at line 377 of file Generator_Rn.h.

4.6.4.22 boost::shared_ptr<**Generator_Rn**> Generator_Rn_SD::makeGenerator_Rn ( ) const [inline]

To make a Generator_Rn out of a Generator_Rn_SD.

Definition at line 264 of file Generator_Rn.h.

4.6.4.23 void Generator_Rn_SD::makeSum (const boost::shared_ptr< Generator_Rn_SD > & *gn1*, const boost::shared_ptr< Generator_Rn_SD > & *gn2*) [inline]

Definition at line 381 of file Generator_Rn.h.

4.6.4.24 void Generator_Rn_SD::negate ( ) [inline]

Definition at line 288 of file Generator_Rn.h.

4.6.4.25 double Generator_Rn_SD::normalize ( ) [inline]

Definition at line 404 of file Generator_Rn.h.

4.6.4.26 unsigned int Generator_Rn_SD::numberOfFacets ( ) const [inline]

Return the total number of support faces.

Definition at line 375 of file Generator_Rn.h.

4.6.4.27 void Generator_Rn_SD::orderFacets ( ) [inline]

Definition at line 368 of file Generator_Rn.h.

4.6.4.28 void Generator_Rn_SD::removeFacet (unsigned int *i*) throw std::out_of_range, std::domain_error) [inline]

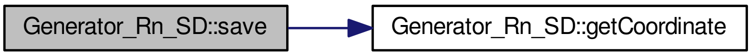
Remove the i-th facet in list.

Definition at line 338 of file Generator_Rn.h.

4.6.4.29 void Generator_Rn_SD::save (std::ostream & *this_ostream*) const [inline]

Definition at line 436 of file Generator_Rn.h.

Here is the call graph for this function:



4.6.4.30 void Generator_Rn_SD::setAllFacets (const std::vector< unsigned int > & *AF*) [inline]

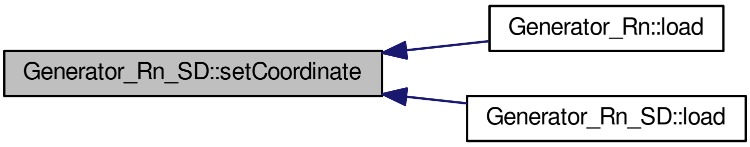
Insert a new support facet for the current generator.

Definition at line 318 of file Generator_Rn.h.

4.6.4.31 void Generator_Rn_SD::setCoordinate (unsigned int *i*, double *val*) [inline]

Definition at line 270 of file Generator_Rn.h.

Here is the caller graph for this function:



4.6.4.32 void Generator_Rn_SD::setFacet (unsigned int *F*) [inline]

Insert a new support facet for the current generator.

Definition at line 315 of file Generator_Rn.h.

4.6.4.33 void Generator_Rn_SD::setGeneratorNumber (unsigned int *gn*) [inline]

Definition at line 274 of file Generator_Rn.h.

4.6.4.34 void Generator_Rn_SD::setStatus (Status *st*) [inline]

Definition at line 278 of file Generator_Rn.h.

4.6.4.35 const vector<double>& Generator_Rn_SD::vect ( ) const [inline]

Definition at line 286 of file Generator_Rn.h.

4.6.5 Friends And Related Function Documentation

4.6.5.1 friend class Generator_Rn [friend]

Definition at line 226 of file Generator_Rn.h.

4.6.6 Member Data Documentation

4.6.6.1 vector<double> Generator_Rn_SD::_coordinates [protected]

The set of coordinates.

Definition at line 446 of file Generator_Rn.h.

4.6.6.2 unsigned int Generator_Rn_SD::_generatorNumber [protected]

The SD generator embeds its own number.

Definition at line 450 of file Generator_Rn.h.

4.6.6.3 Status Generator_Rn_SD::_status [protected]

The SD generator embeds its status to trace the operations.

Definition at line 452 of file Generator_Rn.h.

4.6.6.4 std::vector< unsigned int > Generator_Rn_SD::_supportIntFacets [protected]

Contain the list of all support facets.

Definition at line 448 of file Generator_Rn.h.

The documentation for this class was generated from the following file:

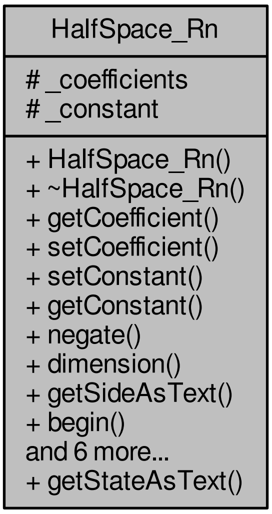
- /home/vindelos/CPP/I2M/politopix/trunk/Generator_Rn.h

4.7 HalfSpace_Rn Class Reference

A half-space whose frontier is a linear (n-1) dimension space.
_constant + _coefficients[0].x1 + ... + _coefficients[n-1].xn >= 0.

`#include <HalfSpace_Rn.h>`

Collaboration diagram for HalfSpace_Rn:



Public Types

- enum State {
  hs_ON = 0, hs_IN = 1, hs_OUT = 2, hs_UNKNOWN = 3,
  hs_IN_OR_OUT = 4 }

Public Member Functions

- HalfSpace_Rn (unsigned int n)

  Constructor.
- ∼HalfSpace_Rn ()
- double getCoefficient (unsigned int i) const throw (std::out_of_range)
- void setCoefficient (unsigned int i, double c) throw (std::out_of_range)
- void setConstant (double c)
- double getConstant () const
- void negate ()
- int dimension () const
- std::string getSideAsText () const
- boost::numeric::ublas::vector
  < double >::const_iterator begin () const
- boost::numeric::ublas::vector
  < double >::const_iterator end () const
- const
  boost::numeric::ublas::vector
  < double > & vect () const
- double computeDistancePointHyperplane (const boost::numeric::ublas::vector< double > &thisPoint) const
- double computeDistancePointHyperplane (const boost::numeric::ublas::vector< double > &thisPoint, double halfSpaceNorm) const
- double computeDistancePointHyperplane (const boost::numeric::ublas::vector< double > &thisPoint, boost↩ ::numeric::ublas::vector< double > &projectedPoint, double halfSpaceNorm) const
- void dump (std::ostream &this_ostream) const

Static Public Member Functions

- static std::string getStateAsText (const HalfSpace_Rn::State &)

Protected Attributes

- boost::numeric::ublas::vector
  < double > _coefficients

  The normal vector.
- double _constant

  The second member constant.

4.7.1 Detailed Description

A half-space whose frontier is a linear (n-1) dimension space.
_constant + _coefficients[0].x1 + ... + _coefficients[n-1].xn >= 0.

Definition at line 37 of file HalfSpace_Rn.h.

4.7.2 Member Enumeration Documentation

4.7.2.1 enum HalfSpace_Rn::State

Enumerator

hs_ON

hs_IN

hs_OUT

hs_UNKNOWN

hs_IN_OR_OUT

Definition at line 41 of file HalfSpace_Rn.h.

4.7.3 Constructor & Destructor Documentation

4.7.3.1 HalfSpace_Rn::HalfSpace_Rn (unsigned int *n*)

Constructor.

Definition at line 29 of file HalfSpace_Rn.cpp.

4.7.3.2 HalfSpace_Rn::∼HalfSpace_Rn ( )

Definition at line 32 of file HalfSpace_Rn.cpp.

4.7.4 Member Function Documentation

4.7.4.1 boost::numeric::ublas::vector<double>::const_iterator HalfSpace_Rn::begin ( ) const [inline]

Definition at line 67 of file HalfSpace_Rn.h.

4.7.4.2 double HalfSpace_Rn::computeDistancePointHyperplane (const boost::numeric::ublas::vector< double > & *thisPoint*) const [inline]

Definition at line 75 of file HalfSpace_Rn.h.

4.7.4.3 double HalfSpace_Rn::computeDistancePointHyperplane (const boost::numeric::ublas::vector< double > & *thisPoint,* double *halfSpaceNorm*) const [inline]

Definition at line 83 of file HalfSpace_Rn.h.

4.7.4.4 double HalfSpace_Rn::computeDistancePointHyperplane (const boost::numeric::ublas::vector< double > & *thisPoint,* boost::numeric::ublas::vector< double > & *projectedPoint,* double *halfSpaceNorm*) const [inline]

Definition at line 89 of file HalfSpace_Rn.h.

4.7.4.5 int HalfSpace_Rn::dimension ( ) const [inline]

Definition at line 63 of file HalfSpace_Rn.h.

4.7.4.6 void HalfSpace_Rn::dump (std::ostream & *this_ostream*) const [inline]

Definition at line 99 of file HalfSpace_Rn.h.

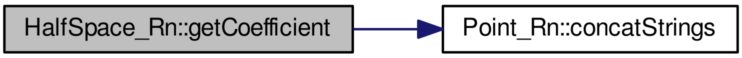
4.7.4.7 boost::numeric::ublas::vector<double>::const_iterator HalfSpace_Rn::end ( ) const [inline]

Definition at line 69 of file HalfSpace_Rn.h.

4.7.4.8 double HalfSpace_Rn::getCoefficient (unsigned int *i*) const throw std::out_of_range)

Definition at line 35 of file HalfSpace_Rn.cpp.

Here is the call graph for this function:



4.7.4.9 double HalfSpace_Rn::getConstant ( ) const [inline]

Definition at line 59 of file HalfSpace_Rn.h.

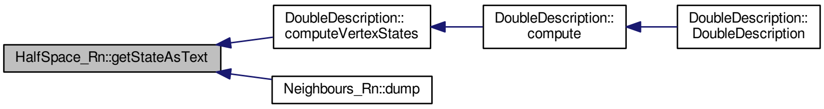
4.7.4.10 std::string HalfSpace_Rn::getSideAsText ( ) const [inline]

Definition at line 65 of file HalfSpace_Rn.h.

4.7.4.11 std::string HalfSpace_Rn::getStateAsText (const HalfSpace_Rn::State & *state*) [static]

Definition at line 57 of file HalfSpace_Rn.cpp.

Here is the caller graph for this function:



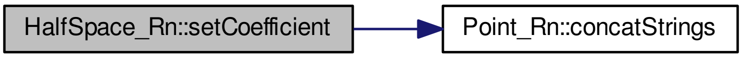
4.7.4.12 void HalfSpace_Rn::negate ( ) [inline]

Definition at line 61 of file HalfSpace_Rn.h.

4.7.4.13 void HalfSpace_Rn::setCoefficient (unsigned int *i*, double *c*) throw std::out_of_range)

Definition at line 46 of file HalfSpace_Rn.cpp.

Here is the call graph for this function:



4.7.4.14 void HalfSpace_Rn::setConstant (double *c*) [inline]

Definition at line 57 of file HalfSpace_Rn.h.

4.7.4.15 const boost::numeric::ublas::vector<double>& HalfSpace_Rn::vect ( ) const [inline]

Definition at line 71 of file HalfSpace_Rn.h.

4.7.5 Member Data Documentation

4.7.5.1 boost::numeric::ublas::vector<double> HalfSpace_Rn::_coefficients [protected]

The normal vector.

Definition at line 112 of file HalfSpace_Rn.h.

4.7.5.2 double HalfSpace_Rn::_constant [protected]

The second member constant.

Definition at line 114 of file HalfSpace_Rn.h.

The documentation for this class was generated from the following files:

- /home/vindelos/CPP/I2M/politopix/trunk/HalfSpace_Rn.h
- /home/vindelos/CPP/I2M/politopix/trunk/HalfSpace_Rn.cpp

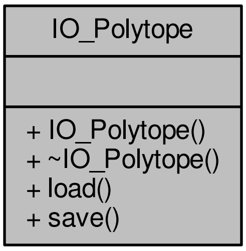
4.8 IO_Polytope Class Reference

Read/write polytopes.
The way we store polytopes :
1st line : comments = "# Dimension NumberOfHalfspaces NumberOfGenerators"
2nd line : cartesian_space_dimension number_of_facets number_of_generators
3rd line : comments = "# HALFSPACES : a0 + a1.x1 + ... + an.xn >= 0."
4th line : a00 a10 ... an0
k-th line : a0k a1k ... ank
(k+1)-th line : comments = "# GENERATORS : V = (v1, ..., vn)"
(k+2)-th line : v11 ... v1n

l-th line : vl1 ... vln
(l+1)-th line : comments = "# FACETS PER GENERATOR : {Fi1, Fi2, ...}"
(l+2)-th line the neigh : Fr ... Fs
m-th line : Fs ... Ft
If (number_of_vertices == 0) then compute the vertices from the
facets including the polytope into a huge cube containing it.
In this case the blocks "GENERATORS" and "FACETS PER GENERATOR" are ignored.

```
#include <IO_Polytope.h>
```

Collaboration diagram for IO_Polytope:



Public Member Functions

- IO_Polytope ()
- ∼IO_Polytope ()

Static Public Member Functions

- static polito_EXPORT void load (const std::string &filename, boost::shared_ptr< PolyhedralCone_Rn > P↩ OLY) throw (std::ios_base::failure,std::out_of_range)

  Load the main data format to store polytopes.
- static polito_EXPORT void save (const std::string &filename, boost::shared_ptr< PolyhedralCone_Rn > P↩ OLY) throw (std::ios_base::failure)

  Save the polytope to the main data format.

4.8.1 Detailed Description

Read/write polytopes.
The way we store polytopes :
1st line : comments = "# Dimension NumberOfHalfspaces NumberOfGenerators"
2nd line : cartesian_space_dimension number_of_facets number_of_generators
3rd line : comments = "# HALFSPACES : a0 + a1.x1 + ... + an.xn >= 0."
4th line : a00 a10 ... an0
k-th line : a0k a1k ... ank
(k+1)-th line : comments = "# GENERATORS : V = (v1, ..., vn)"
(k+2)-th line : v11 ... v1n
l-th line : vl1 ... vln
(l+1)-th line : comments = "# FACETS PER GENERATOR : {Fi1, Fi2, ...}"
(l+2)-th line the neigh : Fr ... Fs

m-th line : Fs ... Ft
If (number_of_vertices == 0) then compute the vertices from the
facets including the polytope into a huge cube containing it.
In this case the blocks "GENERATORS" and "FACETS PER GENERATOR" are ignored.

Definition at line 50 of file IO_Polytope.h.

4.8.2 Constructor & Destructor Documentation

4.8.2.1 IO_Polytope::IO_Polytope ( ) [inline]

Definition at line 53 of file IO_Polytope.h.

4.8.2.2 IO_Polytope::∼IO_Polytope ( ) [inline]

Definition at line 55 of file IO_Polytope.h.

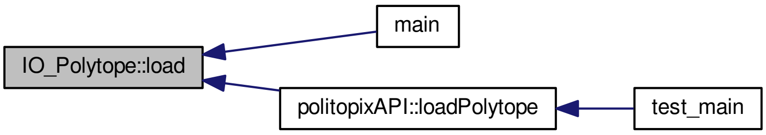
4.8.3 Member Function Documentation

4.8.3.1 void IO_Polytope::load (const std::string & *filename*, boost::shared_ptr< PolyhedralCone_Rn > *POLY*) throw std::ios_base::failure, std::out_of_range) [static]

Load the main data format to store polytopes.

Definition at line 26 of file IO_Polytope.cpp.

Here is the caller graph for this function:

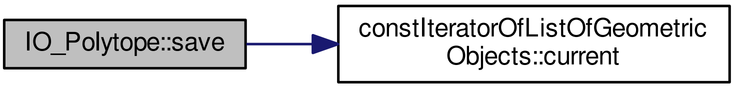


4.8.3.2 void IO_Polytope::save (const std::string & *filename*, boost::shared_ptr< PolyhedralCone_Rn > *POLY*) throw std::ios_base::failure) [static]

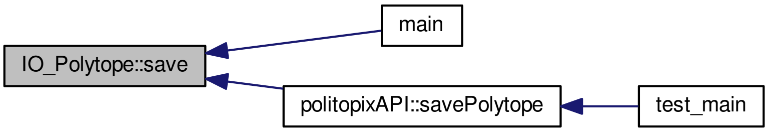
Save the polytope to the main data format.

Definition at line 133 of file IO_Polytope.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



The documentation for this class was generated from the following files:

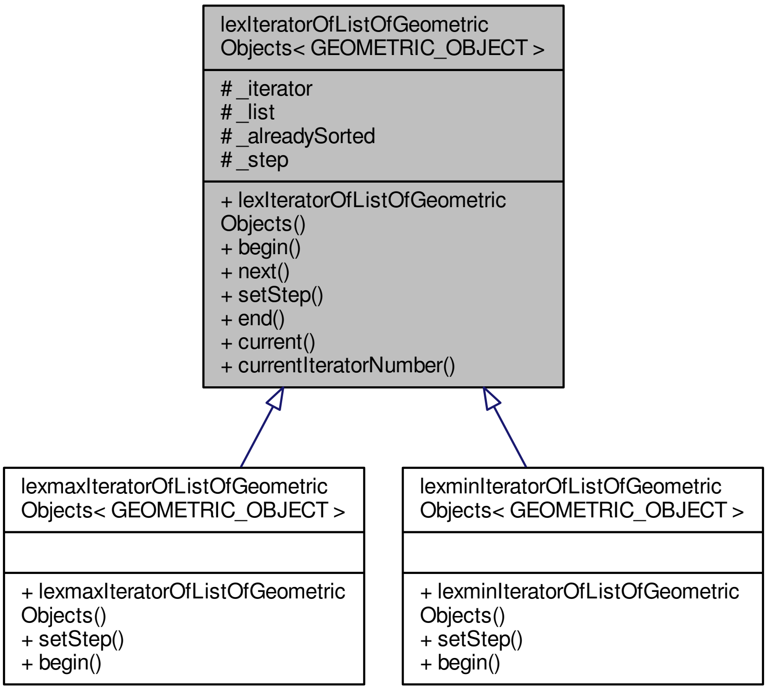
- /home/vindelos/CPP/I2M/politopix/trunk/IO_Polytope.h
- /home/vindelos/CPP/I2M/politopix/trunk/IO_Polytope.cpp

4.9 lexIteratorOfListOfGeometricObjects< GEOMETRIC_OBJECT > Class Template Reference

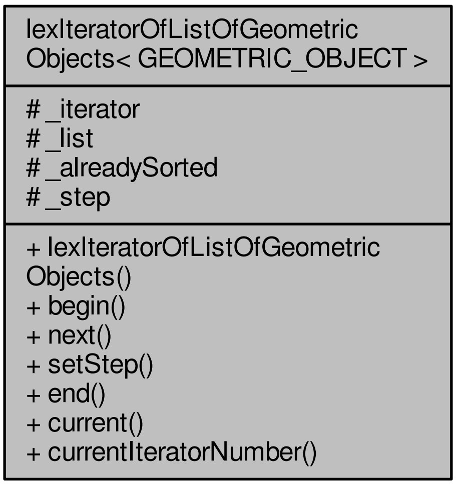
Insert the half-spaces in the list in a lexicographically order, whether min or max.

```
#include <GeometricObjectIterator_Rn.h>
```

Inheritance diagram for lexIteratorOfListOfGeometricObjects< GEOMETRIC_OBJECT >:


lexIteratorOfListOfGeometric Objects< GEOMETRIC_OBJECT >
# _iterator
# _list
# _alreadySorted
# _step
+ lexIteratorOfListOfGeometric Objects()
+ begin()
+ next()
+ setStep()
+ end()
+ current()
+ currentIteratorNumber()
lexmaxIteratorOfListOfGeometric Objects< GEOMETRIC_OBJECT >
+ lexmaxIteratorOfListOfGeometric Objects()
+ setStep()
+ begin()
lexminIteratorOfListOfGeometric Objects< GEOMETRIC_OBJECT >
+ lexminIteratorOfListOfGeometric Objects()
+ setStep()
+ begin()

Collaboration diagram for lexIteratorOfListOfGeometricObjects< GEOMETRIC_OBJECT >:


lexIteratorOfListOfGeometric Objects< GEOMETRIC_OBJECT >
# _iterator
# _list
# _alreadySorted
# _step
+ lexIteratorOfListOfGeometric Objects()
+ begin()
+ next()
+ setStep()
+ end()
+ current()
+ currentIteratorNumber()

Public Member Functions

- lexIteratorOfListOfGeometricObjects (listOfGeometricObjects< GEOMETRIC_OBJECT > &l)

  Constructor.
- void begin ()

  Move the iterator at the beginning of the list.
- void next ()

  Move the iterator one step forward.
- void setStep (unsigned int n)

  Step forward in the list geometric elements.
- bool end () const

  Tell whether we have reached the end of the list.
- const GEOMETRIC_OBJECT current ()

  Return the current geometric element.
- int currentIteratorNumber () const

  Return the current position in the list.

Protected Attributes

- unsigned int _iterator

  The current position in the list.
- listOfGeometricObjects
  < GEOMETRIC_OBJECT > & _list

  The actual list of geometric elements.
- bool _alreadySorted

  Not to do the same job twice.
- unsigned int _step

  Sort after a step.

4.9.1 Detailed Description

template<class GEOMETRIC_OBJECT>class lexIteratorOfListOfGeometricObjects< GEOMETRIC_OBJECT >

Insert the half-spaces in the list in a lexicographically order, whether min or max.

Definition at line 195 of file GeometricObjectIterator_Rn.h.

4.9.2 Constructor & Destructor Documentation

4.9.2.1 template<class GEOMETRIC_OBJECT > lexIteratorOfListOfGeometricObjects< GEOMETRIC_OBJECT >::lexIteratorOfListOfGeometricObjects (listOfGeometricObjects< GEOMETRIC_OBJECT > & *l*) `[inline]`

Constructor.

Definition at line 199 of file GeometricObjectIterator_Rn.h.

4.9.3 Member Function Documentation

4.9.3.1 template<class GEOMETRIC_OBJECT > void lexIteratorOfListOfGeometricObjects< GEOMETRIC_OBJECT >::begin ( ) `[inline]`

Move the iterator at the beginning of the list.

Definition at line 204 of file GeometricObjectIterator_Rn.h.

4.9.3.2 template<class GEOMETRIC_OBJECT > const GEOMETRIC_OBJECT lexIteratorOfListOfGeometricObjects< GEOMETRIC_OBJECT >::current ( ) [inline]

Return the current geometric element.

Definition at line 216 of file GeometricObjectIterator_Rn.h.

4.9.3.3 template<class GEOMETRIC_OBJECT > int lexIteratorOfListOfGeometricObjects< GEOMETRIC_OBJECT >::currentIteratorNumber ( ) const [inline]

Return the current position in the list.

Definition at line 224 of file GeometricObjectIterator_Rn.h.

4.9.3.4 template<class GEOMETRIC_OBJECT > bool lexIteratorOfListOfGeometricObjects< GEOMETRIC_OBJECT >::end ( ) const [inline]

Tell whether we have reached the end of the list.

Definition at line 213 of file GeometricObjectIterator_Rn.h.

4.9.3.5 template<class GEOMETRIC_OBJECT > void lexIteratorOfListOfGeometricObjects< GEOMETRIC_OBJECT >::next ( ) [inline]

Move the iterator one step forward.

Definition at line 207 of file GeometricObjectIterator_Rn.h.

4.9.3.6 template<class GEOMETRIC_OBJECT > void lexIteratorOfListOfGeometricObjects< GEOMETRIC_OBJECT >::setStep (unsigned int *n*) [inline]

Step forward in the list geometric elements.

Definition at line 210 of file GeometricObjectIterator_Rn.h.

4.9.4 Member Data Documentation

4.9.4.1 template<class GEOMETRIC_OBJECT > bool lexIteratorOfListOfGeometricObjects< GEOMETRIC_OBJECT >::_alreadySorted [protected]

Not to do the same job twice.

Definition at line 232 of file GeometricObjectIterator_Rn.h.

4.9.4.2 template<class GEOMETRIC_OBJECT > unsigned int lexIteratorOfListOfGeometricObjects< GEOMETRIC_OBJECT >::_iterator [protected]

The current position in the list.

Definition at line 228 of file GeometricObjectIterator_Rn.h.

4.9.4.3 template<class GEOMETRIC_OBJECT > listOfGeometricObjects<GEOMETRIC_OBJECT>& lexIteratorOfListOfGeometricObjects< GEOMETRIC_OBJECT >::_list [protected]

The actual list of geometric elements.

Definition at line 230 of file GeometricObjectIterator_Rn.h.

4.9.4.4 template< class GEOMETRIC_OBJECT > unsigned int lexIteratorOfListOfGeometricObjects< GEOMETRIC_OBJECT >::_step [protected]

Sort after a step.

Definition at line 234 of file GeometricObjectIterator_Rn.h.

The documentation for this class was generated from the following file:

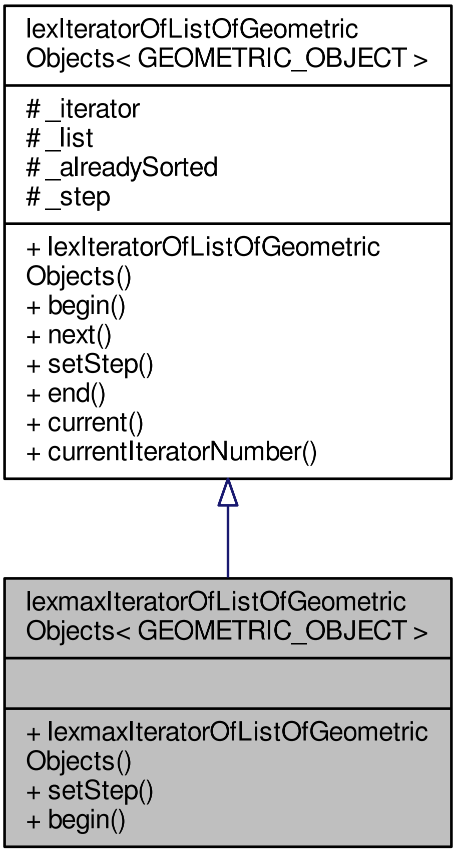
- /home/vindelos/CPP/I2M/politopix/trunk/GeometricObjectIterator_Rn.h

4.10 lexmaxIteratorOfListOfGeometricObjects< GEOMETRIC_OBJECT > Class Template Reference

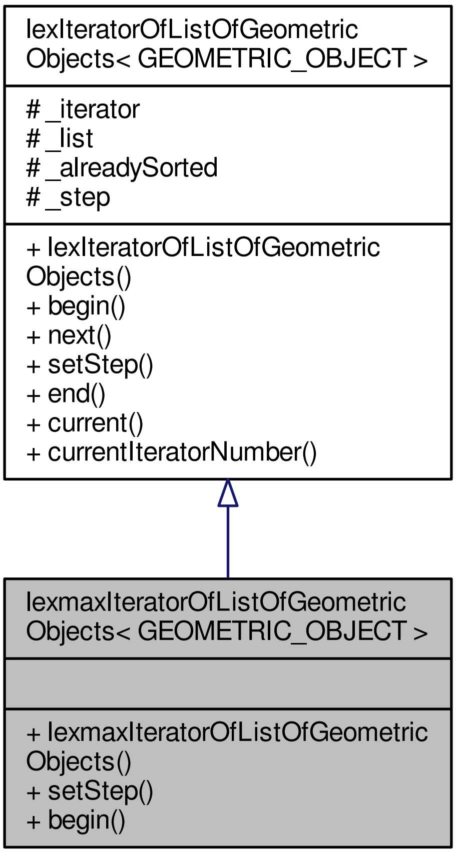
Insert the half-spaces in the list in lexicographically decreasing order.

`#include <GeometricObjectIterator_Rn.h>`

Inheritance diagram for lexmaxIteratorOfListOfGeometricObjects< GEOMETRIC_OBJECT >:


lexIteratorOfListOfGeometric Objects< GEOMETRIC_OBJECT >
# _iterator
# _list
# _alreadySorted
# _step
+ lexIteratorOfListOfGeometric Objects()
+ begin()
+ next()
+ setStep()
+ end()
+ current()
+ currentIteratorNumber()
lexmaxIteratorOfListOfGeometric Objects< GEOMETRIC_OBJECT >
+ lexmaxIteratorOfListOfGeometric Objects()
+ setStep()
+ begin()

Collaboration diagram for lexmaxIteratorOfListOfGeometricObjects< GEOMETRIC_OBJECT >:


lexIteratorOfListOfGeometric
Objects< GEOMETRIC_OBJECT >
# _iterator
# _list
# _alreadySorted
# _step
+ lexIteratorOfListOfGeometric
Objects()
+ begin()
+ next()
+ setStep()
+ end()
+ current()
+ currentIteratorNumber()
lexmaxIteratorOfListOfGeometric
Objects< GEOMETRIC_OBJECT >
+ lexmaxIteratorOfListOfGeometric
Objects()
+ setStep()
+ begin()

Public Member Functions

- lexmaxIteratorOfListOfGeometricObjects (listOfGeometricObjects< GEOMETRIC_OBJECT > &l)

  Constructor.
- void setStep (unsigned int n)

  Move the iterator at the beginning of the list.
- void begin ()

  Move the iterator at the beginning of the list.

Additional Inherited Members

4.10.1 Detailed Description

template<class GEOMETRIC_OBJECT>class lexmaxIteratorOfListOfGeometricObjects< GEOMETRIC_OBJECT >

Insert the half-spaces in the list in lexicographically decreasing order.

Definition at line 264 of file GeometricObjectIterator_Rn.h.

4.10.2 Constructor & Destructor Documentation

4.10.2.1 template<class GEOMETRIC_OBJECT > **lexmaxIteratorOfListOfGeometricObjects**< GEOMETRIC_OBJECT >::**lexmaxIteratorOfListOfGeometricObjects** (listOfGeometricObjects< GEOMETRIC_OBJECT > & *l*) `[inline]`

Constructor.

Definition at line 269 of file GeometricObjectIterator_Rn.h.

4.10.3 Member Function Documentation

4.10.3.1 template<class GEOMETRIC_OBJECT > void **lexmaxIteratorOfListOfGeometricObjects**< GEOMETRIC_OBJECT >::begin ( ) `[inline]`

Move the iterator at the beginning of the list.

Definition at line 284 of file GeometricObjectIterator_Rn.h.

4.10.3.2 template<class GEOMETRIC_OBJECT > void **lexmaxIteratorOfListOfGeometricObjects**< GEOMETRIC_OBJECT >::setStep (unsigned int *n*) `[inline]`

Move the iterator at the beginning of the list.

Definition at line 273 of file GeometricObjectIterator_Rn.h.

The documentation for this class was generated from the following file:

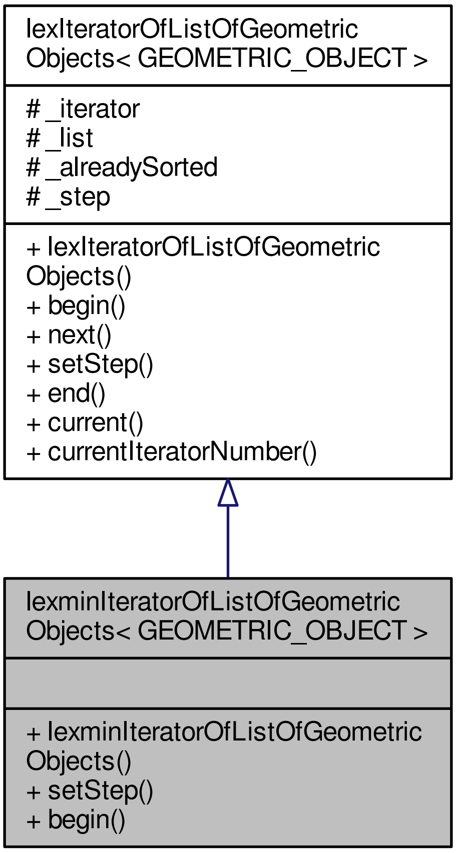
- /home/vindelos/CPP/I2M/politopix/trunk/GeometricObjectIterator_Rn.h

4.11 lexminIteratorOfListOfGeometricObjects< GEOMETRIC_OBJECT > Class Template Reference

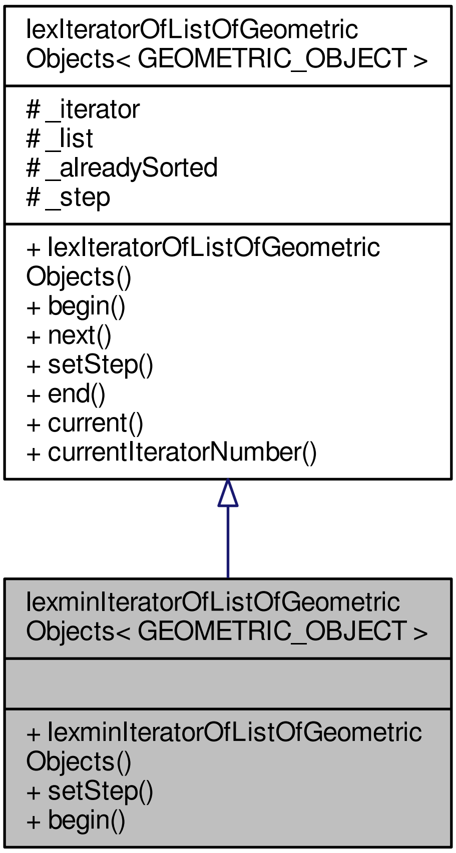
Insert the half-spaces in the list in lexicographically increasing order.

`#include <GeometricObjectIterator_Rn.h>`

Inheritance diagram for lexminIteratorOfListOfGeometricObjects< GEOMETRIC_OBJECT >:


lexIteratorOfListOfGeometric
Objects< GEOMETRIC_OBJECT >
# _iterator
# _list
# _alreadySorted
# _step
+ lexIteratorOfListOfGeometric
Objects()
+ begin()
+ next()
+ setStep()
+ end()
+ current()
+ currentIteratorNumber()
lexminIteratorOfListOfGeometric
Objects< GEOMETRIC_OBJECT >
+ lexminIteratorOfListOfGeometric
Objects()
+ setStep()
+ begin()

Collaboration diagram for lexminIteratorOfListOfGeometricObjects< GEOMETRIC_OBJECT >:


lexIteratorOfListOfGeometric Objects< GEOMETRIC_OBJECT >
# _iterator
# _list
# _alreadySorted
# _step
+ lexIteratorOfListOfGeometric Objects()
+ begin()
+ next()
+ setStep()
+ end()
+ current()
+ currentIteratorNumber()
lexminIteratorOfListOfGeometric Objects< GEOMETRIC_OBJECT >
+ lexminIteratorOfListOfGeometric Objects()
+ setStep()
+ begin()

Public Member Functions

- lexminIteratorOfListOfGeometricObjects (listOfGeometricObjects< GEOMETRIC_OBJECT > &l)

    Constructor.
- void setStep (unsigned int n)

    Step forward in the list geometric elements.
- void begin ()

    Move the iterator at the beginning of the list.

Additional Inherited Members

4.11.1 Detailed Description

template<class GEOMETRIC_OBJECT>class lexminIteratorOfListOfGeometricObjects< GEOMETRIC_OBJECT >

Insert the half-spaces in the list in lexicographically increasing order.

Definition at line 239 of file GeometricObjectIterator_Rn.h.

4.11.2 Constructor & Destructor Documentation

4.11.2.1 template<class GEOMETRIC_OBJECT > **lexminIteratorOfListOfGeometricObjects**< GEOMETRIC_OBJECT >::**lexminIteratorOfListOfGeometricObjects** (**listOfGeometricObjects**< GEOMETRIC_OBJECT > & *l*) [inline]

Constructor.

Definition at line 244 of file GeometricObjectIterator_Rn.h.

4.11.3 Member Function Documentation

4.11.3.1 template<class GEOMETRIC_OBJECT > void **lexminIteratorOfListOfGeometricObjects**< GEOMETRIC_OBJECT >::begin ( ) [inline]

Move the iterator at the beginning of the list.

Definition at line 259 of file GeometricObjectIterator_Rn.h.

4.11.3.2 template<class GEOMETRIC_OBJECT > void **lexminIteratorOfListOfGeometricObjects**< GEOMETRIC_OBJECT >::setStep (unsigned int *n*) [inline]

Step forward in the list geometric elements.

Definition at line 248 of file GeometricObjectIterator_Rn.h.

The documentation for this class was generated from the following file:

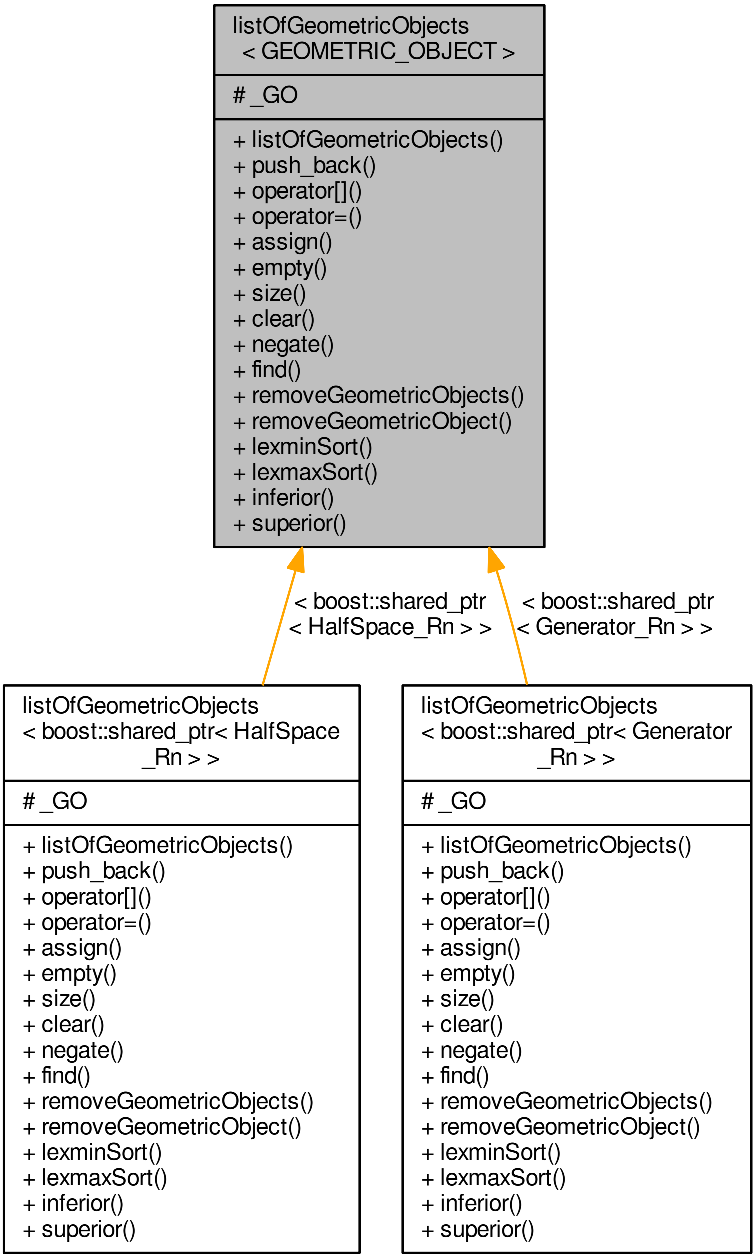
- /home/vindelos/CPP/I2M/politopix/trunk/GeometricObjectIterator_Rn.h

4.12 listOfGeometricObjects< GEOMETRIC_OBJECT > Class Template Reference

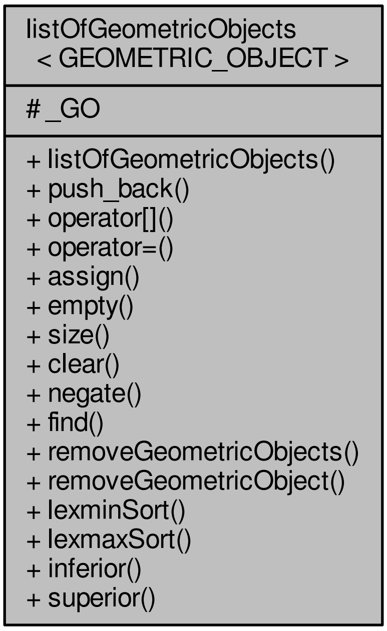
This class is designed to contain the list of all generators or half-spaces representing a polytope or a polyhedral cone.

`#include <GeometricObjectIterator_Rn.h>`

Inheritance diagram for listOfGeometricObjects< GEOMETRIC_OBJECT >:


listOfGeometricObjects < GEOMETRIC_OBJECT >
# _GO
+ listOfGeometricObjects()
+ push_back()
+ operator[]()
+ operator=()
+ assign()
+ empty()
+ size()
+ clear()
+ negate()
+ find()
+ removeGeometricObjects()
+ removeGeometricObject()
+ lexminSort()
+ lexmaxSort()
+ inferior()
+ superior()
< boost::shared_ptr < HalfSpace_Rn > >
< boost::shared_ptr < Generator_Rn > >
listOfGeometricObjects < boost::shared_ptr< HalfSpace _Rn > >
# _GO
+ listOfGeometricObjects()
+ push_back()
+ operator[]()
+ operator=()
+ assign()
+ empty()
+ size()
+ clear()
+ negate()
+ find()
+ removeGeometricObjects()
+ removeGeometricObject()
+ lexminSort()
+ lexmaxSort()
+ inferior()
+ superior()
listOfGeometricObjects < boost::shared_ptr< Generator _Rn > >
# _GO
+ listOfGeometricObjects()
+ push_back()
+ operator[]()
+ operator=()
+ assign()
+ empty()
+ size()
+ clear()
+ negate()
+ find()
+ removeGeometricObjects()
+ removeGeometricObject()
+ lexminSort()
+ lexmaxSort()
+ inferior()
+ superior()

Collaboration diagram for listOfGeometricObjects< GEOMETRIC_OBJECT >:



Public Member Functions

- listOfGeometricObjects ()

  Constructor.
- void push_back (const GEOMETRIC_OBJECT &gn)

  Include a new half space in the list.
- const GEOMETRIC_OBJECT & operator[] (unsigned int i) const

  Return the i-th generator.
- void operator= (const listOfGeometricObjects< GEOMETRIC_OBJECT > &listOfGN)

  Copies all elements from listOfGN to _GN.
- void assign (const listOfGeometricObjects< GEOMETRIC_OBJECT > &listOfGN)

  Copies all elements from listOfGN to _GN.
- bool empty () const

  Check whether the set is empty or not.
- unsigned int size () const

  Get the total number of genuine facets.
- void clear ()

  Clear the whole list.
- void negate ()

  Multiply all generators or half-spaces by -1.
- unsigned int find (const GEOMETRIC_OBJECT &GO) const throw (std::out_of_range)

  Find a given object in list..
- void removeGeometricObjects (const std::set< GEOMETRIC_OBJECT > &setToRemove)

  Get rid of all the objects stored in the set.

- void removeGeometricObject (unsigned int j)

  Remove the geometric object number j from the list.
- void lexminSort (unsigned int step)
- void lexmaxSort (unsigned int step)

Static Public Member Functions

- static bool inferior (const GEOMETRIC_OBJECT &HS1, const GEOMETRIC_OBJECT &HS2)

  Tell whether a given object is declared inferior to another one.
- static bool superior (const GEOMETRIC_OBJECT &HS1, const GEOMETRIC_OBJECT &HS2)

  The opposite of the function inferior(HS1, HS2)

Protected Attributes

- std::vector< GEOMETRIC_OBJECT > _GO

  The full list of half spaces or generators for example.

4.12.1 Detailed Description

template<class GEOMETRIC_OBJECT>class listOfGeometricObjects< GEOMETRIC_OBJECT >

This class is designed to contain the list of all generators or half-spaces representing a polytope or a polyhedral cone.

Definition at line 40 of file GeometricObjectIterator_Rn.h.

4.12.2 Constructor & Destructor Documentation

4.12.2.1 template<class GEOMETRIC_OBJECT> listOfGeometricObjects< GEOMETRIC_OBJECT >::listOfGeometricObjects ( ) [inline]

Constructor.

Definition at line 44 of file GeometricObjectIterator_Rn.h.

4.12.3 Member Function Documentation

4.12.3.1 template<class GEOMETRIC_OBJECT> void listOfGeometricObjects< GEOMETRIC_OBJECT >::assign (const listOfGeometricObjects< GEOMETRIC_OBJECT > & *listOfGN*) [inline]

Copies all elements from listOfGN to _GN.

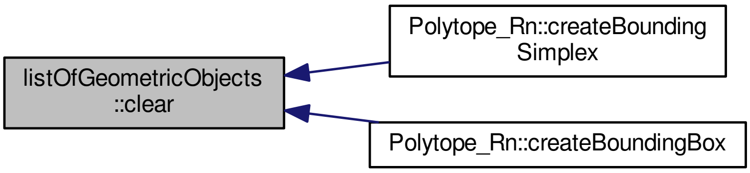
Definition at line 57 of file GeometricObjectIterator_Rn.h.

4.12.3.2 template<class GEOMETRIC_OBJECT> void listOfGeometricObjects< GEOMETRIC_OBJECT >::clear ( ) [inline]

Clear the whole list.

Definition at line 67 of file GeometricObjectIterator_Rn.h.

Here is the caller graph for this function:



4.12.3.3 template<class GEOMETRIC_OBJECT> bool listOfGeometricObjects< GEOMETRIC_OBJECT >::empty ( ) const [inline]

Check whether the set is empty or not.

Definition at line 61 of file GeometricObjectIterator_Rn.h.

4.12.3.4 template<class GEOMETRIC_OBJECT> unsigned int listOfGeometricObjects< GEOMETRIC_OBJECT >::find (const GEOMETRIC_OBJECT & *GO*) const throw std::out_of_range) [inline]

Find a given object in list..

Definition at line 76 of file GeometricObjectIterator_Rn.h.

4.12.3.5 template<class GEOMETRIC_OBJECT> static bool listOfGeometricObjects< GEOMETRIC_OBJECT >::inferior (const GEOMETRIC_OBJECT & *HS1*, const GEOMETRIC_OBJECT & *HS2*) [inline], [static]

Tell whether a given object is declared inferior to another one.

Definition at line 104 of file GeometricObjectIterator_Rn.h.

4.12.3.6 template<class GEOMETRIC_OBJECT> void listOfGeometricObjects< GEOMETRIC_OBJECT >::lexmaxSort (unsigned int *step*) [inline]

Definition at line 129 of file GeometricObjectIterator_Rn.h.

4.12.3.7 template<class GEOMETRIC_OBJECT> void listOfGeometricObjects< GEOMETRIC_OBJECT >::lexminSort (unsigned int *step*) [inline]

Definition at line 123 of file GeometricObjectIterator_Rn.h.

4.12.3.8 template<class GEOMETRIC_OBJECT> void listOfGeometricObjects< GEOMETRIC_OBJECT >::negate ( ) [inline]

Multiply all generators or half-spaces by -1.

Definition at line 70 of file GeometricObjectIterator_Rn.h.

4.12.3.9 template<class GEOMETRIC_OBJECT> void **listOfGeometricObjects**< GEOMETRIC_OBJECT >::operator= (const **listOfGeometricObjects**< GEOMETRIC_OBJECT > & *listOfGN*) `[inline]`

Copies all elements from listOfGN to _GN.

Definition at line 53 of file GeometricObjectIterator_Rn.h.

4.12.3.10 template<class GEOMETRIC_OBJECT> const GEOMETRIC_OBJECT& **listOfGeometricObjects**< GEOMETRIC_OBJECT >::operator[] (unsigned int *i*) const `[inline]`

Return the i-th generator.

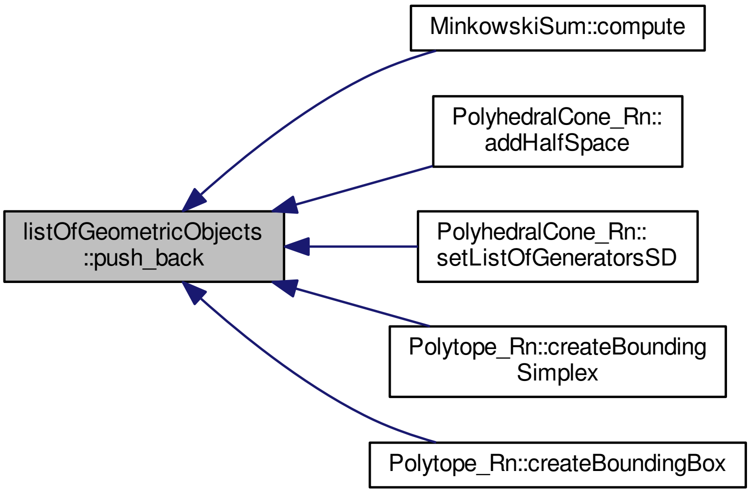
Definition at line 50 of file GeometricObjectIterator_Rn.h.

4.12.3.11 template<class GEOMETRIC_OBJECT> void **listOfGeometricObjects**< GEOMETRIC_OBJECT >::push_back (const GEOMETRIC_OBJECT & *gn*) `[inline]`

Include a new half space in the list.

Definition at line 47 of file GeometricObjectIterator_Rn.h.

Here is the caller graph for this function:


MinkowskiSum::compute
PolyhedralCone_Rn::addHalfSpace
listOfGeometricObjects::push_back
PolyhedralCone_Rn::setListOfGeneratorsSD
Polytope_Rn::createBoundingSimplex
Polytope_Rn::createBoundingBox

4.12.3.12 template<class GEOMETRIC_OBJECT> void **listOfGeometricObjects**< GEOMETRIC_OBJECT >::removeGeometricObject (unsigned int *j*) `[inline]`

Remove the geometric object number *j* from the list.

Definition at line 101 of file GeometricObjectIterator_Rn.h.

4.12.3.13 template<class GEOMETRIC_OBJECT> void **listOfGeometricObjects**< GEOMETRIC_OBJECT >::removeGeometricObjects (const std::set< GEOMETRIC_OBJECT > & *setToRemove*) `[inline]`

Get rid of all the objects stored in the set.

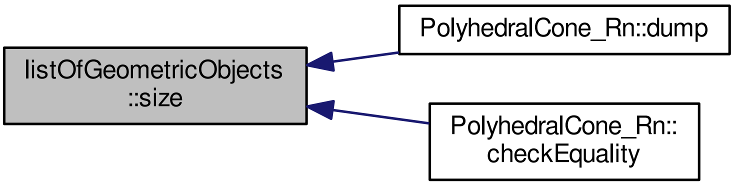
Definition at line 91 of file GeometricObjectIterator_Rn.h.

4.12.3.14 template<class GEOMETRIC_OBJECT> unsigned int listOfGeometricObjects< GEOMETRIC_OBJECT >::size ( ) const [inline]

Get the total number of genuine facets.

Definition at line 64 of file GeometricObjectIterator_Rn.h.

Here is the caller graph for this function:



4.12.3.15 template<class GEOMETRIC_OBJECT> static bool listOfGeometricObjects< GEOMETRIC_OBJECT >::superior (const GEOMETRIC_OBJECT & *HS1*, const GEOMETRIC_OBJECT & *HS2*) [inline], [static]

The opposite of the function inferior(HS1, HS2)

Definition at line 119 of file GeometricObjectIterator_Rn.h.

4.12.4 Member Data Documentation

4.12.4.1 template<class GEOMETRIC_OBJECT> std::vector< GEOMETRIC_OBJECT > listOfGeometricObjects< GEOMETRIC_OBJECT >::_GO [protected]

The full list of half spaces or generators for example.

Definition at line 137 of file GeometricObjectIterator_Rn.h.

The documentation for this class was generated from the following file:

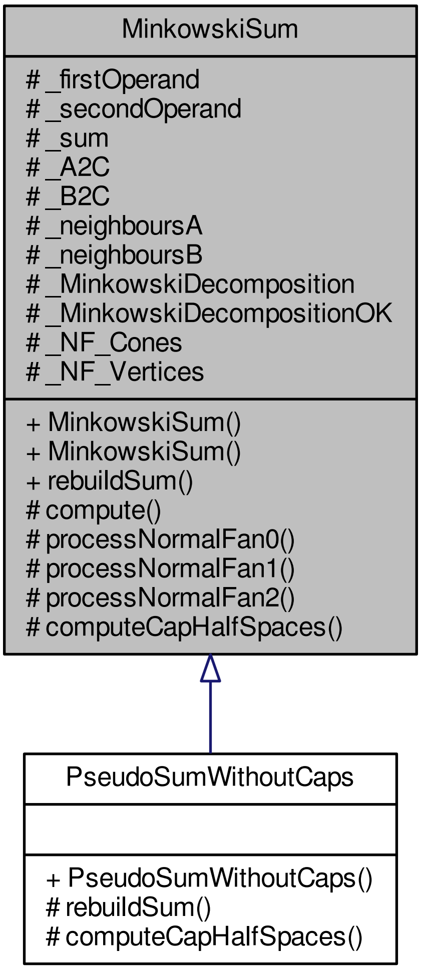
- /home/vindelos/CPP/I2M/politopix/trunk/GeometricObjectIterator_Rn.h

4.13 MinkowskiSum Class Reference

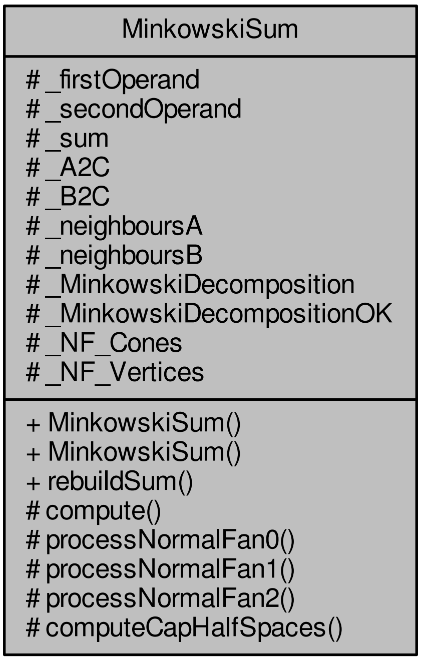
Compute the Minkowski sum of two polytopes.

`#include <PolyhedralAlgorithms_Rn.h>`

Inheritance diagram for MinkowskiSum:



Collaboration diagram for MinkowskiSum:



Public Member Functions

- MinkowskiSum (const boost::shared_ptr< Polytope_Rn > &A, const boost::shared_ptr< Polytope_Rn > &B, boost::shared_ptr< Polytope_Rn > &C)

    Compute the Minkowski sum of two polytopes.
- MinkowskiSum (const boost::shared_ptr< Polytope_Rn > &A, const boost::shared_ptr< Polytope_Rn > &B, boost::shared_ptr< Polytope_Rn > &C, const std::vector< std::vector< int > > &genitorsOfGeneratorsA, const std::vector< std::vector< int > > &genitorsOfGeneratorsB, std::vector< std::vector< int > > &traceGenerators)

    Compute the Minkowski sum of two polytopes.
- boost::shared_ptr< Polytope_Rn > rebuildSum (const std::set< unsigned int > &firstOperandCaps, const std::set< unsigned int > &secondOperandCaps, std::set< unsigned int > &newCaps, double bb_size=1000.)

    Remove the cap half-spaces stored in sets and then truncate again.

Protected Member Functions

- boost::shared_ptr< Polytope_Rn > compute () throw (std::domain_error)
- void processNormalFan0 ()

    Do the final job after having intersected all dual cones. The reduction process simply compares all dual cones generators.
- void processNormalFan1 ()

    Do the final job after having intersected all dual cones. The reduction process uses neighbourhood properties to identify dual cones generators.
- void processNormalFan2 ()

Do the final job after having intersected all dual cones. The reduction process builds half-spaces and identifies them with they generators lists.

- void computeCapHalfSpaces (const std::set< unsigned int > &firstOperandCaps, const std::set< unsigned int > &secondOperandCaps, std::set< unsigned int > &sumCaps) const throw (std::domain_error)

    Return the cap half-spaces of the sum in function of the two operands cap half-spaces.

Protected Attributes

- const boost::shared_ptr
  < Polytope_Rn > _firstOperand
- const boost::shared_ptr
  < Polytope_Rn > _secondOperand
- boost::shared_ptr< Polytope_Rn > _sum
- std::vector< std::vector
  < unsigned int > > _A2C

    Store the polyhedrical cap in C of each vertex of A.
- std::vector< std::vector
  < unsigned int > > _B2C

    Store the polyhedrical cap in C of each vertex of B.
- std::vector< std::vector
  < unsigned int > > _neighboursA

    Store the neighbours of each vertex of A.
- std::vector< std::vector
  < unsigned int > > _neighboursB

    Store the neighbours of each vertex of B.
- std::vector< std::pair
  < unsigned int, unsigned int > > _MinkowskiDecomposition

    Store the genitors in A and B of each vertex of C.
- std::vector< bool > _MinkowskiDecompositionOK

    Tell whether _MinkowskiDecomposition had to be considered or not.
- std::vector< boost::shared_ptr
  < PolyhedralCone_Rn > > _NF_Cones

    The normal fan polyhedrical cones list.
- std::vector< boost::shared_ptr
  < Generator_Rn > > _NF_Vertices

    The list of C vertices.

4.13.1 Detailed Description

Compute the Minkowski sum of two polytopes.

Definition at line 140 of file PolyhedralAlgorithms_Rn.h.

4.13.2 Constructor & Destructor Documentation

4.13.2.1 MinkowskiSum::MinkowskiSum (const boost::shared_ptr< Polytope_Rn > & *A*, const boost::shared_ptr< Polytope_Rn > & *B*, boost::shared_ptr< Polytope_Rn > & *C*) [inline]

Compute the Minkowski sum of two polytopes.

Definition at line 145 of file PolyhedralAlgorithms_Rn.h.

4.13.2.2 MinkowskiSum::MinkowskiSum (const boost::shared_ptr< **Polytope_Rn** > & *A*, const boost::shared_ptr< **Polytope_Rn** > & *B*, boost::shared_ptr< **Polytope_Rn** > & *C*, const std::vector< std::vector< int > > & *genitorsOfGeneratorsA*, const std::vector< std::vector< int > > & *genitorsOfGeneratorsB*, std::vector< std::vector< int > > & *traceGenerators*) [inline]

Compute the Minkowski sum of two polytopes.

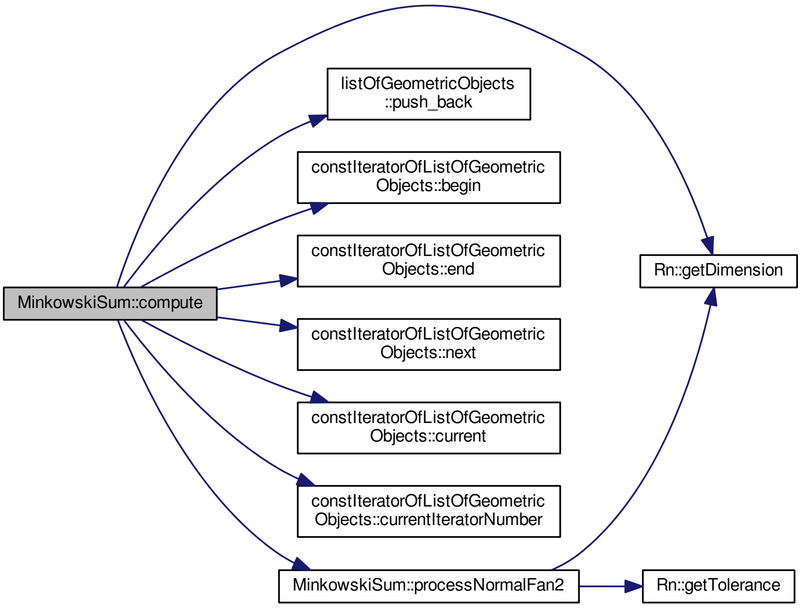
Definition at line 152 of file PolyhedralAlgorithms_Rn.h.

4.13.3 Member Function Documentation

4.13.3.1 boost::shared_ptr< **Polytope_Rn** > MinkowskiSum::compute ( ) throw std::domain_error) [protected]

Definition at line 349 of file PolyhedralAlgorithms_Rn.cpp.

Here is the call graph for this function:


listOfGeometricObjects ::push_back
constIteratorOfListOfGeometric Objects::begin
constIteratorOfListOfGeometric Objects::end
constIteratorOfListOfGeometric Objects::next
constIteratorOfListOfGeometric Objects::current
constIteratorOfListOfGeometric Objects::currentIteratorNumber
MinkowskiSum::processNormalFan2
MinkowskiSum::compute
Rn::getDimension
Rn::getTolerance

4.13.3.2 void MinkowskiSum::computeCapHalfSpaces (const std::set< unsigned int > & *firstOperandCaps*, const std::set< unsigned int > & *secondOperandCaps*, std::set< unsigned int > & *sumCaps*) const throw std::domain_error) [protected]

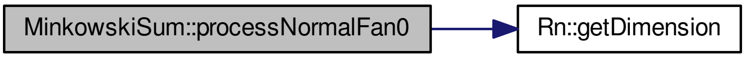
Return the cap half-spaces of the sum in function of the two operands cap half-spaces.

4.13.3.3 void MinkowskiSum::processNormalFan0 ( ) [protected]

Do the final job after having intersected all dual cones. The reduction process simply compares all dual cones generators.

Definition at line 788 of file PolyhedralAlgorithms_Rn.cpp.

Here is the call graph for this function:

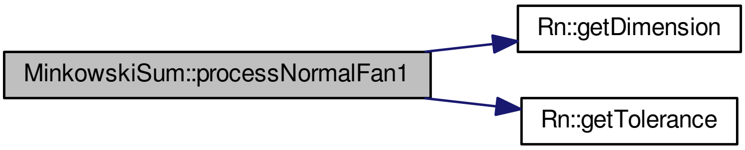


4.13.3.4 void MinkowskiSum::processNormalFan1 ( ) [protected]

Do the final job after having intersected all dual cones. The reduction process uses neighbourhood properties to identify dual cones generators.

Definition at line 661 of file PolyhedralAlgorithms_Rn.cpp.

Here is the call graph for this function:

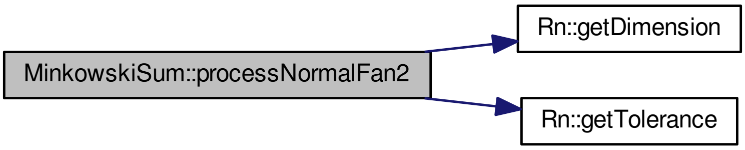


4.13.3.5 void MinkowskiSum::processNormalFan2 ( ) [protected]

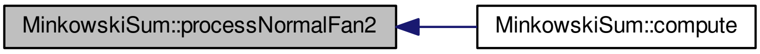
Do the final job after having intersected all dual cones. The reduction process builds half-spaces and identifies them with they generators lists.

Definition at line 523 of file PolyhedralAlgorithms_Rn.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



4.13.3.6 boost::shared_ptr<Polytope_Rn> MinkowskiSum::rebuildSum (const std::set< unsigned int > & *firstOperandCaps*, const std::set< unsigned int > & *secondOperandCaps*, std::set< unsigned int > & *newCaps*, double *bb_size* = 1000.)

Remove the cap half-spaces stored in sets and then truncate again.

Returns

The new sum

4.13.4 Member Data Documentation

4.13.4.1 std::vector< std::vector<unsigned int> > MinkowskiSum::_A2C [protected]

Store the polyhedrical cap in C of each vertex of A.

Definition at line 214 of file PolyhedralAlgorithms_Rn.h.

4.13.4.2 std::vector< std::vector<unsigned int> > MinkowskiSum::_B2C [protected]

Store the polyhedrical cap in C of each vertex of B.

Definition at line 216 of file PolyhedralAlgorithms_Rn.h.

4.13.4.3 const boost::shared_ptr<Polytope_Rn> MinkowskiSum::_firstOperand [protected]

Definition at line 205 of file PolyhedralAlgorithms_Rn.h.

4.13.4.4 std::vector< std::pair< unsigned int, unsigned int > > MinkowskiSum::_MinkowskiDecomposition [protected]

Store the genitors in A and B of each vertex of C.

Definition at line 230 of file PolyhedralAlgorithms_Rn.h.

4.13.4.5 std::vector< bool > MinkowskiSum::_MinkowskiDecompositionOK [protected]

Tell whether _MinkowskiDecomposition had to be considered or not.

Definition at line 232 of file PolyhedralAlgorithms_Rn.h.

4.13.4.6 std::vector< std::vector<unsigned int> > MinkowskiSum::_neighboursA [protected]

Store the neighbours of each vertex of A.

neighboursA(a_i,a_j)==1 <=> a_i R a_j

Definition at line 220 of file PolyhedralAlgorithms_Rn.h.

4.13.4.7 std::vector< std::vector<unsigned int> > MinkowskiSum::_neighboursB [protected]

Store the neighbours of each vertex of B.

neighboursB(b_i,b_j)==1 <=> b_u R b_v

Definition at line 223 of file PolyhedralAlgorithms_Rn.h.

4.13.4.8 std::vector< boost::shared_ptr<**PolyhedralCone_Rn**> > MinkowskiSum::_NF_Cones [protected]

The normal fan polyhedrical cones list.

Definition at line 235 of file PolyhedralAlgorithms_Rn.h.

4.13.4.9 std::vector< boost::shared_ptr<**Generator_Rn**> > MinkowskiSum::_NF_Vertices [protected]

The list of C vertices.

Definition at line 237 of file PolyhedralAlgorithms_Rn.h.

4.13.4.10 const boost::shared_ptr<**Polytope_Rn**> MinkowskiSum::_secondOperand [protected]

Definition at line 206 of file PolyhedralAlgorithms_Rn.h.

4.13.4.11 boost::shared_ptr<**Polytope_Rn**> MinkowskiSum::_sum [protected]

Definition at line 207 of file PolyhedralAlgorithms_Rn.h.

The documentation for this class was generated from the following files:

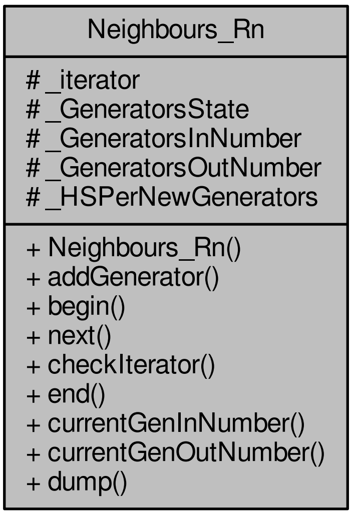
- /home/vindelos/CPP/I2M/politopix/trunk/PolyhedralAlgorithms_Rn.h
- /home/vindelos/CPP/I2M/politopix/trunk/PolyhedralAlgorithms_Rn.cpp

4.14 Neighbours_Rn Class Reference

Class dedicated to degeneration processing when looking for neighbours.
Let *A* be a polytope of $\mathbb{R}^n, A = H_1^+ \cap H_2^+ \cap ... H_{n-1}^+ \cap ... H_k^+$ where *k*>*n*.
Let *[v_i, v_j]* be a segment of two vertices of *A* such as:

```
#include <Neighbours_Rn.h>
```

Collaboration diagram for Neighbours_Rn:



Public Member Functions

- Neighbours_Rn ()
- void addGenerator (const std::vector< unsigned int > &commonFacets, unsigned int numbergenIN, unsigned int numbergenOUT, HalfSpace_Rn::State state)

  Tell whether a pseudo neighbor is a genuine one comparing set of half-spaces.
- void begin ()

  Iterator function.
- void next ()

  Iterator function.
- void checkIterator ()

  Make sure we don't point on a generator with state ON.
- bool end ()

  Iterator function.
- unsigned int currentGenInNumber ()

  Iterator function.
- unsigned int currentGenOutNumber ()

  Iterator function.
- void dump (std::ostream &ofs)

  Display the content on the stream passed as an argument.

Protected Attributes

- unsigned int _iterator

  A runner to iterate through the list of genuine neighbors.
- std::vector< HalfSpace_Rn::State > _GeneratorsState

  The pair of generators state.

- std::vector< unsigned int > _GeneratorsInNumber

  The generator numbers IN in a global list.
- std::vector< unsigned int > _GeneratorsOutNumber

  The generator numbers OUT in a global list.
- std::vector< std::vector
  < unsigned int > > _HSPerNewGenerators

  For each generator, store all raw pointers on their corresponding half-spaces.

4.14.1 Detailed Description

Class dedicated to degeneration processing when looking for neighbours.
Let *A* be a polytope of $\mathbb{R}^n, A = H_1^+ \cap H_2^+ \cap ... H_{n-1}^+ \cap ... H_k^+$ where *k*>*n*.
Let *[v$_i$, v$_j$]* be a segment of two vertices of *A* such as:

- they share (n-1) hyperplanes in common: $H_{ij} = \{H_1, H_2, ... H_{n-1}\}$
- there exists an hyperplane $\mathscr{H}$ separating *[v$_j$, v$_k$]*

We call *[v$_i$, v$_j$]* a pseudo-edge if it respects the first assumption. The question is: to which condition *[v$_i$, v$_j$]* is a genuine edge? Can we answer the question only by processing the pseudo-edges separated by the hyperplane $\mathscr{H}$? The straight line $(v_i, v_j) \subset H_1 \cap H_2 \cap ... H_{n-1}$ with $\mathscr{H} \neq H_u$, $u \in \{1, ..., n-1\}$. So $H_1 \cap H_2 \cap ... H_{n-1} \cap H_n^+ \cap ... H_k^+ =$ *F$_A$* is a face of *A* of dimension at least 1. Let's assume *[v$_i$, v$_j$]* is not an edge of *A*, then it is not an edge of *F$_A$* and we cannot have *F$_A$* included neither in $\mathscr{H}^+$ nor in $\mathscr{H}^-$ as $\mathscr{H}$ separates *[v$_i$, v$_j$]*. So $\mathscr{H}$ separates *F$_A$* and we can find an edge *[v$_a$, v$_b$]* of *F$_A$* such as:

- $\mathscr{H}$ separates *[v$_a$, v$_b$]* or
- $\mathscr{H}$ passes through *v$_a$* or
- $\mathscr{H}$ passes through *v$_b$*

As *[v$_a$, v$_b$]* is an edge of *F$_A$*, *v$_a$* and *v$_b$* share in common the list of hyperplanes *H$_{ab}$* . *H$_{ab}$* contains the (n-1) half-spaces $H_1, H_2, ... H_{n-1}$ and others because the intersection $H_1 \cap H_2 \cap ... H_{n-1}$ does not define a straight line. So if *[v$_i$, v$_j$]* is not an edge of *A*, we can find a genuine edge *[v$_a$, v$_b$]* , intersecting with $\mathscr{H}$, such as $H_{ij} \subset H_{ab}$

Definition at line 59 of file Neighbours_Rn.h.

4.14.2 Constructor & Destructor Documentation

4.14.2.1 Neighbours_Rn::Neighbours_Rn ( ) [inline]

Definition at line 62 of file Neighbours_Rn.h.

4.14.3 Member Function Documentation

4.14.3.1 void Neighbours_Rn::addGenerator (const std::vector< unsigned int > & *commonFacets,* unsigned int *numbergenIN,* unsigned int *numbergenOUT,* HalfSpace_Rn::State *state*) [inline]

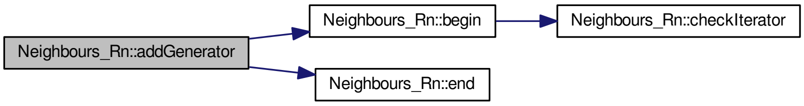
Tell whether a pseudo neighbor is a genuine one comparing set of half-spaces.

Parameters

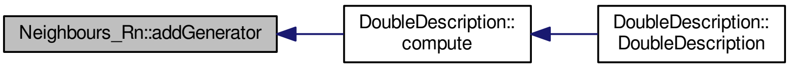
commonFacets	the set of common half-spaces pointers between *this* and *gen*
numbergenIN	the generator number candidate to be a genuine end of edge
numbergenOUT	the generator number candidate to be the other genuine end of edge
state	equal to HalfSpace_Rn::hs_IN_OR_OUT or HalfSpace_Rn::hs_ON according to the edge property

Definition at line 69 of file Neighbours_Rn.h.

Here is the call graph for this function:



Here is the caller graph for this function:

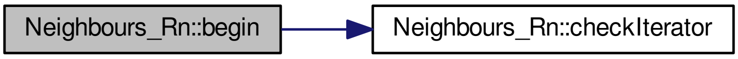

Neighbours_Rn::addGenerator
DoubleDescription:: compute
DoubleDescription:: DoubleDescription

4.14.3.2 void Neighbours_Rn::begin ( ) [inline]

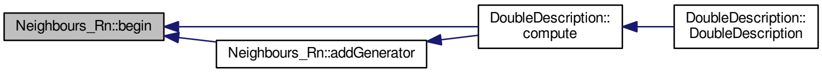
Iterator function.

Definition at line 128 of file Neighbours_Rn.h.

Here is the call graph for this function:



Here is the caller graph for this function:

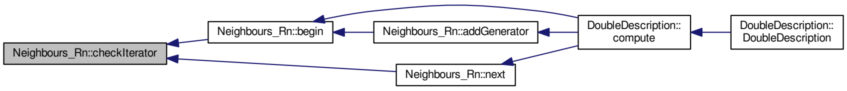

Neighbours_Rn::begin
Neighbours_Rn::addGenerator
DoubleDescription:: compute
DoubleDescription:: DoubleDescription

4.14.3.3 void Neighbours_Rn::checkIterator ( ) [inline]

Make sure we don't point on a generator with state ON.

Definition at line 134 of file Neighbours_Rn.h.

Here is the caller graph for this function:

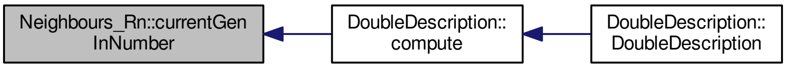

Neighbours_Rn::checkIterator
Neighbours_Rn::begin
Neighbours_Rn::addGenerator
DoubleDescription::compute
DoubleDescription::DoubleDescription
Neighbours_Rn::next

4.14.3.4 unsigned int Neighbours_Rn::currentGenInNumber ( ) [inline]

Iterator function.

Definition at line 143 of file Neighbours_Rn.h.

Here is the caller graph for this function:

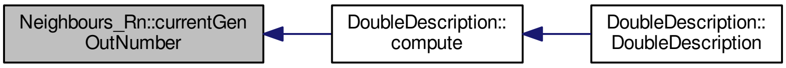

Neighbours_Rn::currentGenInNumber
DoubleDescription::compute
DoubleDescription::DoubleDescription

4.14.3.5 unsigned int Neighbours_Rn::currentGenOutNumber ( ) [inline]

Iterator function.

Definition at line 146 of file Neighbours_Rn.h.

Here is the caller graph for this function:

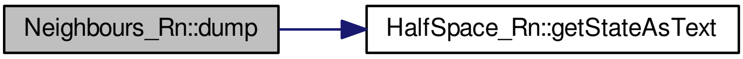

Neighbours_Rn::currentGenOutNumber
DoubleDescription::compute
DoubleDescription::DoubleDescription

4.14.3.6 void Neighbours_Rn::dump (std::ostream & *ofs*) [inline]

Display the content on the stream passed as an argument.

Definition at line 149 of file Neighbours_Rn.h.

Here is the call graph for this function:

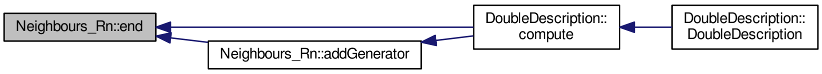


4.14.3.7 bool Neighbours_Rn::end ( ) [inline]

Iterator function.

Definition at line 140 of file Neighbours_Rn.h.

Here is the caller graph for this function:

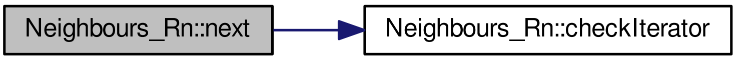


4.14.3.8 void Neighbours_Rn::next ( ) [inline]

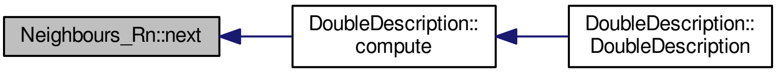
Iterator function.

Definition at line 131 of file Neighbours_Rn.h.

Here is the call graph for this function:



Here is the caller graph for this function:



4.14.4 Member Data Documentation

4.14.4.1 std::vector< unsigned int > Neighbours_Rn::_GeneratorsInNumber [protected]

The generator numbers IN in a global list.

Definition at line 169 of file Neighbours_Rn.h.

4.14.4.2 std::vector< unsigned int > Neighbours_Rn::_GeneratorsOutNumber [protected]

The generator numbers OUT in a global list.

Definition at line 171 of file Neighbours_Rn.h.

4.14.4.3 std::vector< HalfSpace_Rn::State > Neighbours_Rn::_GeneratorsState [protected]

The pair of generators state.

Definition at line 167 of file Neighbours_Rn.h.

4.14.4.4 std::vector< std::vector< unsigned int > > Neighbours_Rn::_HSPerNewGenerators [protected]

For each generator, store all raw pointers on their corresponding half-spaces.

Definition at line 173 of file Neighbours_Rn.h.

4.14.4.5 unsigned int Neighbours_Rn::_iterator [protected]

A runner to iterate through the list of genuine neighbors.

Definition at line 165 of file Neighbours_Rn.h.

The documentation for this class was generated from the following file:

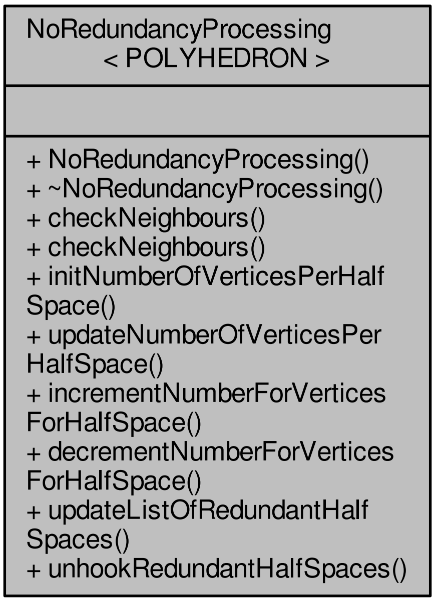
- /home/vindelos/CPP/I2M/politopix/trunk/Neighbours_Rn.h

4.15 NoRedundancyProcessing< POLYHEDRON > Class Template Reference

Makes the assumption we do not need to process redundant half-spaces in a specific way.

```
#include <DoubleDescription_Rn.h>
```

Collaboration diagram for NoRedundancyProcessing< POLYHEDRON >:


NoRedundancyProcessing
< POLYHEDRON >
+ NoRedundancyProcessing()
+ ~NoRedundancyProcessing()
+ checkNeighbours()
+ checkNeighbours()
+ initNumberOfVerticesPerHalf
Space()
+ updateNumberOfVerticesPer
HalfSpace()
+ incrementNumberForVertices
ForHalfSpace()
+ decrementNumberForVertices
ForHalfSpace()
+ updateListOfRedundantHalf
Spaces()
+ unhookRedundantHalfSpaces()

Public Member Functions

- NoRedundancyProcessing ()
- virtual ∼NoRedundancyProcessing ()
- virtual bool checkNeighbours (POLYHEDRON poly, const boost::shared_ptr< Generator_Rn > &genIn, const boost::shared_ptr< Generator_Rn > &genOut, std::vector< boost::shared_ptr< HalfSpace_Rn > > &commonFacets)

    Check whether two generators are neighbors in the context of not taking into account redundancy.
- virtual bool checkNeighbours (POLYHEDRON poly, const boost::shared_ptr< Generator_Rn > &genIn, const boost::shared_ptr< Generator_Rn > &genOut, std::vector< HalfSpace_Rn ∗ > &commonFacets)

    Check whether two generators are neighbors in the context of not taking into account redundancy.
- void initNumberOfVerticesPerHalfSpace (const std::vector< boost::shared_ptr< Generator_Rn > > &)

    Only useful in the case of dealing and processing redundancy.
- void updateNumberOfVerticesPerHalfSpace (const boost::shared_ptr< HalfSpace_Rn > &, unsigned int)

    Only useful in the case of dealing and processing redundancy.
- void incrementNumberForVerticesForHalfSpace (const boost::shared_ptr< Generator_Rn > &)

    Only useful in the case of dealing and processing redundancy.
- void decrementNumberForVerticesForHalfSpace (const boost::shared_ptr< Generator_Rn > &)

    Only useful in the case of dealing and processing redundancy.
- void updateListOfRedundantHalfSpaces (unsigned int)

    Only useful in the case of dealing and processing redundancy.
- void unhookRedundantHalfSpaces (POLYHEDRON)

4.15.1 Detailed Description

template<class POLYHEDRON>class NoRedundancyProcessing< POLYHEDRON >

Makes the assumption we do not need to process redundant half-spaces in a specific way.

Definition at line 467 of file DoubleDescription_Rn.h.

4.15.2 Constructor & Destructor Documentation

4.15.2.1 template<class POLYHEDRON > NoRedundancyProcessing< POLYHEDRON >::NoRedundancyProcessing ( ) [inline]

Definition at line 469 of file DoubleDescription_Rn.h.

4.15.2.2 template<class POLYHEDRON > virtual NoRedundancyProcessing< POLYHEDRON >::∼NoRedundancyProcessing ( ) [inline], [virtual]

Definition at line 471 of file DoubleDescription_Rn.h.

4.15.3 Member Function Documentation

4.15.3.1 template<class POLYHEDRON > virtual bool NoRedundancyProcessing< POLYHEDRON >::checkNeighbours (POLYHEDRON *poly*, const boost::shared_ptr< Generator_Rn > & *genIn*, const boost::shared_ptr< Generator_Rn > & *genOut*, std::vector< boost::shared_ptr< HalfSpace_Rn > > & *commonFacets*) [inline], [virtual]

Check whether two generators are neighbors in the context of not taking into account redundancy.

Definition at line 474 of file DoubleDescription_Rn.h.

4.15.3.2 template<class POLYHEDRON > virtual bool NoRedundancyProcessing< POLYHEDRON >::checkNeighbours (POLYHEDRON *poly*, const boost::shared_ptr< Generator_Rn > & *genIn*, const boost::shared_ptr< Generator_Rn > & *genOut*, std::vector< HalfSpace_Rn ∗ > & *commonFacets*) [inline], [virtual]

Check whether two generators are neighbors in the context of not taking into account redundancy.

Definition at line 483 of file DoubleDescription_Rn.h.

4.15.3.3 template<class POLYHEDRON > void NoRedundancyProcessing< POLYHEDRON >::decrementNumberForVerticesForHalfSpace (const boost::shared_ptr< Generator_Rn > &) [inline]

Only useful in the case of dealing and processing redundancy.

Definition at line 501 of file DoubleDescription_Rn.h.

4.15.3.4 template<class POLYHEDRON > void NoRedundancyProcessing< POLYHEDRON >::incrementNumberForVerticesForHalfSpace (const boost::shared_ptr< Generator_Rn > &) [inline]

Only useful in the case of dealing and processing redundancy.

Definition at line 498 of file DoubleDescription_Rn.h.

4.15.3.5 template<class POLYHEDRON > void NoRedundancyProcessing< POLYHEDRON >::initNumberOfVerticesPerHalfSpace (const std::vector< boost::shared_ptr< Generator_Rn > > &) [inline]

Only useful in the case of dealing and processing redundancy.

Definition at line 492 of file DoubleDescription_Rn.h.

4.15.3.6 template<class POLYHEDRON > void NoRedundancyProcessing< POLYHEDRON >::unhookRedundantHalfSpaces (POLYHEDRON) [inline]

Definition at line 506 of file DoubleDescription_Rn.h.

4.15.3.7 template<class POLYHEDRON > void NoRedundancyProcessing< POLYHEDRON >::updateListOfRedundantHalfSpaces (unsigned *int*) [inline]

Only useful in the case of dealing and processing redundancy.

Definition at line 504 of file DoubleDescription_Rn.h.

4.15.3.8 template<class POLYHEDRON > void NoRedundancyProcessing< POLYHEDRON >::updateNumberOfVerticesPerHalfSpace (const boost::shared_ptr< HalfSpace_Rn > & , unsigned *int*) [inline]

Only useful in the case of dealing and processing redundancy.

Definition at line 495 of file DoubleDescription_Rn.h.

The documentation for this class was generated from the following file:

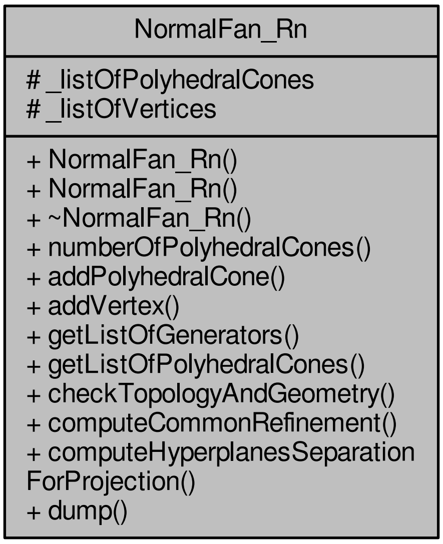
- /home/vindelos/CPP/I2M/politopix/trunk/DoubleDescription_Rn.h

4.16 NormalFan_Rn Class Reference

Model a normal fan.

`#include <NormalFan_Rn.h>`

Collaboration diagram for NormalFan_Rn:



Public Member Functions

- NormalFan_Rn ()

    Constructor.
- NormalFan_Rn (const boost::shared_ptr< Polytope_Rn > &A)

    Constructor.
- ∼NormalFan_Rn ()

    Destructor.
- unsigned int numberOfPolyhedralCones () const

    Get the total number of polyhedral cones.
- void addPolyhedralCone (boost::shared_ptr< PolyhedralCone_Rn > hs)

    Add the current half-space in its list.
- void addVertex (boost::shared_ptr< Generator_Rn > vx)

    Add the current vertex in its list.
- const std::vector
  < boost::shared_ptr
  < Generator_Rn > > & getListOfGenerators () const
- const std::vector
  < boost::shared_ptr
  < PolyhedralCone_Rn > > & getListOfPolyhedralCones () const
- bool checkTopologyAndGeometry () const throw (std::domain_error)
- void computeCommonRefinement (const NormalFan_Rn &NA, const NormalFan_Rn &NB)

    Compute $N(A+B) = N(A) \wedge N(B)$.
- void computeHyperplanesSeparationForProjection (const std::vector< boost::shared_ptr< HalfSpace_Rn > > &, boost::shared_ptr< Polytope_Rn > &)
- void dump (std::ostream &out) const

    Dump the polyhedral structure on std::cout.

Protected Attributes

- std::vector< boost::shared_ptr < PolyhedralCone_Rn > > _listOfPolyhedralCones

  The list of polyhedral cones partitioning the whole space.
- std::vector< boost::shared_ptr < Generator_Rn > > _listOfVertices

  The list of vertices attached to their respective dual polyhedral cones.

Friends

- class constIteratorOfListOfPolyhedralCones

4.16.1 Detailed Description

Model a normal fan.

Definition at line 37 of file NormalFan_Rn.h.

4.16.2 Constructor & Destructor Documentation

4.16.2.1 NormalFan_Rn::NormalFan_Rn ( ) [inline]

Constructor.

Definition at line 42 of file NormalFan_Rn.h.

4.16.2.2 NormalFan_Rn::NormalFan_Rn (const boost::shared_ptr< **Polytope_Rn** > & *A*)

Constructor.

Definition at line 31 of file NormalFan_Rn.cpp.

4.16.2.3 NormalFan_Rn::∼NormalFan_Rn ( ) [inline]

Destructor.

Definition at line 48 of file NormalFan_Rn.h.

4.16.3 Member Function Documentation

4.16.3.1 void NormalFan_Rn::addPolyhedralCone (boost::shared_ptr< **PolyhedralCone_Rn** > *hs*) [inline]

Add the current half-space in its list.

Definition at line 54 of file NormalFan_Rn.h.

4.16.3.2 void NormalFan_Rn::addVertex (boost::shared_ptr< **Generator_Rn** > *vx*) [inline]

Add the current vertex in its list.

Definition at line 57 of file NormalFan_Rn.h.

4.16.3.3 bool NormalFan_Rn::checkTopologyAndGeometry ( ) const throw std::domain_error)

Definition at line 70 of file NormalFan_Rn.cpp.

4.16.3.4 void NormalFan_Rn::computeCommonRefinement (const **NormalFan_Rn** & *NA,* const **NormalFan_Rn** & *NB*)

Compute $N(A+B) = N(A) \wedge N(B)$.

Compute the intersection of all polyhedral cones from the first normal fan N(A) with all polyhedral cones from the second normal fan N(B).

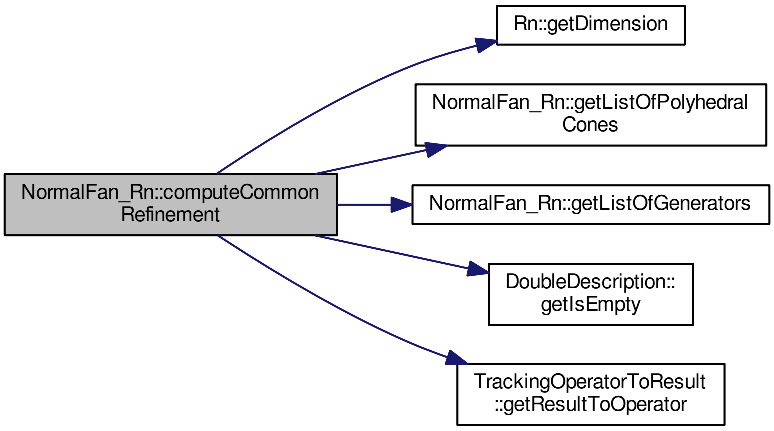
$$N(A+B) = N(A) \wedge N(B) = \left\{ C_{a_i} \bigcap C_{b_j}, \forall C_{a_i} \in N(A), \forall C_{b_j} \in N(B) \right\}$$

Parameters

NA	The first normal fan computed from polytope A
NB	The second normal fan computed from polytope B

Definition at line 174 of file NormalFan_Rn.cpp.

Here is the call graph for this function:


NormalFan_Rn::computeCommon Refinement
Rn::getDimension
NormalFan_Rn::getListOfPolyhedral Cones
NormalFan_Rn::getListOfGenerators
DoubleDescription:: getIsEmpty
TrackingOperatorToResult ::getResultToOperator

4.16.3.5 void NormalFan_Rn::computeHyperplanesSeparationForProjection (const std::vector< boost::shared_ptr< **HalfSpace_Rn** > > & *allHS,* boost::shared_ptr< **Polytope_Rn** > & *Pol*)

@

@

@

@

@

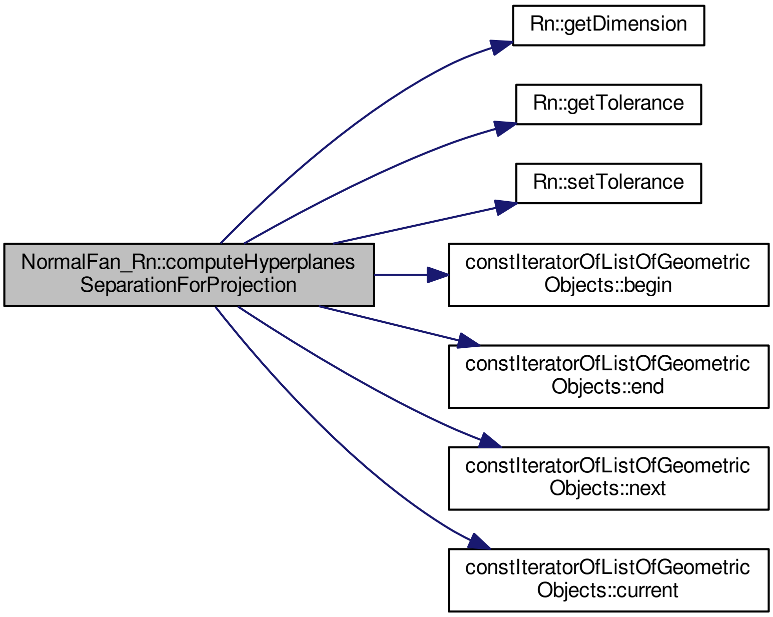
@

@

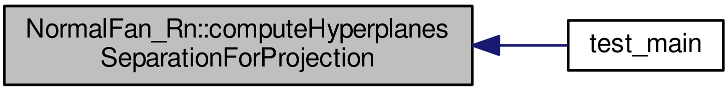
@

Definition at line 74 of file NormalFan_Rn.cpp.

Here is the call graph for this function:


NormalFan_Rn::computeHyperplanes SeparationForProjection
Rn::getDimension
Rn::getTolerance
Rn::setTolerance
constIteratorOfListOfGeometric Objects::begin
constIteratorOfListOfGeometric Objects::end
constIteratorOfListOfGeometric Objects::next
constIteratorOfListOfGeometric Objects::current

Here is the caller graph for this function:

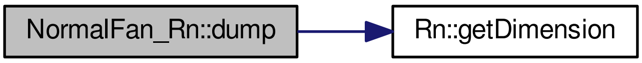


4.16.3.6 void NormalFan_Rn::dump (std::ostream & *out*) const

Dump the polyhedral structure on std::cout.

Definition at line 269 of file NormalFan_Rn.cpp.

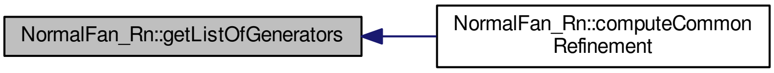
Here is the call graph for this function:



4.16.3.7 const std::vector< boost::shared_ptr<Generator_Rn> >& NormalFan_Rn::getListOfGenerators ( ) const [inline]

Definition at line 59 of file NormalFan_Rn.h.

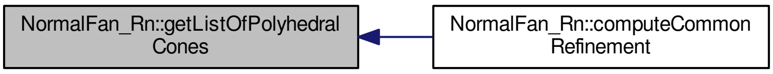
Here is the caller graph for this function:


NormalFan_Rn::getListOfGenerators
NormalFan_Rn::computeCommon Refinement

4.16.3.8 const std::vector< boost::shared_ptr<PolyhedralCone_Rn> >& NormalFan_Rn::getListOfPolyhedralCones ( ) const [inline]

Definition at line 61 of file NormalFan_Rn.h.

Here is the caller graph for this function:


NormalFan_Rn::getListOfPolyhedral Cones
NormalFan_Rn::computeCommon Refinement

4.16.3.9 unsigned int NormalFan_Rn::numberOfPolyhedralCones ( ) const [inline]

Get the total number of polyhedral cones.

Definition at line 51 of file NormalFan_Rn.h.

4.16.4 Friends And Related Function Documentation

4.16.4.1 friend class constIteratorOfListOfPolyhedralCones [friend]

Definition at line 38 of file NormalFan_Rn.h.

4.16.5 Member Data Documentation

4.16.5.1 std::vector< boost::shared_ptr<PolyhedralCone_Rn> > NormalFan_Rn::_listOfPolyhedralCones [protected]

The list of polyhedral cones partitioning the whole space.

Definition at line 86 of file NormalFan_Rn.h.

4.16.5.2 std::vector< boost::shared_ptr<Generator_Rn> > NormalFan_Rn::_listOfVertices [protected]

The list of vertices attached to their respective dual polyhedral cones.

Definition at line 88 of file NormalFan_Rn.h.

The documentation for this class was generated from the following files:

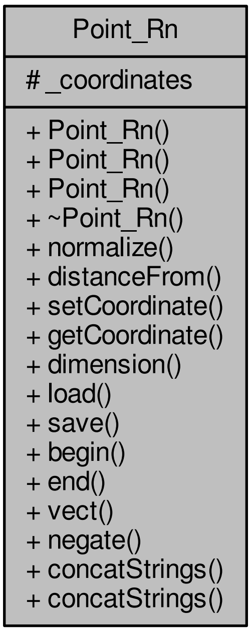
- /home/vindelos/CPP/I2M/politopix/trunk/NormalFan_Rn.h
- /home/vindelos/CPP/I2M/politopix/trunk/NormalFan_Rn.cpp

4.17 Point_Rn Class Reference

Creation of a n-coordinate geometric point designed to be shared by its neighbour faces.

```
#include <Point_Rn.h>
```

Collaboration diagram for Point_Rn:



Public Member Functions

- Point_Rn (unsigned int n)

    Create a n-coordinates point.
- Point_Rn (unsigned int n, double u)

    Fill the n-dimensional point with the constant u.
- Point_Rn (double u1, double u2, double u3)

    Create a 3-dimensional point.
- virtual ∼Point_Rn ()
- double normalize ()
- double distanceFrom (const Point_Rn &)
- void setCoordinate (unsigned int i, double val) throw (std::out_of_range)
- double getCoordinate (unsigned int i) const throw (std::out_of_range)
- int dimension () const
- void load (std::istream &this_istream)
- void save (std::ostream &this_ostream) const
- vector< double >::const_iterator begin () const
- vector< double >::const_iterator end () const
- const vector< double > & vect () const
- void negate ()

Static Public Member Functions

- static std::string concatStrings (int i, const std::string &functionName)

    Useful function to provide error message to the exception mechanism.

- static std::string concatStrings (int i, double val, const std::string &functionName)

  Useful function to provide error message to the exception mechanism.

Protected Attributes

- vector< double > _coordinates

4.17.1 Detailed Description

Creation of a n-coordinate geometric point designed to be shared by its neighbour faces.

Definition at line 34 of file Point_Rn.h.

4.17.2 Constructor & Destructor Documentation

4.17.2.1 Point_Rn::Point_Rn (unsigned int *n*)

Create a n-coordinates point.

Definition at line 30 of file Point_Rn.cpp.

4.17.2.2 Point_Rn::Point_Rn (unsigned int *n,* double *u*)

Fill the n-dimensional point with the constant u.

Definition at line 34 of file Point_Rn.cpp.

4.17.2.3 Point_Rn::Point_Rn (double *u1,* double *u2,* double *u3*)

Create a 3-dimensional point.

Definition at line 38 of file Point_Rn.cpp.

4.17.2.4 Point_Rn::∼Point_Rn ( ) [virtual]

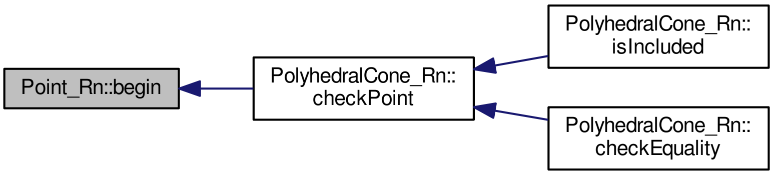
Definition at line 45 of file Point_Rn.cpp.

4.17.3 Member Function Documentation

4.17.3.1 vector<double>::const_iterator Point_Rn::begin ( ) const [inline]

Definition at line 64 of file Point_Rn.h.

Here is the caller graph for this function:

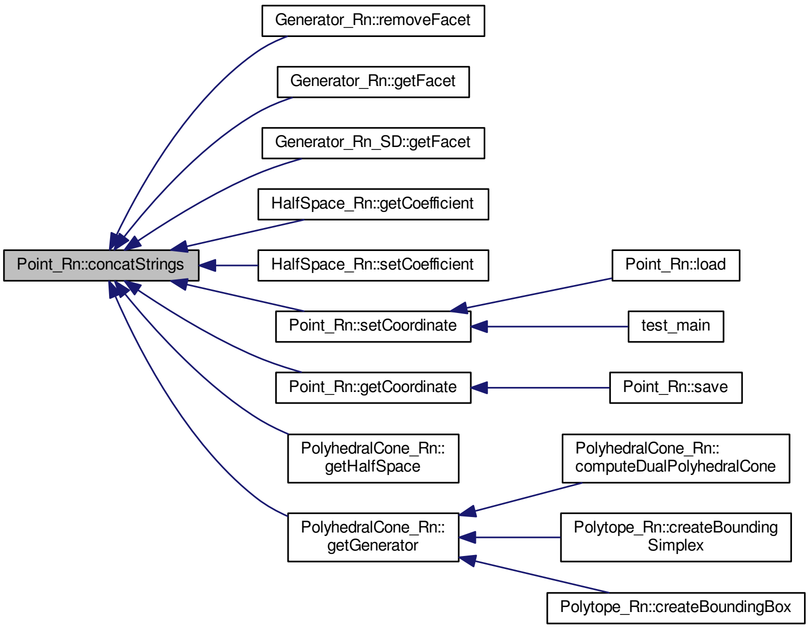

Point_Rn::begin
PolyhedralCone_Rn::checkPoint
PolyhedralCone_Rn::isIncluded
PolyhedralCone_Rn::checkEquality

4.17.3.2 std::string Point_Rn::concatStrings (int *i*, const std::string & *functionName*) [static]

Useful function to provide error message to the exception mechanism.

Definition at line 79 of file Point_Rn.cpp.

Here is the caller graph for this function:


Generator_Rn::removeFacet
Generator_Rn::getFacet
Generator_Rn_SD::getFacet
HalfSpace_Rn::getCoefficient
Point_Rn::concatStrings
HalfSpace_Rn::setCoefficient
Point_Rn::load
Point_Rn::setCoordinate
test_main
Point_Rn::getCoordinate
Point_Rn::save
PolyhedralCone_Rn::getHalfSpace
PolyhedralCone_Rn::computeDualPolyhedralCone
PolyhedralCone_Rn::getGenerator
Polytope_Rn::createBoundingSimplex
Polytope_Rn::createBoundingBox

4.17.3.3 std::string Point_Rn::concatStrings (int *i*, double *val*, const std::string & *functionName*) [static]

Useful function to provide error message to the exception mechanism.

Definition at line 93 of file Point_Rn.cpp.

4.17.3.4 int Point_Rn::dimension ( ) const [inline]

Definition at line 58 of file Point_Rn.h.

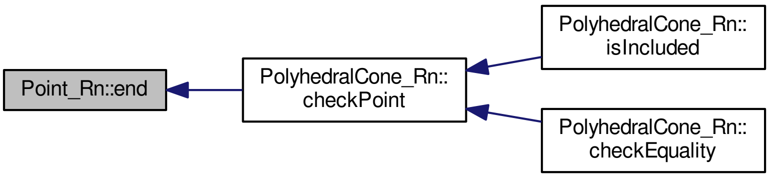
4.17.3.5 double Point_Rn::distanceFrom (const Point_Rn & *P*)

Definition at line 54 of file Point_Rn.cpp.

4.17.3.6 vector<double>::const_iterator Point_Rn::end ( ) const [inline]

Definition at line 66 of file Point_Rn.h.

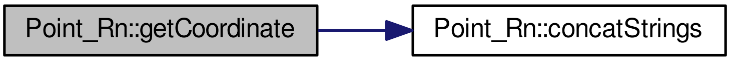
Here is the caller graph for this function:



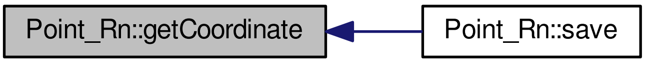
4.17.3.7 double Point_Rn::getCoordinate (unsigned int *i*) const throw std::out_of_range)

Definition at line 70 of file Point_Rn.cpp.

Here is the call graph for this function:



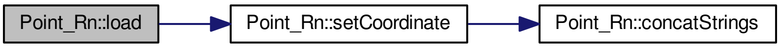
Here is the caller graph for this function:



4.17.3.8 void Point_Rn::load (std::istream & *this_istream*)

Definition at line 109 of file Point_Rn.cpp.

Here is the call graph for this function:



4.17.3.9 void Point_Rn::negate ( ) [inline]

Definition at line 70 of file Point_Rn.h.

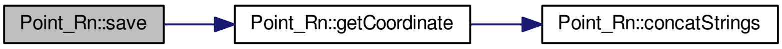
4.17.3.10 double Point_Rn::normalize ( )

Definition at line 48 of file Point_Rn.cpp.

4.17.3.11 void Point_Rn::save (std::ostream & *this_ostream*) const

Definition at line 117 of file Point_Rn.cpp.

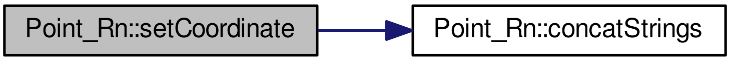
Here is the call graph for this function:



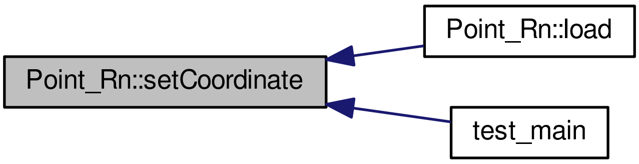
4.17.3.12 void Point_Rn::setCoordinate (unsigned int *i*, double *val*) throw std::out_of_range)

Definition at line 61 of file Point_Rn.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



4.17.3.13 const vector<double>& Point_Rn::vect ( ) const [inline]

Definition at line 68 of file Point_Rn.h.

4.17.4 Member Data Documentation

4.17.4.1 vector<double> Point_Rn::_coordinates [protected]

Definition at line 80 of file Point_Rn.h.

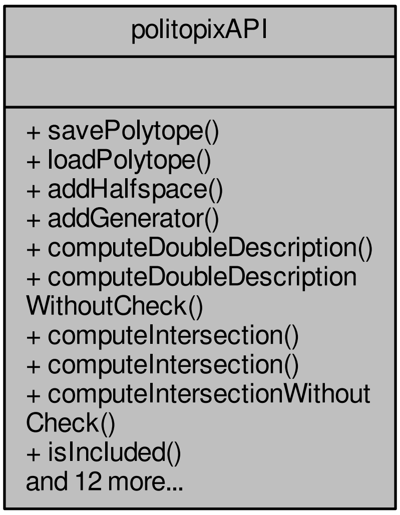
The documentation for this class was generated from the following files:

- /home/vindelos/CPP/I2M/politopix/trunk/Point_Rn.h
- /home/vindelos/CPP/I2M/politopix/trunk/Point_Rn.cpp

4.18 politopixAPI Class Reference

```
#include <politopixAPI.h>
```

Collaboration diagram for politopixAPI:



Static Public Member Functions

- static polito_EXPORT int savePolytope (const string &pathA, boost::shared_ptr< Polytope_Rn > &A) throw (ios_base::failure)

  Save a polytope in the corresponding file name.
- static polito_EXPORT int loadPolytope (const string &pathA, boost::shared_ptr< Polytope_Rn > &A) throw (ios_base::failure)

  Load a polytope from the corresponding file name.
- static polito_EXPORT int addHalfspace (boost::shared_ptr< Polytope_Rn > &A, const boost::shared_ptr< HalfSpace_Rn > &HS)

  Add a new half-space into the polytope data structure.
- static polito_EXPORT int addGenerator (boost::shared_ptr< Polytope_Rn > &A, const boost::shared_ptr< Generator_Rn > &GN)

  Add a new generator into the polytope data structure.
- static polito_EXPORT int computeDoubleDescription (boost::shared_ptr< Polytope_Rn > &A, double bb_size=1000.) throw (invalid_argument, out_of_range, ios_base::failure, logic_error)

  Compute the HV-description for a given H-polytope or V-polytope with the Double Description algorithm.
- static polito_EXPORT int computeDoubleDescriptionWithoutCheck (boost::shared_ptr< Polytope_Rn > &A, double bb_size=1000.) throw (invalid_argument, out_of_range, ios_base::failure, logic_error)

  Compute the HV-description for a given H-polytope or V-polytope with the Double Description algorithm.
- static polito_EXPORT int computeIntersection (const boost::shared_ptr< Polytope_Rn > &A, const boost::shared_ptr< Polytope_Rn > &B, boost::shared_ptr< Polytope_Rn > &C) throw (invalid_argument, out_of_range, ios_base::failure, logic_error)

  Compute the intersection between two HV-polytopes with the Double Description algorithm.
- static polito_EXPORT int computeIntersection (boost::shared_ptr< Polytope_Rn > &A, const boost::shared_ptr< Polytope_Rn > &B) throw (invalid_argument, out_of_range, ios_base::failure, logic_error)

  Compute the intersection between two HV-polytopes with the Double Description algorithm.

- static polito_EXPORT int computeIntersectionWithoutCheck (boost::shared_ptr< Polytope_Rn > &A, const boost::shared_ptr< Polytope_Rn > &B) throw (invalid_argument, out_of_range, ios_base::failure, logic_error)

  Compute the intersection between two HV-polytopes with the Double Description algorithm.
- static polito_EXPORT bool isIncluded (const boost::shared_ptr< Polytope_Rn > &A, const boost::shared_ptr< Polytope_Rn > &B) throw (invalid_argument, out_of_range, ios_base::failure, logic_error)

  Test whether the polytope A V-description is inside the polytope B H-description.
- static polito_EXPORT int computeMinkowskiSumOfPolytopes (const boost::shared_ptr< Polytope_Rn > &A, const boost::shared_ptr< Polytope_Rn > &B, boost::shared_ptr< Polytope_Rn > &C) throw (invalid_argument, out_of_range, ios_base::failure, logic_error)

  Compute the Minkowski sum between two HV-polytopes.
- static polito_EXPORT int computeMinkowskiSumOfPolytopes (const boost::shared_ptr< Polytope_Rn > &A, const boost::shared_ptr< Polytope_Rn > &B, boost::shared_ptr< Polytope_Rn > &C, const std::vector< std::vector< int > > &genitorsOfGeneratorsA, const std::vector< std::vector< int > > &genitorsOfGeneratorsB, std::vector< std::vector< int > > &traceGenerators) throw (invalid_argument, out_of_range, ios_base::failure, logic_error)

  Compute the Minkowski sum between two HV-polytopes tracing the generators.
- static polito_EXPORT int checkEqualityOfPolytopes (const boost::shared_ptr< Polytope_Rn > &A, const boost::shared_ptr< Polytope_Rn > &B, bool getFaceMapping=false) throw (invalid_argument, out_of_range, ios_base::failure, logic_error)

  Check whether two HV-polytopes are identical Check whether the vertices of A are inside B half-spaces and vice-versa. Perform also some topological verifications.
- static polito_EXPORT bool checkEqualityOfVertices (const boost::shared_ptr< Polytope_Rn > &A, const boost::shared_ptr< Polytope_Rn > &B) throw (invalid_argument, out_of_range, ios_base::failure, logic_error)

  Check whether two V-polytopes are identical Check whether the sets of vertices of A and B are equal.
- static polito_EXPORT int checkTopologyAndGeometry (const boost::shared_ptr< PolyhedralCone_Rn > &A)

  Check whether a HV-polytopes is correct.
- static polito_EXPORT int makeCube (boost::shared_ptr< Polytope_Rn > &A, double M)

  Create a cube whose vertices will be (+-M, ..., +-M)
- static polito_EXPORT int PolarPolytope (const boost::shared_ptr< Polytope_Rn > &original_pol, boost::shared_ptr< Polytope_Rn > &polar_pol)

  Compute the polar polytope.
- static polito_EXPORT int Translate (boost::shared_ptr< Polytope_Rn > &pol, const boost::numeric::ublas::vector< double > &v2t)

  Translate a polytope or polyhedral cone by the given vector.
- static polito_EXPORT double computeVolume (const boost::shared_ptr< Polytope_Rn > P) throw (invalid_argument, out_of_range, ios_base::failure)

  Return the volume of the given polytope P with its double description. The implemented algorithm can be found in Volume Computation for Polytopes: Strategies and Performances by ***Andreas Enge*** *in Encyclopedia of Optimization 2nd edition, p 4032-4073.*
- static polito_EXPORT int pseudoIntersection (const boost::shared_ptr< Polytope_Rn > &A, const boost::shared_ptr< Polytope_Rn > &B, boost::shared_ptr< Polytope_Rn > &C, const std::set< unsigned int > &firstOperandCaps, const std::set< unsigned int > &secondOperandCaps, std::set< unsigned int > &newCaps, double bb_size=1000.) throw (invalid_argument,out_of_range,ios_base::failure,logic_error)

  Remove all cap half-spaces and then compute the intersection of two capped polytopes.
- static polito_EXPORT int pseudoSum (const boost::shared_ptr< Polytope_Rn > &A, const boost::shared_ptr< Polytope_Rn > &B, boost::shared_ptr< Polytope_Rn > &C, const std::set< unsigned int > &firstOperandCaps, const std::set< unsigned int > &secondOperandCaps, std::set< unsigned int > &newCaps, double bb_size=1000.) throw (invalid_argument,out_of_range,ios_base::failure,logic_error)

  Compute the Minkowski sum of two polytopes and then remove all cap half-spaces to truncate again.
- static polito_EXPORT int computeVoronoiDiagram (const boost::shared_ptr< Polytope_Rn > &inputSpace, const std::vector< Point_Rn > &listOfSeeds, std::vector< boost::shared_ptr< Polytope_Rn > > &VoronoiCells) throw (std::length_error)

  Compute the Voronoi Diagram in a n-dimensional space i.e. a partitioning of an input space into regions based on distance to points called seeds.

4.18.1 Detailed Description

Definition at line 41 of file politopixAPI.h.

4.18.2 Member Function Documentation

4.18.2.1 int politopixAPI::addGenerator (boost::shared_ptr< **Polytope_Rn** > & *A*, const boost::shared_ptr< **Generator_Rn** > & *GN*) [static]

Add a new generator into the polytope data structure.

Parameters

A	The current polytope
GN	The corresponding generator

Returns

TEST_OK or 0 if the process was successful, TEST_KO or -1 if something went wrong.

Definition at line 55 of file politopixAPI.cpp.

4.18.2.2 int politopixAPI::addHalfspace (boost::shared_ptr< **Polytope_Rn** > & *A*, const boost::shared_ptr< **HalfSpace_Rn** > & *HS*) [static]

Add a new half-space into the polytope data structure.

Parameters

A	The current polytope
HS	The corresponding half-space

Returns

TEST_OK or 0 if the process was successful, TEST_KO or -1 if something went wrong.

Definition at line 50 of file politopixAPI.cpp.

4.18.2.3 int politopixAPI::checkEqualityOfPolytopes (const boost::shared_ptr< **Polytope_Rn** > & *A*, const boost::shared_ptr< **Polytope_Rn** > & *B*, bool *getFaceMapping* = false) throw invalid_argument, out_of_range, ios_base::failure, logic_error) [static]

Check whether two HV-polytopes are identical Check whether the vertices of A are inside B half-spaces and vice-versa. Perform also some topological verifications.

Parameters

A	The 1st HV-polytope
B	The 2nd HV-polytope
getFaceMapping	If true, print the mapping between the generators and faces of both polytopes in case of equality.

Returns

TEST_OK or 0 if the process was successful, TEST_KO or -1 if something went wrong.

Definition at line 411 of file politopixAPI.cpp.

Here is the caller graph for this function:

politopixAPI::checkEquality OfPolytopes
test_main

4.18.2.4 bool politopixAPI::checkEqualityOfVertices (const boost::shared_ptr< Polytope_Rn > & *A*, const boost::shared_ptr< Polytope_Rn > & *B*) throw invalid_argument, out_of_range, ios_base::failure, logic_error) [static]

Check whether two V-polytopes are identical Check whether the sets of vertices of A and B are equal.

Parameters

A	The 1st V-polytope
B	The 2nd V-polytope

Returns

true if

$$\mathcal{V}_A = \mathcal{V}_B$$

, false otherwise.

Definition at line 449 of file politopixAPI.cpp.

Here is the caller graph for this function:

politopixAPI::checkEquality OfVertices
test_main

4.18.2.5 int politopixAPI::checkTopologyAndGeometry (const boost::shared_ptr< PolyhedralCone_Rn > & *A*) [static]

Check whether a HV-polytopes is correct.

Parameters

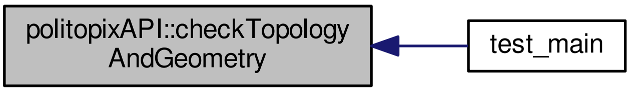
A	A HV-polytope

Returns

TEST_OK or 0 if the process was successful, TEST_KO or -1 if something went wrong.

Definition at line 515 of file politopixAPI.cpp.

Here is the caller graph for this function:



4.18.2.6 int politopixAPI::computeDoubleDescription (boost::shared_ptr< **Polytope_Rn** > & *A*, double *bb_size* = `1000.`) throw invalid_argument, out_of_range, ios_base::failure, logic_error) `[static]`

Compute the HV-description for a given H-polytope or V-polytope with the Double Description algorithm.

Parameters

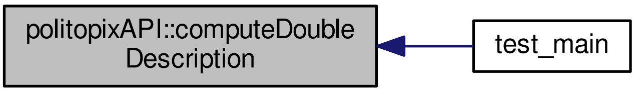
A	The H-polytope or V-polytope
bb_size	The origin centered bounding box size providing the V-description

Returns

TEST_OK or 0 if the process was successful, TEST_KO or -1 if something went wrong.

Definition at line 112 of file politopixAPI.cpp.

Here is the caller graph for this function:



4.18.2.7 int politopixAPI::computeDoubleDescriptionWithoutCheck (boost::shared_ptr< **Polytope_Rn** > & *A*, double *bb_size* = `1000.`) throw invalid_argument, out_of_range, ios_base::failure, logic_error) `[static]`

Compute the HV-description for a given H-polytope or V-polytope with the Double Description algorithm.

Parameters

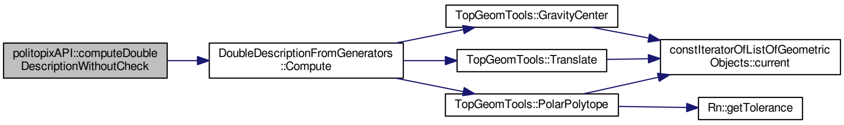
A	The H-polytope or V-polytope

Returns

TEST_OK or 0 if the process was successful, TEST_KO or -1 if something went wrong.

Definition at line 60 of file politopixAPI.cpp.

Here is the call graph for this function:


politopixAPI::computeDouble DescriptionWithoutCheck
DoubleDescriptionFromGenerators ::Compute
TopGeomTools::GravityCenter
TopGeomTools::Translate
TopGeomTools::PolarPolytope
constIteratorOfListOfGeometric Objects::current
Rn::getTolerance

4.18.2.8 int politopixAPI::computeIntersection (const boost::shared_ptr< Polytope_Rn > & *A*, const boost::shared_ptr< Polytope_Rn > & *B*, boost::shared_ptr< Polytope_Rn > & *C*) throw invalid_argument, out_of_range, ios_base::failure, logic_error) [static]

Compute the intersection between two HV-polytopes with the Double Description algorithm.

Parameters

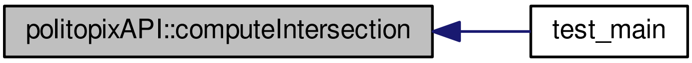
A	The 1st HV-polytope
B	The 2nd HV-polytope
C	The previously allocated result

Returns

TEST_OK or 0 if the process was successful, TEST_KO or -1 if something went wrong.

Definition at line 173 of file politopixAPI.cpp.

Here is the caller graph for this function:



4.18.2.9 int politopixAPI::computeIntersection (boost::shared_ptr< Polytope_Rn > & *A*, const boost::shared_ptr< Polytope_Rn > & *B*) throw invalid_argument, out_of_range, ios_base::failure, logic_error) [static]

Compute the intersection between two HV-polytopes with the Double Description algorithm.

Parameters

A	The 1st HV-polytope, then the result
B	The 2nd HV-polytope

Returns

TEST_OK or 0 if the process was successful, TEST_KO or -1 if something went wrong.

Definition at line 120 of file politopixAPI.cpp.

4.18.2.10 int politopixAPI::computeIntersectionWithoutCheck (boost::shared_ptr< **Polytope_Rn** > & *A*, const boost::shared_ptr< **Polytope_Rn** > & *B*) throw invalid_argument, out_of_range, ios_base::failure, logic_error) `[static]`

Compute the intersection between two HV-polytopes with the Double Description algorithm.

Parameters

A	The 1st HV-polytope, then the result
B	The 2nd HV-polytope

Returns

TEST_OK or 0 if the process was successful, TEST_KO or -1 if something went wrong.

Definition at line 278 of file politopixAPI.cpp.

4.18.2.11 int politopixAPI::computeMinkowskiSumOfPolytopes (const boost::shared_ptr< **Polytope_Rn** > & *A*, const boost::shared_ptr< **Polytope_Rn** > & *B*, boost::shared_ptr< **Polytope_Rn** > & *C*) throw invalid_argument, out_of_range, ios_base::failure, logic_error) `[static]`

Compute the Minkowski sum between two HV-polytopes.

Parameters

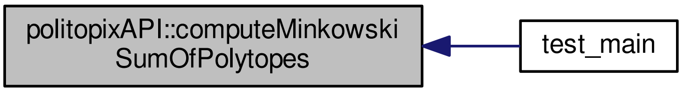
A	The 1st HV-polytope
B	The 2nd HV-polytope
C	The previously allocated sum

Returns

TEST_OK or 0 if the process was successful, TEST_KO or -1 if something went wrong.

Definition at line 331 of file politopixAPI.cpp.

Here is the caller graph for this function:



4.18.2.12 int politopixAPI::computeMinkowskiSumOfPolytopes (const boost::shared_ptr< Polytope_Rn > & *A*, const boost::shared_ptr< Polytope_Rn > & *B*, boost::shared_ptr< Polytope_Rn > & *C*, const std::vector< std::vector< int > > & *genitorsOfGeneratorsA*, const std::vector< std::vector< int > > & *genitorsOfGeneratorsB*, std::vector< std::vector< int > > & *traceGenerators*) throw invalid_argument, out_of_range, ios_base::failure, logic_error) [static]

Compute the Minkowski sum between two HV-polytopes tracing the generators.

Parameters

A	The 1st HV-polytope
B	The 2nd HV-polytope
C	The previously allocated sum
traceGenerators	Give for each generator of C, the list of numbers identifying its genitors in A and B

Returns

TEST_OK or 0 if the process was successful, TEST_KO or -1 if something went wrong.

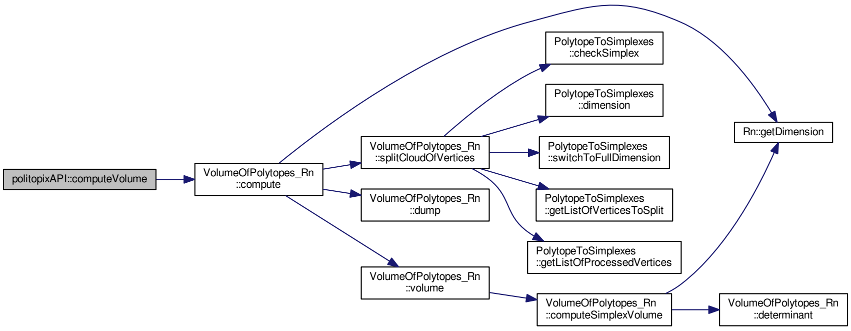
Definition at line 369 of file politopixAPI.cpp.

4.18.2.13 double politopixAPI::computeVolume (const boost::shared_ptr< **Polytope_Rn** > *P*) throw invalid_argument, out_of_range, ios_base::failure) `[static]`

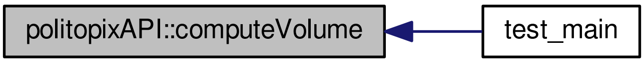
Return the volume of the given polytope P with its double description. The implemented algorithm can be found in *Volume Computation for Polytopes: Strategies and Performances* by **Andreas Enge** in *Encyclopedia of Optimization* 2nd edition, p 4032-4073.

Definition at line 487 of file politopixAPI.cpp.

Here is the call graph for this function:


politopixAPI::computeVolume
VolumeOfPolytopes_Rn ::compute
VolumeOfPolytopes_Rn ::splitCloudOfVertices
VolumeOfPolytopes_Rn ::dump
VolumeOfPolytopes_Rn ::volume
PolytopeToSimplexes ::checkSimplex
PolytopeToSimplexes ::dimension
PolytopeToSimplexes ::switchToFullDimension
PolytopeToSimplexes ::getListOfVerticesToSplit
PolytopeToSimplexes ::getListOfProcessedVertices
VolumeOfPolytopes_Rn ::computeSimplexVolume
VolumeOfPolytopes_Rn ::determinant
Rn::getDimension

Here is the caller graph for this function:



4.18.2.14 int politopixAPI::computeVoronoiDiagram (const boost::shared_ptr< **Polytope_Rn** > & *inputSpace,* const std::vector< **Point_Rn** > & *listOfSeeds,* std::vector< boost::shared_ptr< **Polytope_Rn** > > & *VoronoiCells*) throw std::length_error) `[static]`

Compute the Voronoi Diagram in a n-dimensional space i.e. a partitioning of an input space into regions based on distance to points called seeds.

Parameters

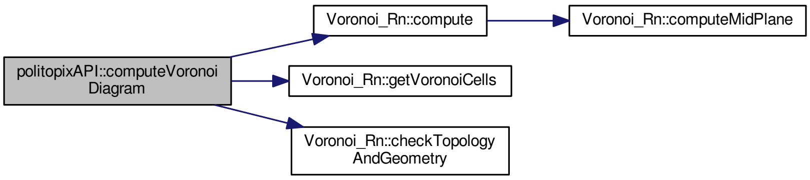
inputSpace	The input HV-polytope, most of the times a parallelepiped
listOfSeeds	The list of points to be considered as seeds
VoronoiCells	The list of the returned HV-polytopes partitioning the input space

Returns

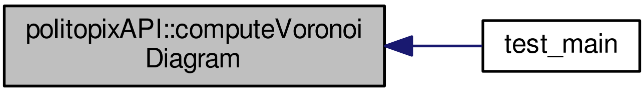
TEST_OK or 0 if the process was successful, TEST_KO or -1 if something went wrong.

Definition at line 629 of file politopixAPI.cpp.

Here is the call graph for this function:


Voronoi_Rn::compute
Voronoi_Rn::computeMidPlane
politopixAPI::computeVoronoi Diagram
Voronoi_Rn::getVoronoiCells
Voronoi_Rn::checkTopology AndGeometry

Here is the caller graph for this function:



4.18.2.15 bool politopixAPI::isIncluded (const boost::shared_ptr< **Polytope_Rn** > & *A*, const boost::shared_ptr< **Polytope_Rn** > & *B*) throw invalid_argument, out_of_range, ios_base::failure, logic_error) [static]

Test whether the polytope A V-description is inside the polytope B H-description.

Parameters

A	The 1st V-polytope
B	The 2nd H-polytope

Returns

true if

$$A \subset B$$

, false otherwise.

Definition at line 240 of file politopixAPI.cpp.

4.18.2.16 int politopixAPI::loadPolytope (const string & *pathA*, boost::shared_ptr< **Polytope_Rn** > & *A*) throw ios_base::failure) `[static]`

Load a polytope from the corresponding file name.

Parameters

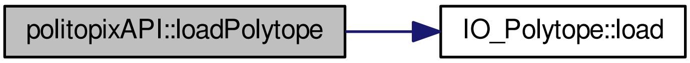
pathA	The name of the current file
A	The previously allocated polytope

Returns

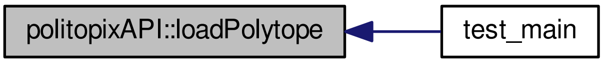
TEST_OK or 0 if the process was successful, TEST_KO or -1 if something went wrong.

Definition at line 38 of file politopixAPI.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



4.18.2.17 int politopixAPI::makeCube (boost::shared_ptr< Polytope_Rn > & *A*, double *M*) [static]

Create a cube whose vertices will be (+-M, ..., +-M)

Parameters

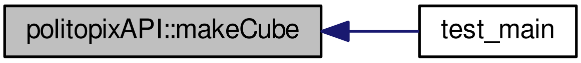
A	The cube variable
M	The cube half side length.

Returns

TEST_OK or 0 if the process was successful, TEST_KO or -1 if something went wrong.

Definition at line 519 of file politopixAPI.cpp.

Here is the caller graph for this function:



4.18.2.18 int politopixAPI::PolarPolytope (const boost::shared_ptr< **Polytope_Rn** > & *original_pol,* boost::shared_ptr< **Polytope_Rn** > & *polar_pol*) [static]

Compute the polar polytope.

Parameters

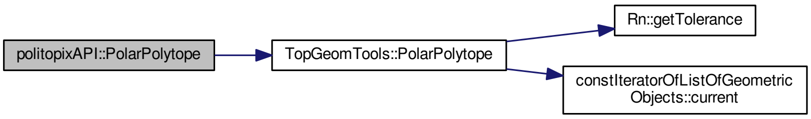
original_pol	The input polytope
polar_pol	The polar polytope

Returns

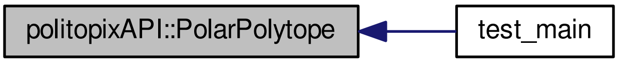
TEST_OK or 0 if the process was successful, TEST_KO or -1 if something went wrong.

Definition at line 525 of file politopixAPI.cpp.

Here is the call graph for this function:


politopixAPI::PolarPolytope
TopGeomTools::PolarPolytope
Rn::getTolerance
constIteratorOfListOfGeometric Objects::current

Here is the caller graph for this function:



4.18.2.19 int politopixAPI::pseudoIntersection (const boost::shared_ptr< Polytope_Rn > & *A,* const boost::shared_ptr< Polytope_Rn > & *B,* boost::shared_ptr< Polytope_Rn > & *C,* const std::set< unsigned int > & *firstOperandCaps,* const std::set< unsigned int > & *secondOperandCaps,* std::set< unsigned int > & *newCaps,* double *bb_size =* `1000.`) throw invalid_argument, out_of_range, ios_base::failure, logic_error) `[static]`

Remove all cap half-spaces and then compute the intersection of two capped polytopes.

Parameters

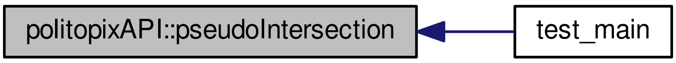
A	The 1st HV-polytope
B	The 2nd HV-polytope
C	The previously allocated intersection

Returns

TEST_OK or 0 if the process was successful, TEST_KO or -1 if something went wrong.

Definition at line 533 of file politopixAPI.cpp.

Here is the caller graph for this function:



4.18.2.20 int politopixAPI::pseudoSum (const boost::shared_ptr< **Polytope_Rn** > & *A,* const boost::shared_ptr< **Polytope_Rn** > & *B,* boost::shared_ptr< **Polytope_Rn** > & *C,* const std::set< unsigned int > & *firstOperandCaps,* const std::set< unsigned int > & *secondOperandCaps,* std::set< unsigned int > & *newCaps,* double *bb_size* = `1000.`) throw invalid_argument, out_of_range, ios_base::failure, logic_error) `[static]`

Compute the Minkowski sum of two polytopes and then remove all cap half-spaces to truncate again.

Parameters

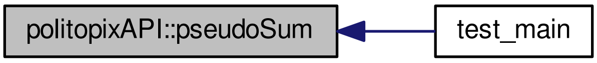
A	The 1st HV-polytope
B	The 2nd HV-polytope
C	The previously allocated sum

Returns

TEST_OK or 0 if the process was successful, TEST_KO or -1 if something went wrong.

Definition at line 581 of file politopixAPI.cpp.

Here is the caller graph for this function:



4.18.2.21 int politopixAPI::savePolytope (const string & *pathA*, boost::shared_ptr< **Polytope_Rn** > & *A*) throw ios_base::failure) [static]

Save a polytope in the corresponding file name.

Parameters

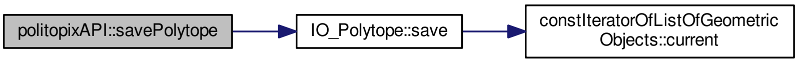
pathA	The name of the current file
A	The corresponding polytope

Returns

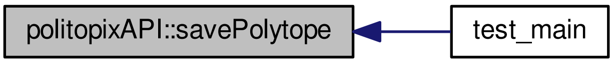
TEST_OK or 0 if the process was successful, TEST_KO or -1 if something went wrong.

Definition at line 26 of file politopixAPI.cpp.

Here is the call graph for this function:


politopixAPI::savePolytope
IO_Polytope::save
constIteratorOfListOfGeometric Objects::current

Here is the caller graph for this function:



4.18.2.22 int politopixAPI::Translate (boost::shared_ptr< Polytope_Rn > & *pol*, const boost::numeric::ublas::vector< double > & *v2t*) [static]

Translate a polytope or polyhedral cone by the given vector.

Parameters

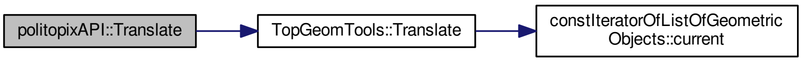
pol	The corresponding polytope
v2t	The translation vector

Returns

TEST_OK or 0 if the process was successful, TEST_KO or -1 if something went wrong.

Definition at line 529 of file politopixAPI.cpp.

Here is the call graph for this function:



The documentation for this class was generated from the following files:

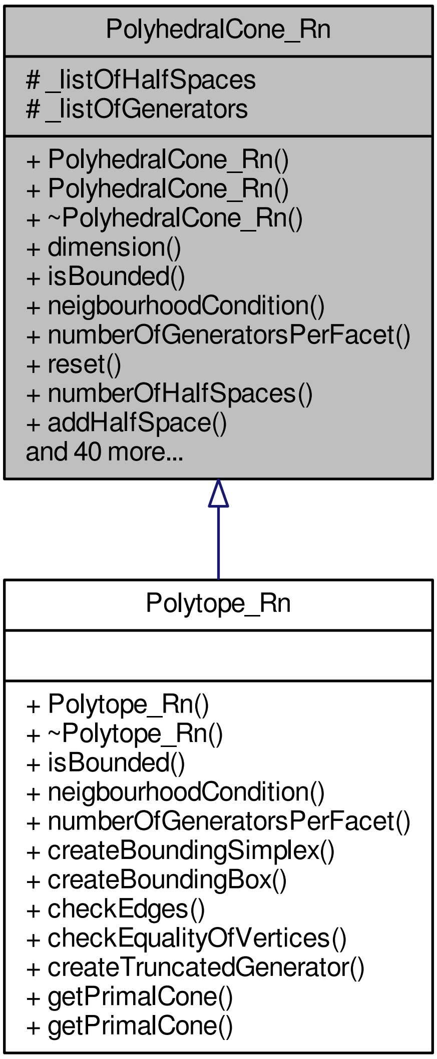
- /home/vindelos/CPP/I2M/politopix/trunk/politopixAPI.h
- /home/vindelos/CPP/I2M/politopix/trunk/politopixAPI.cpp

4.19 PolyhedralCone_Rn Class Reference

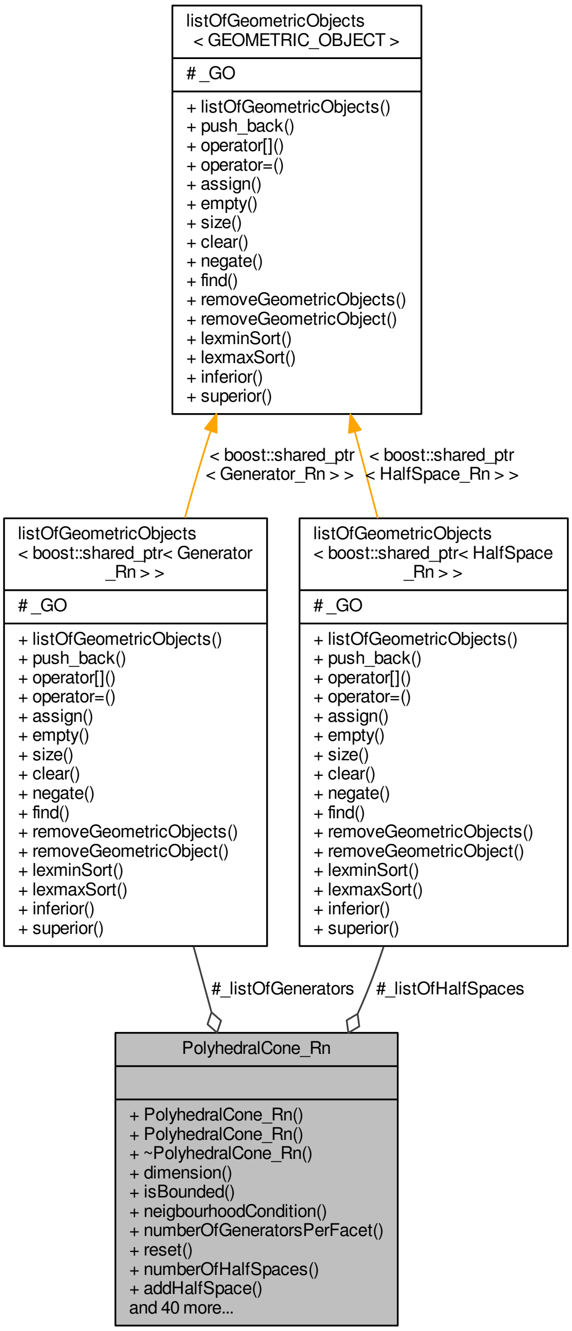
Model a polyhedral cone using its two equivalent definitions : the convex hull and the half-space intersection. We store its edges in _listOfHS and the positive combination of these vectors generates the polyhedral cone.

```
#include <PolyhedralCone_Rn.h>
```

Inheritance diagram for PolyhedralCone_Rn:



Collaboration diagram for PolyhedralCone_Rn:



Public Member Functions

- PolyhedralCone_Rn ()

    Constructor.

- PolyhedralCone_Rn (const PolyhedralCone_Rn &A)

    Constructor.

- virtual ∼PolyhedralCone_Rn ()

Destructor.

- virtual unsigned int dimension () const

  Return the space dimension.

- virtual bool isBounded () const

  Tell whether this polyhedron is bounded or not, polyhedral cones are not.

- virtual unsigned int neigbourhoodCondition () const

  Two edges are neighbours in a polyhedral cone <=> they share at least (n-2) facets.

- virtual unsigned int numberOfGeneratorsPerFacet () const

  Each facet in a polyhedral cone has got (n-1) edges.

- void reset ()

  Remove all half-spaces and generators.

- unsigned int numberOfHalfSpaces () const

  Get the total number of half-spaces.

- boost::shared_ptr< HalfSpace_Rn > addHalfSpace (boost::shared_ptr< HalfSpace_Rn > hs, bool check=false)

  Add the current half-space in its list.

- void removeHalfSpace (unsigned int j)

  Remove the half-space number j from its list.

- void removeHalfSpaces (const std::set< boost::shared_ptr< HalfSpace_Rn > > &setOfRedundantHS)

  Remove the half-space number j from its list.

- unsigned int getHalfSpaceNumber (const boost::shared_ptr< HalfSpace_Rn > &F) const throw (std::out_of_range)

  For a given half-space, return its list index.

- const boost::shared_ptr
  < HalfSpace_Rn > & getHalfSpace (unsigned int i) const throw (std::out_of_range)

  Return the i-th generator.

- listOfGeometricObjects
  < boost::shared_ptr
  < HalfSpace_Rn > > & getListOfHalfSpaces ()

  Return the list of half-spaces.

- const listOfGeometricObjects
  < boost::shared_ptr
  < HalfSpace_Rn > > & getListOfHalfSpaces () const

  Return the list of half-spaces.

- unsigned int numberOfGenerators () const

  Give the total number of generators.

- void addGenerator (boost::shared_ptr< Generator_Rn > vx)

  Add the given generator.

- const boost::shared_ptr
  < Generator_Rn > & getGenerator (unsigned int i) const throw (std::out_of_range)

  Return the i-th generator.

- unsigned int getGeneratorNumber (boost::shared_ptr< Generator_Rn > G) const throw (std::out_of_range,std::invalid_argument)

  For a given generator, return its list index.

- const listOfGeometricObjects
  < boost::shared_ptr
  < Generator_Rn > > & getListOfGenerators () const

  Return the list of generators.

- unsigned int getListOfGeneratorsSD (std::vector< boost::shared_ptr< Generator_Rn_SD > > &currentListOfGeneratorsSD)
- void setListOfGeneratorsSD (const std::vector< boost::shared_ptr< Generator_Rn_SD > > &gnList)

  Set a new list of generators. The list of half-spaces should have been previously set.

- void relocateGenerators ()
- void setListOfGenerators (const listOfGeometricObjects< boost::shared_ptr< Generator_Rn > > &gnList)

  Set a new list of generators.
- bool checkNeighbours (const boost::shared_ptr< Generator_Rn > &V1, const boost::shared_ptr< Generator_Rn > &V2, std::vector< boost::shared_ptr< HalfSpace_Rn > > &commonFacets, const std::set< boost::shared_ptr< HalfSpace_Rn > > &listOfRedundantHS) const throw (std::invalid_argument)
- bool isIncluded (const boost::shared_ptr< PolyhedralCone_Rn > &B) const

  Test whether the current polytope V-description is inside the polytope B H-description.
- bool checkNeighbours (const boost::shared_ptr< Generator_Rn > &V1, const boost::shared_ptr< Generator_Rn > &V2, std::vector< boost::shared_ptr< HalfSpace_Rn > > &commonFacets, unsigned int topologicalCode=1) const throw (std::invalid_argument)
- bool checkNeighbours (const boost::shared_ptr< Generator_Rn > &V1, const boost::shared_ptr< Generator_Rn > &V2, std::vector< HalfSpace_Rn ∗ > &commonFacets, unsigned int topologicalCode=1) const throw (std::invalid_argument)

  Check for polytopes that vertices share (n-1) facets. For polyhedral cones, it must check that vectors share (n-2) facets.
- bool checkNeighbours (const boost::shared_ptr< Generator_Rn_SD > &genIn, const boost::shared_ptr< Generator_Rn_SD > &genOut, std::vector< unsigned int > &commonFacets)

  Check for polytopes that vertices share (n-1) facets. For polyhedral cones, it must check that vectors share (n-2) facets.
- bool checkNeighboursWithHSnumbers (const boost::shared_ptr< Generator_Rn > &V1, const boost::shared_ptr< Generator_Rn > &V2, std::vector< HalfSpace_Rn ∗ > &commonFacets, const std::set< boost::shared_ptr< HalfSpace_Rn > > &listOfRedundantHS, unsigned int topologicalCode=1) const throw (std::invalid_argument)

  Check for polytopes that vertices share (n-1) facets. For polyhedral cones, it must check that vectors share (n-2) facets.
- void fillNeighbourMatrix (std::vector< std::vector< unsigned int > > &neighboursA, unsigned int topologicalCode=1) const throw (std::out_of_range)
- virtual bool checkTopologyAndGeometry () const

  *As stated by **Komei Fukuda** "the complexity of the polyhedral verification problem is unknown. Is it in P or in coNP-complete?" So only 3 verifications are made in R^n :*
- HalfSpace_Rn::State checkPoint (const Point_Rn &thisPoint) const

  Check a point state against the whole polyhedron.
- HalfSpace_Rn::State checkPoint (const boost::shared_ptr< Generator_Rn > &point, const boost::shared_ptr< HalfSpace_Rn > &halfSpace, double halfSpaceNorm) const

  Check a point state against a half-space.
- bool checkDuplicateGenerators (unsigned int &a, unsigned int &b)

  Make sure no duplicate generators are stored.
- void checkGenerator (unsigned int vtxNumber, std::ostream &this_ostream) const throw (std::out_of_range)

  Compute all distance from the current point to all half-spaces frontiers.
- bool checkGenerators (const listOfGeometricObjects< boost::shared_ptr< Generator_Rn > > &listGenA, const listOfGeometricObjects< boost::shared_ptr< HalfSpace_Rn > > &listHSB, bool check_all=false) const

  Check the number of facets per generator and make sure it is compliant with its current constraints. It must verify the following property :

  $$\forall G = (g_1,...,g_n) \in \mathbb{R}^n, \exists (n-1) \bar{H}_i = \left\{ x : \sum_{j=1}^{n} a_{ij} x_j \geq 0 \right\} such as G \in \bar{H}_i, i \in \{1,...,n-1\}$$

  .
- void removeGenerator (unsigned int j)

  Remove the generator number j from its list.
- virtual bool checkEdges () const

  Always true in the polyhedral cone case.
- void checkFacet (unsigned int fctNumber, std::ostream &this_ostream) const throw (std::out_of_range)

- bool checkFacets () const

  Detect redundant half spaces and make sure each facet has the right number of generators.

  $$A = Conv(G_1,...,G_k) = \bigcap_{i=1}^{m} \bar{H}_i^+$$

  Check that the polytope does not have generators violating a constraint defined by a half-space.

  $$\forall G = (g_1,...,g_n) \in \mathbb{R}^n, \nexists \bar{H}_i^+ = \left\{ x : \sum_{j=1}^{n} a_{ij}x_j \geq 0 \right\} such\, as\, G \notin \bar{H}_i^+$$

  Check that the polytope does have at least (n-1) generators per half-space frontier.

  $$\forall \bar{H}_i, \exists (n-1)\, G = (g_1,...,g_n) \in \mathbb{R}^n, such\, as\, G \in \bar{H}_i$$

  .
- bool checkEquality (const boost::shared_ptr< PolyhedralCone_Rn > &B, bool getFaceMapping=false) const
- void negate ()

  Compute the symmetrical polyhedral cone.
- virtual void createTruncatedGenerator (const boost::shared_ptr< Generator_Rn_SD > &y, const boost::shared_ptr< Generator_Rn_SD > &z, boost::shared_ptr< Generator_Rn_SD > newG, double ay, double az, double b=0.) const

  Create the intersection edge in the truncating algorithm. It is defined by the intersection between a 2-face and a hyperplane, i.e. a (n-1)-face. The new egde is given by this formula where H is the current half space :

  $$newG = \frac{\langle a,z\rangle}{\langle a,z-y\rangle} y - \frac{\langle a,y\rangle}{\langle a,z-y\rangle} z, H = \left\{ x : \langle a,x\rangle = \sum_{j=1}^{n} a_j x_j \geq 0 \right\}$$

  .
- boost::shared_ptr
  < PolyhedralCone_Rn > computeDualPolyhedralCone () const

  Build the dual polyhedral cone of the current one whose edge are orthogonal to the primal and vice-versa.
- virtual void computeMinkowskiSum (const boost::shared_ptr< PolyhedralCone_Rn > &A, const boost::shared_ptr< PolyhedralCone_Rn > &B)

  Compute the Minkowski sum of two polyhedral cones.
- void dump (std::ostream &this_ostream) const

  Dump the polyhedral structure on std::cout.
- virtual void createBoundingBox (double)

  At the moment this function is useless in the case of polyhedral cones.
- virtual void createBoundingSimplex (double)

  At the moment this function is useless in the case of polyhedral cones.

Protected Attributes

- listOfGeometricObjects
  < boost::shared_ptr
  < HalfSpace_Rn > > _listOfHalfSpaces

  The list of half-spaces defining the polytope.
- listOfGeometricObjects
  < boost::shared_ptr
  < Generator_Rn > > _listOfGenerators

  The convex hull of connected points.

Friends

- class lexIteratorOfListOfHalfSpaces
- class constIteratorOfListOfHalfSpaces

4.19.1 Detailed Description

Model a polyhedral cone using its two equivalent definitions : the convex hull and the half-space intersection. We store its edges in _listOfHS and the positive combination of these vectors generates the polyhedral cone.

Definition at line 45 of file PolyhedralCone_Rn.h.

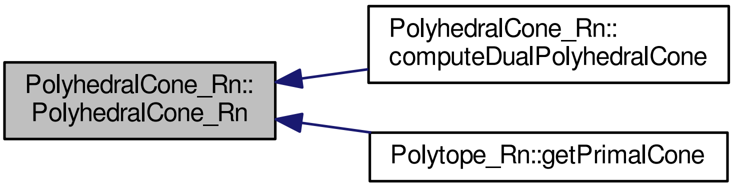
4.19.2 Constructor & Destructor Documentation

4.19.2.1 PolyhedralCone_Rn::PolyhedralCone_Rn ( ) [inline]

Constructor.

Definition at line 51 of file PolyhedralCone_Rn.h.

Here is the caller graph for this function:



4.19.2.2 PolyhedralCone_Rn::PolyhedralCone_Rn (const PolyhedralCone_Rn & *A*) [inline]

Constructor.

Definition at line 54 of file PolyhedralCone_Rn.h.

4.19.2.3 virtual PolyhedralCone_Rn::∼PolyhedralCone_Rn ( ) [inline], [virtual]

Destructor.

Definition at line 60 of file PolyhedralCone_Rn.h.

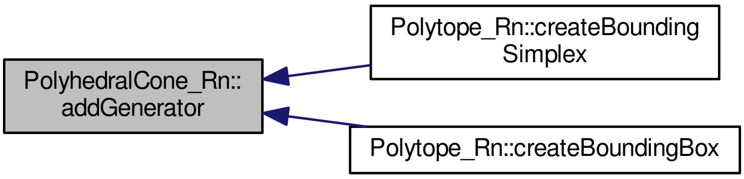
4.19.3 Member Function Documentation

4.19.3.1 void PolyhedralCone_Rn::addGenerator (boost::shared_ptr< Generator_Rn > *vx*) [inline]

Add the given generator.

Definition at line 136 of file PolyhedralCone_Rn.h.

Here is the caller graph for this function:

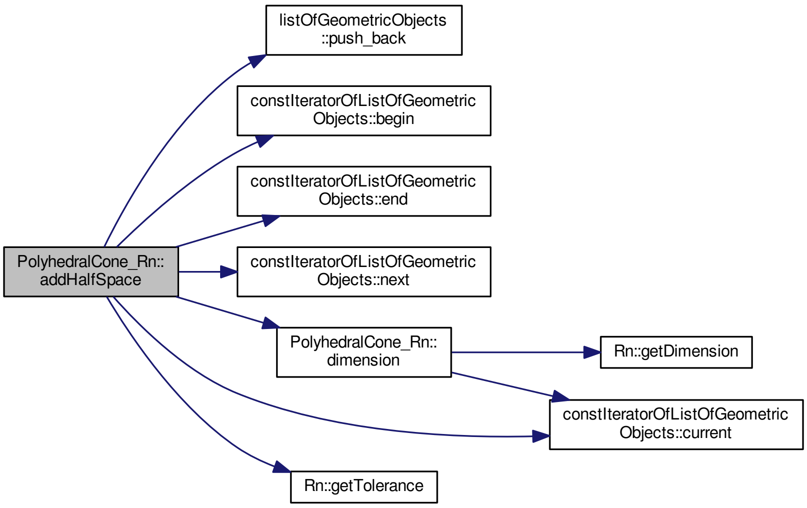


4.19.3.2 boost::shared_ptr< HalfSpace_Rn > PolyhedralCone_Rn::addHalfSpace (boost::shared_ptr< HalfSpace_Rn > *hs*, bool *check* = `false`)

Add the current half-space in its list.

Definition at line 137 of file PolyhedralCone_Rn.cpp.

Here is the call graph for this function:



4.19.3.3 bool PolyhedralCone_Rn::checkDuplicateGenerators (unsigned int & *a*, unsigned int & *b*)

Make sure no duplicate generators are stored.

Parameters

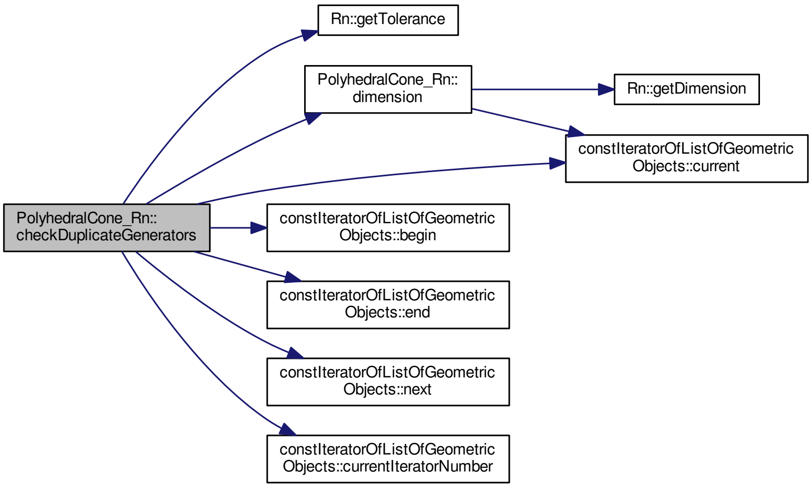
a	in case of equality store in this parameter the index of the first equal generator
b	in case of equality store in this parameter the index of the second equal generator

Returns

true if there are equal generators, false otherwise.

Definition at line 70 of file PolyhedralCone_Rn.cpp.

Here is the call graph for this function:


Rn::getTolerance
PolyhedralCone_Rn::dimension
Rn::getDimension
constIteratorOfListOfGeometricObjects::current
PolyhedralCone_Rn::checkDuplicateGenerators
constIteratorOfListOfGeometricObjects::begin
constIteratorOfListOfGeometricObjects::end
constIteratorOfListOfGeometricObjects::next
constIteratorOfListOfGeometricObjects::currentIteratorNumber

4.19.3.4 virtual bool PolyhedralCone_Rn::checkEdges ( ) const [inline], [virtual]

Always true in the polyhedral cone case.

Reimplemented in Polytope_Rn.

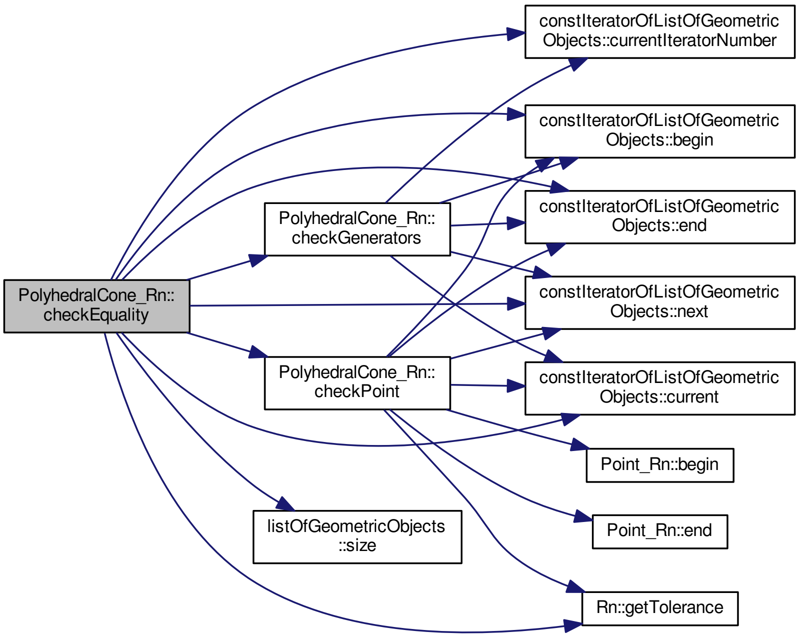
Definition at line 515 of file PolyhedralCone_Rn.h.

4.19.3.5 bool PolyhedralCone_Rn::checkEquality (const boost::shared_ptr< PolyhedralCone_Rn > & *B*, bool *getFaceMapping* = false) const

Check whether the current polyhedral cones and B are equal or not. If the last variable is true, print the mapping between the generators and faces of both polyhedral cones.

Definition at line 382 of file PolyhedralCone_Rn.cpp.

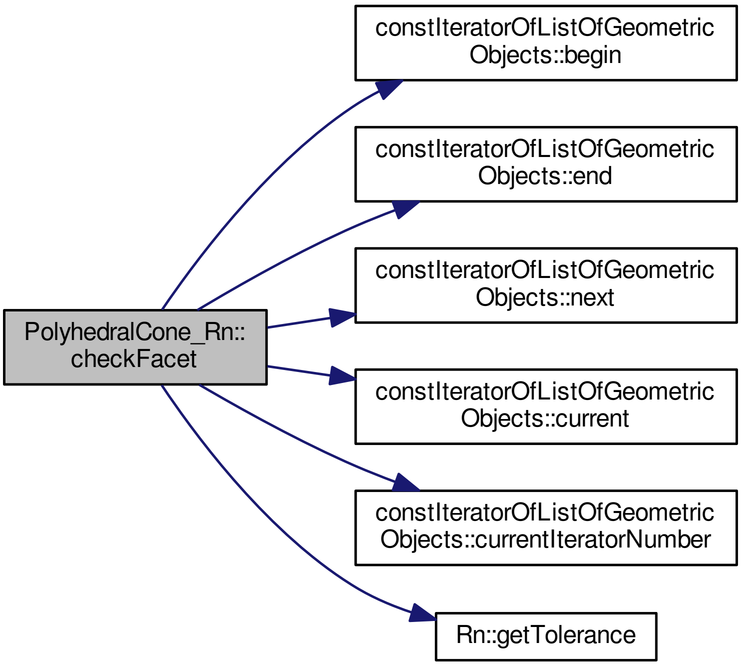
Here is the call graph for this function:


constIteratorOfListOfGeometric Objects::currentIteratorNumber
constIteratorOfListOfGeometric Objects::begin
constIteratorOfListOfGeometric Objects::end
constIteratorOfListOfGeometric Objects::next
constIteratorOfListOfGeometric Objects::current
PolyhedralCone_Rn:: checkGenerators
PolyhedralCone_Rn:: checkEquality
PolyhedralCone_Rn:: checkPoint
Point_Rn::begin
Point_Rn::end
listOfGeometricObjects ::size
Rn::getTolerance

4.19.3.6 void PolyhedralCone_Rn::checkFacet (unsigned int *fctNumber*, std::ostream & *this_ostream*) const throw std::out_of_range) [inline]

Definition at line 517 of file PolyhedralCone_Rn.h.

Here is the call graph for this function:


PolyhedralCone_Rn::
checkFacet
constIteratorOfListOfGeometricObjects::begin
constIteratorOfListOfGeometricObjects::end
constIteratorOfListOfGeometricObjects::next
constIteratorOfListOfGeometricObjects::current
constIteratorOfListOfGeometricObjects::currentIteratorNumber
Rn::getTolerance

4.19.3.7 bool PolyhedralCone_Rn::checkFacets ( ) const [inline]

Detect redundant half spaces and make sure each facet has the right number of generators.

$$A = Conv(G_1, ..., G_k) = \bigcap_{i=1}^{m} \bar{H}_i^+$$

Check that the polytope does not have generators violating a constraint defined by a half-space.

$$\forall G = (g_1, ..., g_n) \in \mathbb{R}^n, \nexists \bar{H}_i^+ = \left\{ x : \sum_{j=1}^{n} a_{ij} x_j \geq 0 \right\} such as G \notin \bar{H}_i^+$$

Check that the polytope does have at least (n-1) generators per half-space frontier.

$$\forall \bar{H}_i, \exists (n-1)\, G = (g_1, ..., g_n) \in \mathbb{R}^n, such as G \in \bar{H}_i$$

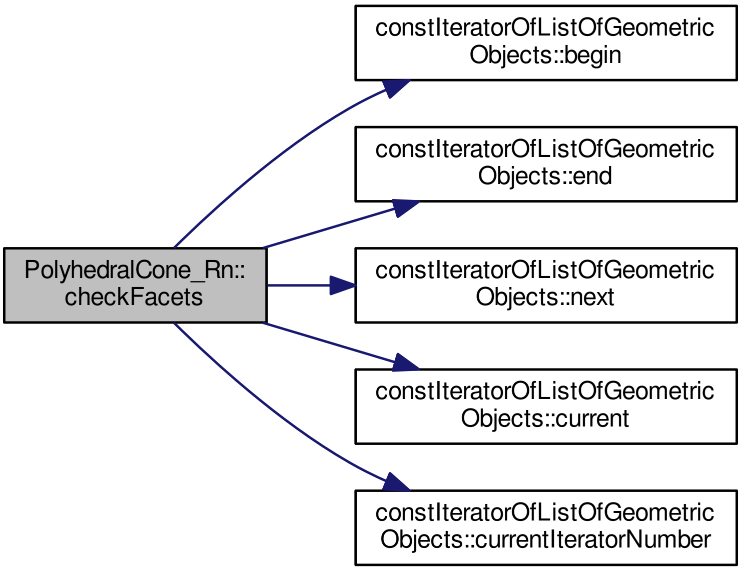
.

Returns

true if everything's OK, false otherwise.

Definition at line 579 of file PolyhedralCone_Rn.h.

Here is the call graph for this function:

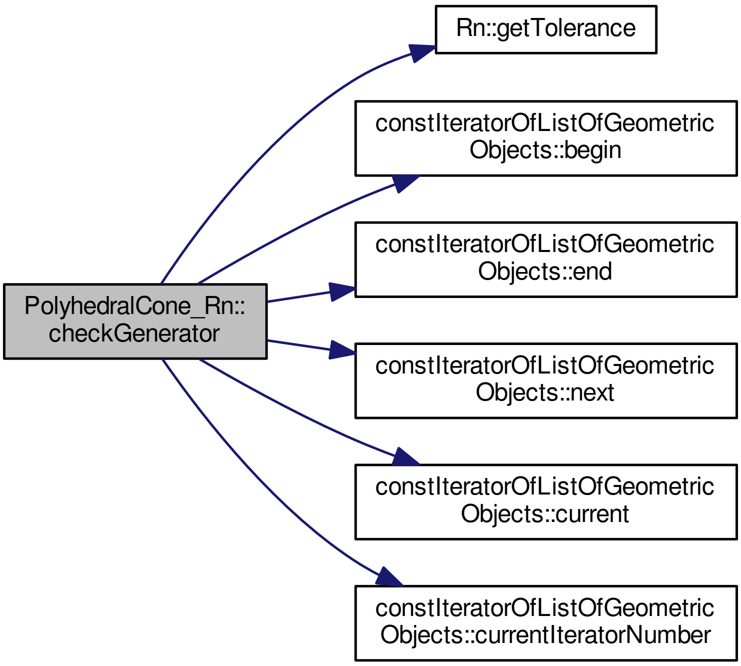

PolyhedralCone_Rn::
checkFacets
constIteratorOfListOfGeometric
Objects::begin
constIteratorOfListOfGeometric
Objects::end
constIteratorOfListOfGeometric
Objects::next
constIteratorOfListOfGeometric
Objects::current
constIteratorOfListOfGeometric
Objects::currentIteratorNumber

4.19.3.8 void PolyhedralCone_Rn::checkGenerator (unsigned int *vtxNumber,* std::ostream & *this_ostream*) const throw std::out_of_range) [inline]

Compute all distance from the current point to all half-spaces frontiers.

Definition at line 421 of file PolyhedralCone_Rn.h.

Here is the call graph for this function:


Rn::getTolerance
constIteratorOfListOfGeometricObjects::begin
constIteratorOfListOfGeometricObjects::end
PolyhedralCone_Rn::checkGenerator
constIteratorOfListOfGeometricObjects::next
constIteratorOfListOfGeometricObjects::current
constIteratorOfListOfGeometricObjects::currentIteratorNumber

4.19.3.9 bool PolyhedralCone_Rn::checkGenerators (const **listOfGeometricObjects**< boost::shared_ptr< **Generator_Rn** > > & *listGenA,* const **listOfGeometricObjects**< boost::shared_ptr< **HalfSpace_Rn** > > & *listHSB,* bool *check_all* = `false`) const `[inline]`

Check the number of facets per generator and make sure it is compliant with its current constraints. It must verify the following property :

$$\forall G=(g_1,...,g_n)\in\mathbb{R}^n,\exists(n-1)\bar{H_i}=\left\{x:\sum_{j=1}^{n}a_{ij}x_j\geq 0\right\}such\,as\,G\in\bar{H_i},i\in\{1,...,n-1\}$$

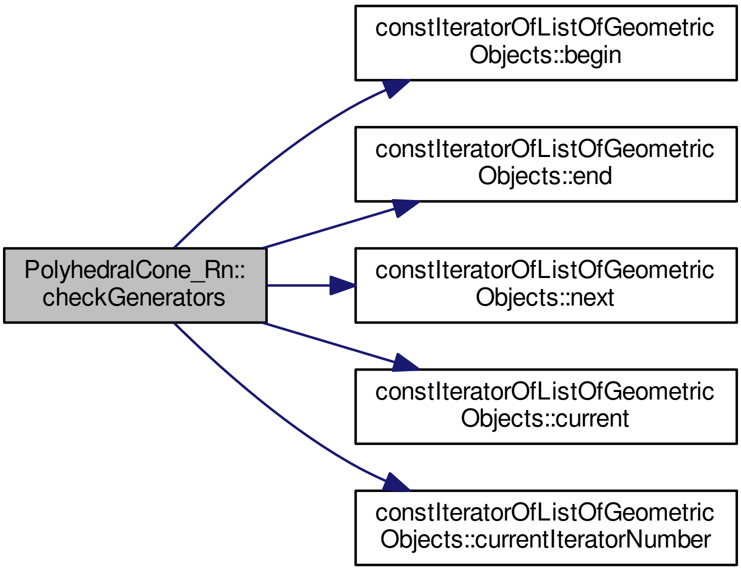
.

Returns

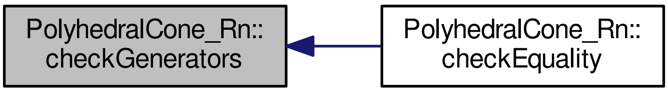
true if everything's OK, false otherwise.

Definition at line 459 of file PolyhedralCone_Rn.h.

Here is the call graph for this function:


PolyhedralCone_Rn::
checkGenerators
constIteratorOfListOfGeometric
Objects::begin
constIteratorOfListOfGeometric
Objects::end
constIteratorOfListOfGeometric
Objects::next
constIteratorOfListOfGeometric
Objects::current
constIteratorOfListOfGeometric
Objects::currentIteratorNumber

Here is the caller graph for this function:



4.19.3.10 bool PolyhedralCone_Rn::checkNeighbours (const boost::shared_ptr< **Generator_Rn** > & *V1,* const boost::shared_ptr< **Generator_Rn** > & *V2,* std::vector< boost::shared_ptr< **HalfSpace_Rn** > > & *commonFacets,* const std::set< boost::shared_ptr< **HalfSpace_Rn** > > & *listOfRedundantHS*) const throw std::invalid_argument) `[inline]`

Check for polytopes that vertices share *(n-1)* facets. For polyhedral cones, it must check that vectors share *(n-2)* facets.

Parameters

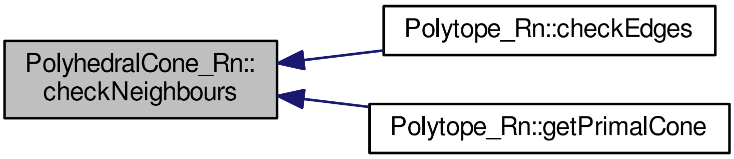
V1	the first vertex to check
V2	the second vertex to check
commonFacets	the set of common facets between V1 and V2
listOf↩ RedundantHS	the set of half-spaces that will not be taken into account to compute commonFacets

Returns

false if they are not neighbours, true otherwise with the list of common facets.

Definition at line 212 of file PolyhedralCone_Rn.h.

Here is the caller graph for this function:



4.19.3.11 bool PolyhedralCone_Rn::checkNeighbours (const boost::shared_ptr< Generator_Rn > & *V1*, const boost::shared_ptr< Generator_Rn > & *V2*, std::vector< boost::shared_ptr< HalfSpace_Rn > > & *commonFacets*, unsigned int *topologicalCode* = 1) const throw std::invalid_argument) [inline]

Check for polytopes that vertices share *(n-1)* facets. For polyhedral cones, it must check that vectors share *(n-2)* facets.

Parameters

V1	the first vertex to check
V2	the second vertex to check
commonFacets	the set of common facets between V1 and V2
topologicalCode	model the level of neighborhood: 1 for an edge, ..., (n-1) for a facet in a n-dimensional space

Returns

false if they are not neighbours, true otherwise with the list of common facets.

Definition at line 243 of file PolyhedralCone_Rn.h.

4.19.3.12 bool PolyhedralCone_Rn::checkNeighbours (const boost::shared_ptr< Generator_Rn > & *V1*, const boost::shared_ptr< Generator_Rn > & *V2*, std::vector< HalfSpace_Rn $*$ > & *commonFacets*, unsigned int *topologicalCode* = 1) const throw std::invalid_argument) [inline]

Check for polytopes that vertices share *(n-1)* facets. For polyhedral cones, it must check that vectors share *(n-2)* facets.

Parameters

V1	the first vertex to check
V2	the second vertex to check
commonFacets	the list of common facets between V1 and V2
topologicalCode	model the level of neighborhood: 1 for an edge, ..., (n-1) for a facet in a n-dimensional space

Returns

false if they are not neighbours, true otherwise with the list of common facets.

Definition at line 270 of file PolyhedralCone_Rn.h.

4.19.3.13 bool PolyhedralCone_Rn::checkNeighbours (const boost::shared_ptr< Generator_Rn_SD > & *genIn,* const boost::shared_ptr< Generator_Rn_SD > & *genOut,* std::vector< unsigned int > & *commonFacets*) [inline]

Check for polytopes that vertices share *(n-1)* facets. For polyhedral cones, it must check that vectors share *(n-2)* facets.

Parameters

genIn	the first vertex to check
genOut	the second vertex to check
commonFacets	the list of common facets indices between genIn and genOut

Definition at line 296 of file PolyhedralCone_Rn.h.

4.19.3.14 bool PolyhedralCone_Rn::checkNeighboursWithHSnumbers (const boost::shared_ptr< Generator_Rn > & *V1,* const boost::shared_ptr< Generator_Rn > & *V2,* std::vector< HalfSpace_Rn ∗ > & *commonFacets,* const std::set< boost::shared_ptr< HalfSpace_Rn > > & *listOfRedundantHS,* unsigned int *topologicalCode =* 1) const throw std::invalid_argument) [inline]

Check for polytopes that vertices share *(n-1)* facets. For polyhedral cones, it must check that vectors share *(n-2)* facets.

Parameters

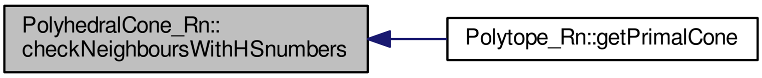
V1	the first vertex to check
V2	the second vertex to check
commonFacets	the list of common facets between V1 and V2
listOf← RedundantHS	the list of facets to ignore when we process V1 and V2
topologicalCode	model the level of neighborhood: 1 for an edge, ..., (n-1) for a facet in a n-dimensional space

Returns

false if they are not neighbours, true otherwise with the list of common facets.

Definition at line 318 of file PolyhedralCone_Rn.h.

Here is the caller graph for this function:



4.19.3.15 HalfSpace_Rn::State PolyhedralCone_Rn::checkPoint (const Point_Rn & *thisPoint*) const

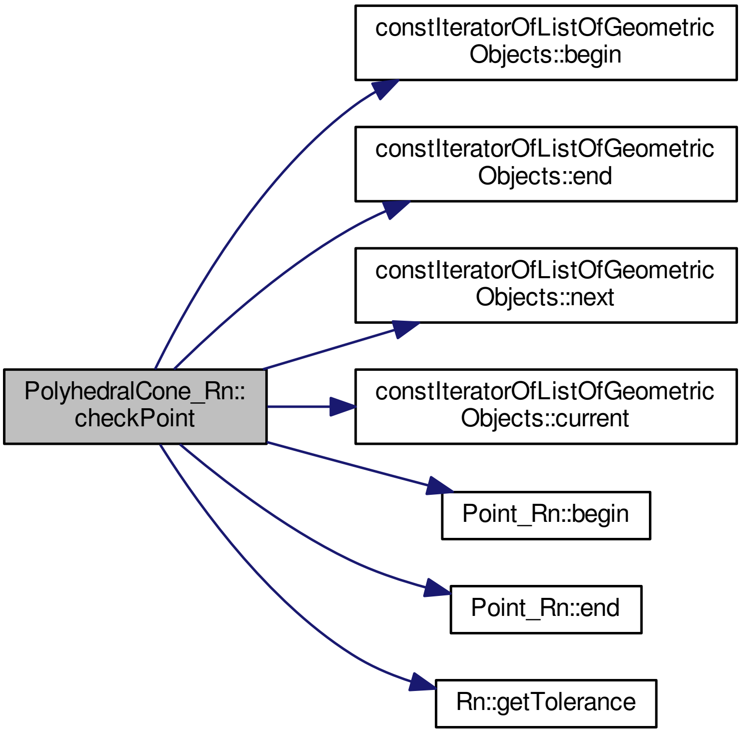
Check a point state against the whole polyhedron.

Returns

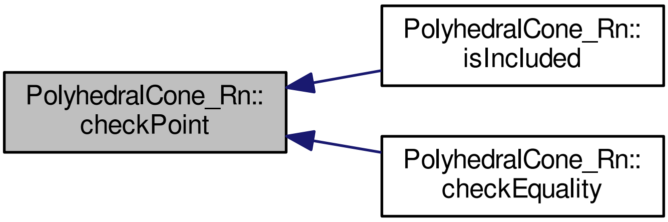
HalfSpace_Rn::OUT, HalfSpace_Rn::IN or HalfSpace_Rn::ON.

Definition at line 29 of file PolyhedralCone_Rn.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



4.19.3.16 **HalfSpace_Rn::State** PolyhedralCone_Rn::checkPoint (const boost::shared_ptr< **Generator_Rn** > & *point,* const boost::shared_ptr< **HalfSpace_Rn** > & *halfSpace,* double *halfSpaceNorm*) const

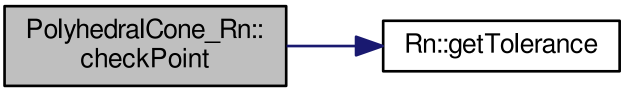
Check a point state against a half-space.

Returns

HalfSpace_Rn::OUT, HalfSpace_Rn::IN or HalfSpace_Rn::ON.

Definition at line 47 of file PolyhedralCone_Rn.cpp.

Here is the call graph for this function:



4.19.3.17 virtual bool PolyhedralCone_Rn::checkTopologyAndGeometry ( ) const `[inline]`, `[virtual]`

As stated by **Komei Fukuda** *"the complexity of the polyhedral verification problem is unknown. Is it in P or in coNP-complete?"* So only 3 verifications are made in R^n :

- check all the generators are inside the H-representation
- check that all generators have at least *(n-1)* facets in the case of a polyhedral cone (*n* for a polytope)
- check that all facets have at least *(n-1)* generators in the case of a polyhedral cone (*n* for a polytope).

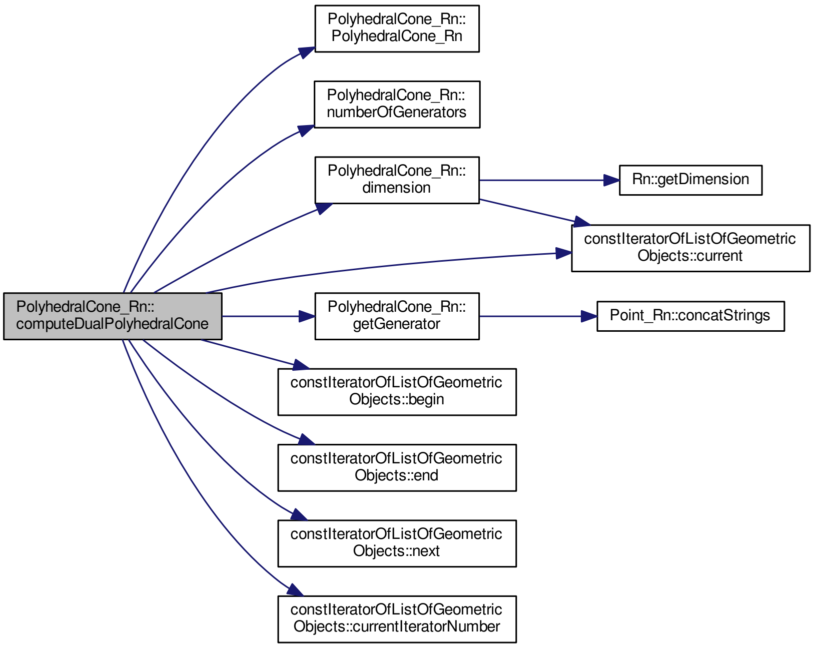
Definition at line 390 of file PolyhedralCone_Rn.h.

4.19.3.18 boost::shared_ptr< PolyhedralCone_Rn > PolyhedralCone_Rn::computeDualPolyhedralCone ( ) const

Build the dual polyhedral cone of the current one whose edge are orthogonal to the primal and vice-versa.

Definition at line 266 of file PolyhedralCone_Rn.cpp.

Here is the call graph for this function:


PolyhedralCone_Rn:: computeDualPolyhedralCone
PolyhedralCone_Rn:: PolyhedralCone_Rn
PolyhedralCone_Rn:: numberOfGenerators
PolyhedralCone_Rn:: dimension
Rn::getDimension
constIteratorOfListOfGeometric Objects::current
PolyhedralCone_Rn:: getGenerator
Point_Rn::concatStrings
constIteratorOfListOfGeometric Objects::begin
constIteratorOfListOfGeometric Objects::end
constIteratorOfListOfGeometric Objects::next
constIteratorOfListOfGeometric Objects::currentIteratorNumber

4.19.3.19 void PolyhedralCone_Rn::computeMinkowskiSum (const boost::shared_ptr< PolyhedralCone_Rn > & *A*, const boost::shared_ptr< PolyhedralCone_Rn > & *B*) [virtual]

Compute the Minkowski sum of two polyhedral cones.

Definition at line 261 of file PolyhedralCone_Rn.cpp.

4.19.3.20 virtual void PolyhedralCone_Rn::createBoundingBox (double) [inline], [virtual]

At the moment this function is useless in the case of polyhedral cones.

Reimplemented in Polytope_Rn.

Definition at line 668 of file PolyhedralCone_Rn.h.

4.19.3.21 virtual void PolyhedralCone_Rn::createBoundingSimplex (double) [inline], [virtual]

At the moment this function is useless in the case of polyhedral cones.

Reimplemented in Polytope_Rn.

Definition at line 671 of file PolyhedralCone_Rn.h.

4.19.3.22 void PolyhedralCone_Rn::createTruncatedGenerator (const boost::shared_ptr< Generator_Rn_SD > & *y*, const boost::shared_ptr< Generator_Rn_SD > & *z*, boost::shared_ptr< Generator_Rn_SD > *newG*, double *ay*, double *az*, double *b* = 0.) const [virtual]

Create the intersection edge in the truncating algorithm. It is defined by the intersection between a 2-face and a hyperplane, i.e. a (n-1)-face. The new egde is given by this formula where H is the current half space :

$$newG = \frac{\langle a,z\rangle}{\langle a,z-y\rangle}y - \frac{\langle a,y\rangle}{\langle a,z-y\rangle}z, H = \left\{ x : \langle a,x\rangle = \sum_{j=1}^{n} a_j x_j \geq 0 \right\}$$

.

Parameters

y	The generator outside the current half-space $\langle a,y\rangle < 0$
z	The generator inside the current half-space $\langle a,z\rangle > 0$
newG	The new generator is the intersection between the 2-face created by y and z, and the current halfspace hyperplane
ay	Scalar product $\langle a,y\rangle$
az	Scalar product $\langle a,z\rangle$
b	Half-space constant, null in the polyhedral cone case.

Reimplemented in Polytope_Rn.

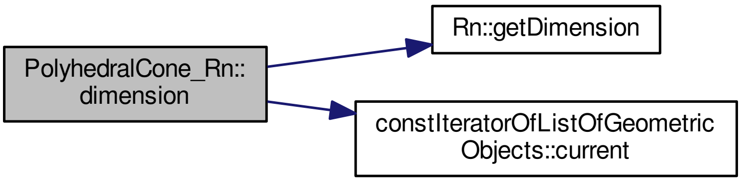
Definition at line 323 of file PolyhedralCone_Rn.cpp.

4.19.3.23 virtual unsigned int PolyhedralCone_Rn::dimension ( ) const [inline], [virtual]

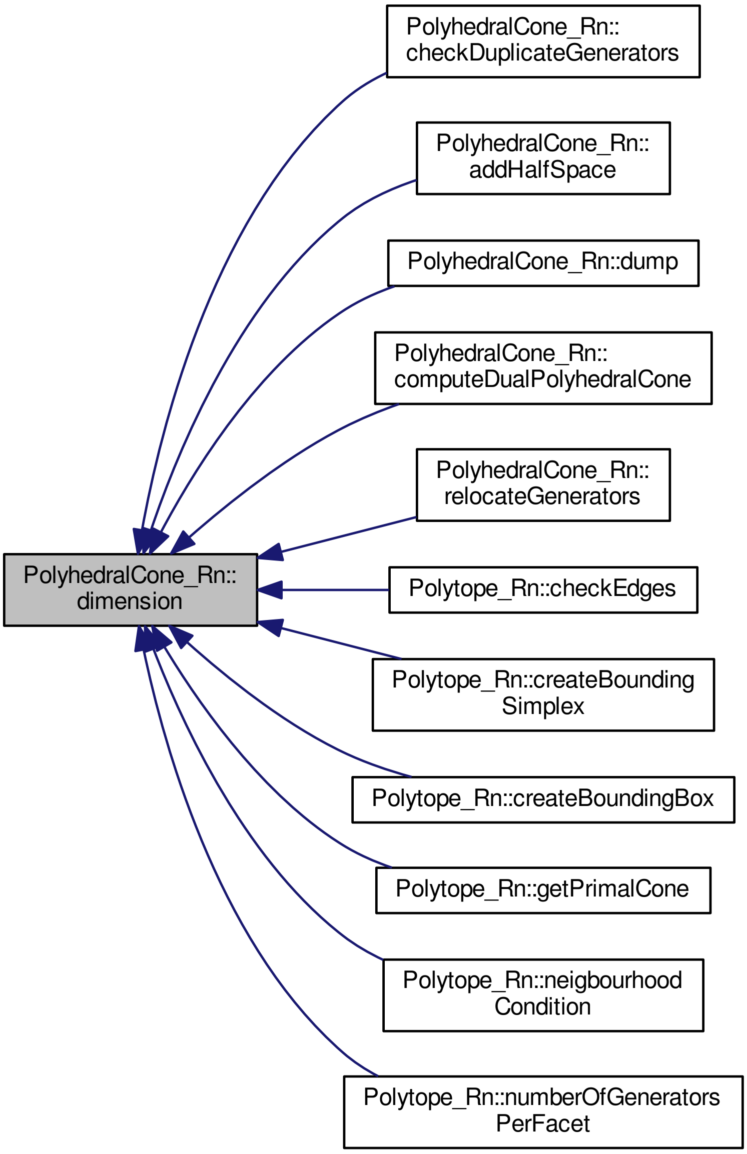
Return the space dimension.

Definition at line 63 of file PolyhedralCone_Rn.h.

Here is the call graph for this function:



Here is the caller graph for this function:

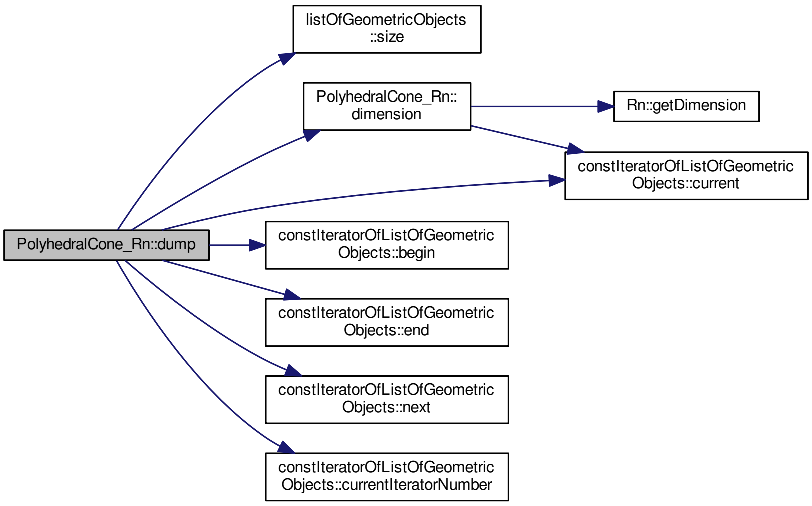


4.19.3.24 void PolyhedralCone_Rn::dump (std::ostream & *this_ostream*) const

Dump the polyhedral structure on std::cout.

Definition at line 204 of file PolyhedralCone_Rn.cpp.

Here is the call graph for this function:


listOfGeometricObjects ::size
PolyhedralCone_Rn:: dimension
Rn::getDimension
constIteratorOfListOfGeometric Objects::current
PolyhedralCone_Rn::dump
constIteratorOfListOfGeometric Objects::begin
constIteratorOfListOfGeometric Objects::end
constIteratorOfListOfGeometric Objects::next
constIteratorOfListOfGeometric Objects::currentIteratorNumber

4.19.3.25 void PolyhedralCone_Rn::fillNeighbourMatrix (std::vector< std::vector< unsigned int > > & *neighboursA*, unsigned int *topologicalCode* = 1) const throw std::out_of_range) [inline]

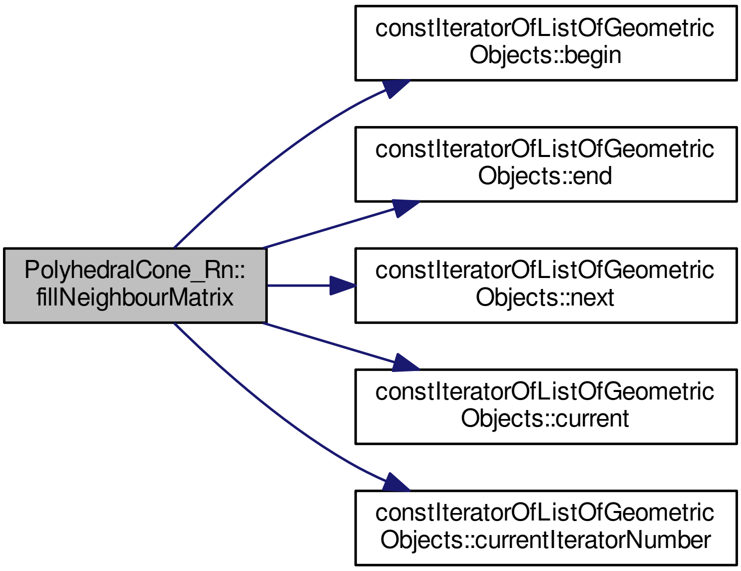
Compute and store all neighborhood relations between generators.

Parameters

neighboursA	a sparse matrix where each line models a generator
topologicalCode	model the level of neighborhood: 1 for an edge, ..., (n-1) for a facet in a n-dimensional space

Definition at line 345 of file PolyhedralCone_Rn.h.

Here is the call graph for this function:

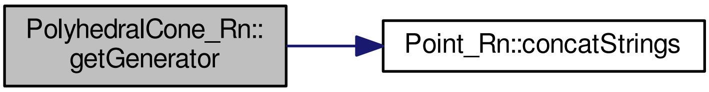

constIteratorOfListOfGeometric Objects::begin
constIteratorOfListOfGeometric Objects::end
PolyhedralCone_Rn:: fillNeighbourMatrix
constIteratorOfListOfGeometric Objects::next
constIteratorOfListOfGeometric Objects::current
constIteratorOfListOfGeometric Objects::currentIteratorNumber

4.19.3.26 const boost::shared_ptr< Generator_Rn > & PolyhedralCone_Rn::getGenerator (unsigned int *i*) const throw std::out_of_range)

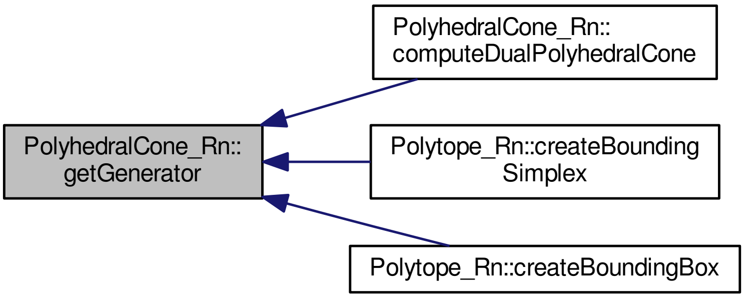
Return the i-th generator.

Definition at line 171 of file PolyhedralCone_Rn.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

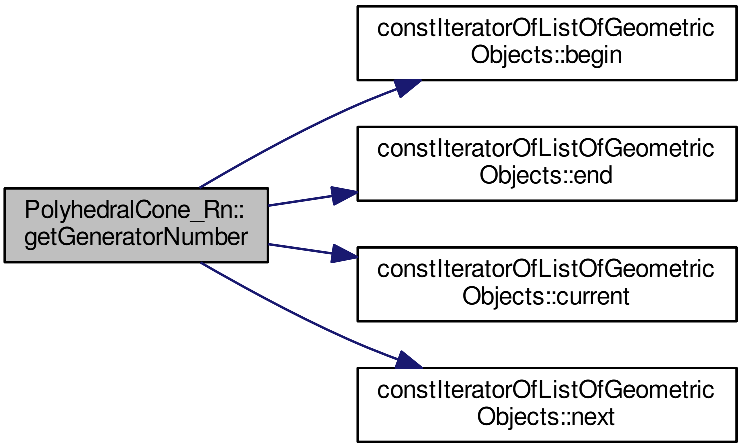

PolyhedralCone_Rn::
computeDualPolyhedralCone
PolyhedralCone_Rn::
getGenerator
Polytope_Rn::createBounding
Simplex
Polytope_Rn::createBoundingBox

4.19.3.27 unsigned int PolyhedralCone_Rn::getGeneratorNumber (boost::shared_ptr< Generator_Rn > *G*) const throw std::out_of_range, std::invalid_argument)

For a given generator, return its list index.

Definition at line 180 of file PolyhedralCone_Rn.cpp.

Here is the call graph for this function:

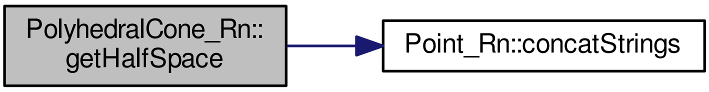

constIteratorOfListOfGeometric
Objects::begin
constIteratorOfListOfGeometric
Objects::end
PolyhedralCone_Rn::
getGeneratorNumber
constIteratorOfListOfGeometric
Objects::current
constIteratorOfListOfGeometric
Objects::next

4.19.3.28 const boost::shared_ptr< HalfSpace_Rn > & PolyhedralCone_Rn::getHalfSpace (unsigned int *i*) const throw std::out_of_range)

Return the i-th generator.

Definition at line 159 of file PolyhedralCone_Rn.cpp.

Here is the call graph for this function:



4.19.3.29 unsigned int PolyhedralCone_Rn::getHalfSpaceNumber (const boost::shared_ptr< **HalfSpace_Rn** > & *F*) const throw std::out_of_range) `[inline]`

For a given half-space, return its list index.

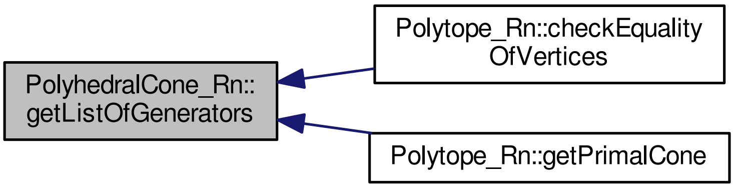
Definition at line 107 of file PolyhedralCone_Rn.h.

4.19.3.30 const **listOfGeometricObjects**< boost::shared_ptr<**Generator_Rn**> >& PolyhedralCone_Rn::getListOfGenerators ( ) const `[inline]`

Return the list of generators.

Definition at line 145 of file PolyhedralCone_Rn.h.

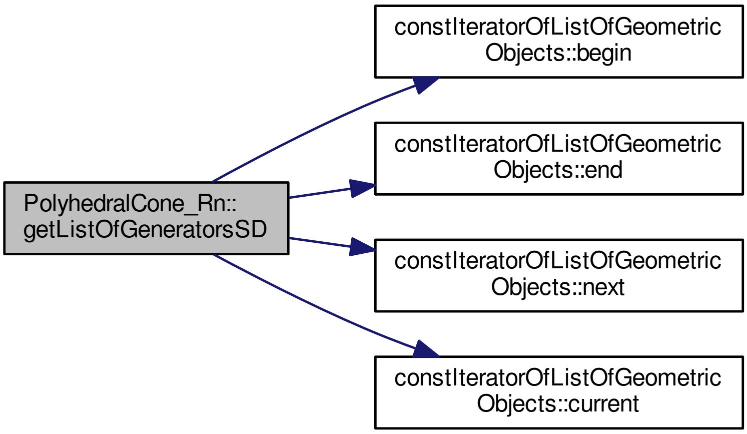
Here is the caller graph for this function:



4.19.3.31 unsigned int PolyhedralCone_Rn::getListOfGeneratorsSD (std::vector< boost::shared_ptr< **Generator_Rn_SD** > > & *currentListOfGeneratorsSD*) `[inline]`

Definition at line 147 of file PolyhedralCone_Rn.h.

Here is the call graph for this function:

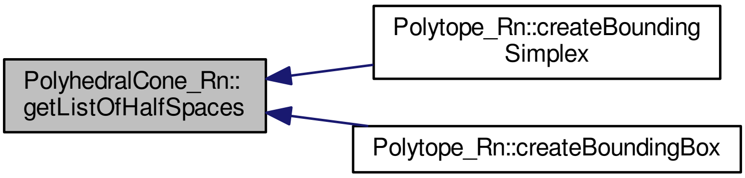

PolyhedralCone_Rn::
getListOfGeneratorsSD
constIteratorOfListOfGeometric
Objects::begin
constIteratorOfListOfGeometric
Objects::end
constIteratorOfListOfGeometric
Objects::next
constIteratorOfListOfGeometric
Objects::current

4.19.3.32 listOfGeometricObjects< boost::shared_ptr<HalfSpace_Rn> >& PolyhedralCone_Rn::getListOfHalfSpaces ( ) [inline]

Return the list of half-spaces.

Definition at line 124 of file PolyhedralCone_Rn.h.

Here is the caller graph for this function:



4.19.3.33 const listOfGeometricObjects< boost::shared_ptr<HalfSpace_Rn> >& PolyhedralCone_Rn::getListOfHalfSpaces ( ) const [inline]

Return the list of half-spaces.

Definition at line 127 of file PolyhedralCone_Rn.h.

4.19.3.34 virtual bool PolyhedralCone_Rn::isBounded ( ) const [inline], [virtual]

Tell whether this polyhedron is bounded or not, polyhedral cones are not.

Reimplemented in Polytope_Rn.

Definition at line 79 of file PolyhedralCone_Rn.h.

4.19.3.35 bool PolyhedralCone_Rn::isIncluded (const boost::shared_ptr< PolyhedralCone_Rn > & *B*) const

Test whether the current polytope V-description is inside the polytope B H-description.

Parameters

B	The H-polytope

Returns

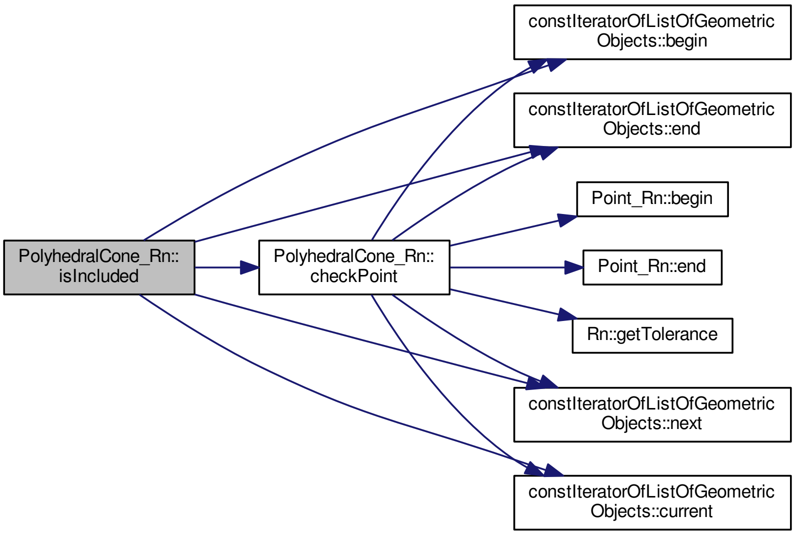
true if

$$this \subset B$$

, false otherwise.

Definition at line 340 of file PolyhedralCone_Rn.cpp.

Here is the call graph for this function:



4.19.3.36 void PolyhedralCone_Rn::negate ( ) [inline]

Compute the symmetrical polyhedral cone.

Definition at line 639 of file PolyhedralCone_Rn.h.

4.19.3.37 virtual unsigned int PolyhedralCone_Rn::neigbourhoodCondition ( ) const [inline], [virtual]

Two edges are neighbours in a polyhedral cone <=> they share at least (n-2) facets.

Reimplemented in Polytope_Rn.

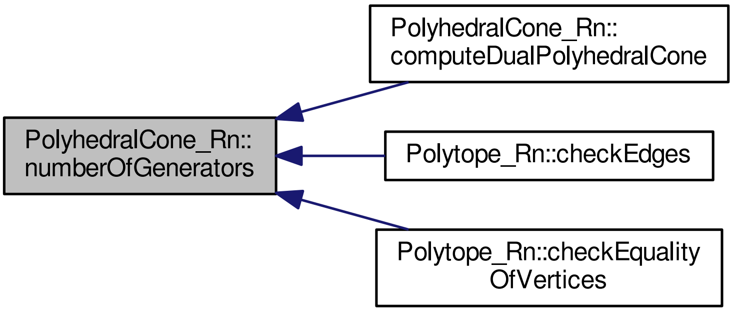
Definition at line 82 of file PolyhedralCone_Rn.h.

4.19.3.38 unsigned int PolyhedralCone_Rn::numberOfGenerators ( ) const [inline]

Give the total number of generators.

Definition at line 133 of file PolyhedralCone_Rn.h.

Here is the caller graph for this function:



4.19.3.39 virtual unsigned int PolyhedralCone_Rn::numberOfGeneratorsPerFacet ( ) const [inline], [virtual]

Each facet in a polyhedral cone has got (n-1) edges.

Reimplemented in Polytope_Rn.

Definition at line 85 of file PolyhedralCone_Rn.h.

4.19.3.40 unsigned int PolyhedralCone_Rn::numberOfHalfSpaces ( ) const [inline]

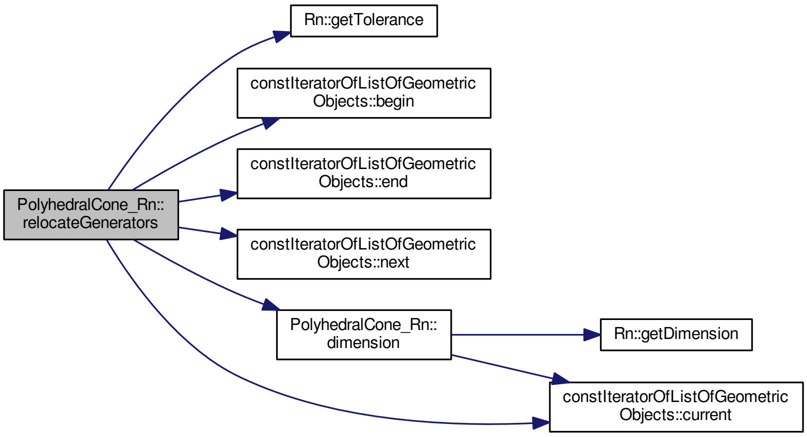
Get the total number of half-spaces.

Definition at line 93 of file PolyhedralCone_Rn.h.

4.19.3.41 void PolyhedralCone_Rn::relocateGenerators ( )

Definition at line 356 of file PolyhedralCone_Rn.cpp.

Here is the call graph for this function:



4.19.3.42 void PolyhedralCone_Rn::removeGenerator (unsigned int *j*) [inline]

Remove the generator number j from its list.

Definition at line 512 of file PolyhedralCone_Rn.h.

4.19.3.43 void PolyhedralCone_Rn::removeHalfSpace (unsigned int *j*) [inline]

Remove the half-space number j from its list.

Definition at line 99 of file PolyhedralCone_Rn.h.

4.19.3.44 void PolyhedralCone_Rn::removeHalfSpaces (const std::set< boost::shared_ptr< **HalfSpace_Rn** > > & *setOfRedundantHS*) [inline]

Remove the half-space number j from its list.

Definition at line 102 of file PolyhedralCone_Rn.h.

4.19.3.45 void PolyhedralCone_Rn::reset ( ) [inline]

Remove all half-spaces and generators.

Definition at line 88 of file PolyhedralCone_Rn.h.

4.19.3.46 void PolyhedralCone_Rn::setListOfGenerators (const **listOfGeometricObjects**< boost::shared_ptr< **Generator_Rn** > > & *gnList*) [inline]

Set a new list of generators.

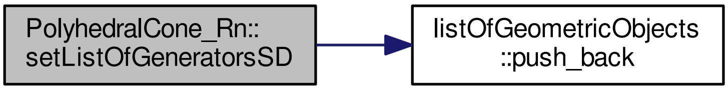
Definition at line 201 of file PolyhedralCone_Rn.h.

4.19.3.47 void PolyhedralCone_Rn::setListOfGeneratorsSD (const std::vector< boost::shared_ptr< Generator_Rn_SD > > & *gnList*) [inline]

Set a new list of generators. The list of half-spaces should have been previously set.

Definition at line 165 of file PolyhedralCone_Rn.h.

Here is the call graph for this function:



4.19.4 Friends And Related Function Documentation

4.19.4.1 friend class constIteratorOfListOfHalfSpaces [friend]

Definition at line 47 of file PolyhedralCone_Rn.h.

4.19.4.2 friend class lexIteratorOfListOfHalfSpaces [friend]

Definition at line 46 of file PolyhedralCone_Rn.h.

4.19.5 Member Data Documentation

4.19.5.1 listOfGeometricObjects< boost::shared_ptr<Generator_Rn> > PolyhedralCone_Rn::_listOfGenerators [protected]

The convex hull of connected points.

Definition at line 677 of file PolyhedralCone_Rn.h.

4.19.5.2 listOfGeometricObjects< boost::shared_ptr<HalfSpace_Rn> > PolyhedralCone_Rn::_listOfHalfSpaces [protected]

The list of half-spaces defining the polytope.

Definition at line 675 of file PolyhedralCone_Rn.h.

The documentation for this class was generated from the following files:

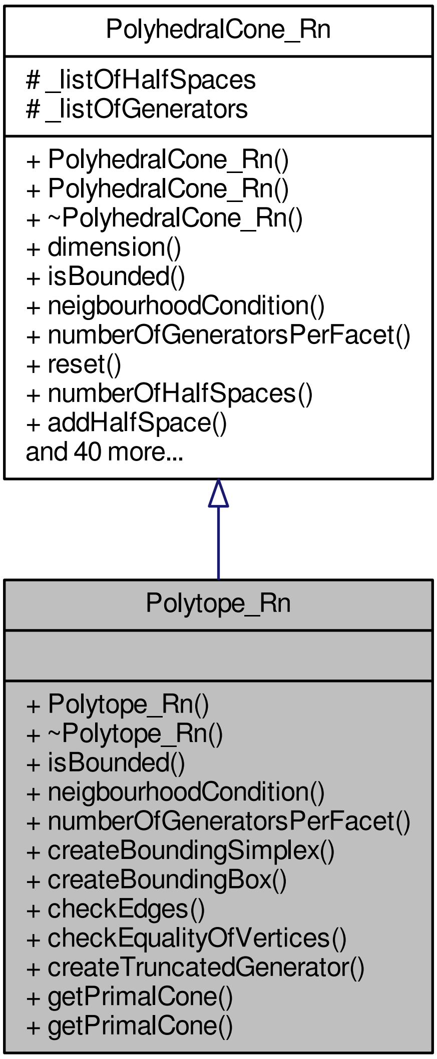
- /home/vindelos/CPP/I2M/politopix/trunk/PolyhedralCone_Rn.h
- /home/vindelos/CPP/I2M/politopix/trunk/PolyhedralCone_Rn.cpp

4.20 Polytope_Rn Class Reference

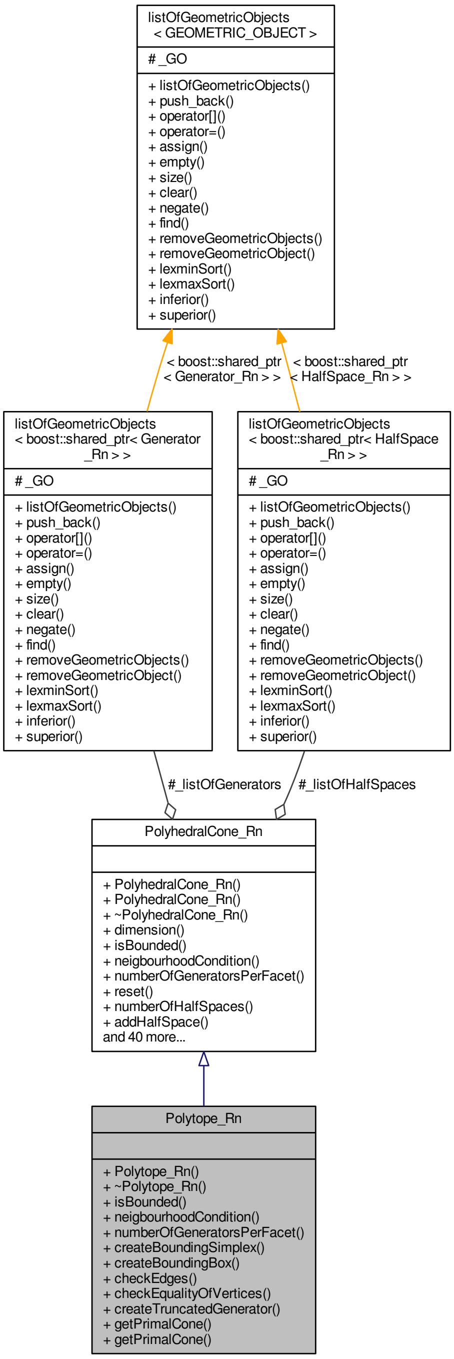
Model a polytope using its two equivalent definitions : the convex hull and the half-space intersection.

```
#include <Polytope_Rn.h>
```

Inheritance diagram for Polytope_Rn:



Collaboration diagram for Polytope_Rn:



Public Member Functions

- Polytope_Rn ()

  Constructor for polytopes i.e. bounded convex polyhedra.
- virtual ∼Polytope_Rn ()

  Destructor.
- virtual bool isBounded () const

Tell whether this polyhedron is bounded or not, polytopes are bounded.

- virtual unsigned int neigbourhoodCondition () const

  Two vertices are neighbours in a polytope <=> they share at least (n-1) facets.

- virtual unsigned int numberOfGeneratorsPerFacet () const

  Each facet in a polytope has got n vertices.

- virtual void createBoundingSimplex (double M)

  Initialize the truncating algorithm building a M-sized simplex around the polytope.

- virtual void createBoundingBox (double M)

  Initialize the truncating algorithm building a M-sized bounding box around the polytope.

- virtual bool checkEdges () const

  Always true in the polyhedral cone case.

- bool checkEqualityOfVertices (const boost::shared_ptr< Polytope_Rn > &B, bool printOnScreen=false) const

  Check whether two V-polytopes are identical Check whether the sets of vertices of A and B are equal.

- virtual void createTruncatedGenerator (const boost::shared_ptr< Generator_Rn_SD > &in, const boost::shared_ptr< Generator_Rn_SD > &out, boost::shared_ptr< Generator_Rn_SD > newV, double ay, double az, double b=0.) const

  This is the intersection vertex in the truncating algorithm, defined by the intersection between an edge and an hyperplane.

- boost::shared_ptr
  < PolyhedralCone_Rn > getPrimalCone (unsigned int i) const throw (std::out_of_range)
- boost::shared_ptr
  < PolyhedralCone_Rn > getPrimalCone (const boost::shared_ptr< Generator_Rn > &vx) const

  Return the primal cone C(vx) of the polytope A associated to the vertex vx.

Additional Inherited Members

4.20.1 Detailed Description

Model a polytope using its two equivalent definitions : the convex hull and the half-space intersection.

Definition at line 34 of file Polytope_Rn.h.

4.20.2 Constructor & Destructor Documentation

4.20.2.1 Polytope_Rn::Polytope_Rn ( ) [inline]

Constructor for polytopes i.e. bounded convex polyhedra.

Definition at line 38 of file Polytope_Rn.h.

4.20.2.2 virtual Polytope_Rn::∼Polytope_Rn ( ) [inline], [virtual]

Destructor.

Definition at line 41 of file Polytope_Rn.h.

4.20.3 Member Function Documentation

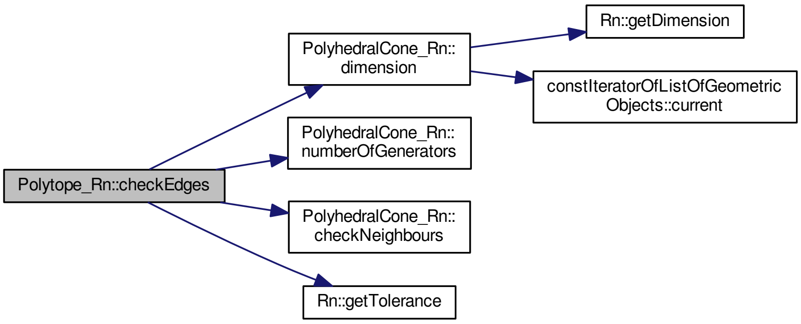
4.20.3.1 bool Polytope_Rn::checkEdges ( ) const [virtual]

Always true in the polyhedral cone case.

Reimplemented from PolyhedralCone_Rn.

Definition at line 115 of file Polytope_Rn.cpp.

Here is the call graph for this function:


Polytope_Rn::checkEdges
PolyhedralCone_Rn::dimension
PolyhedralCone_Rn::numberOfGenerators
PolyhedralCone_Rn::checkNeighbours
Rn::getTolerance
Rn::getDimension
constIteratorOfListOfGeometricObjects::current

4.20.3.2 bool Polytope_Rn::checkEqualityOfVertices (const boost::shared_ptr< Polytope_Rn > & *B*, bool *printOnScreen* = `false`) const

Check whether two V-polytopes are identical Check whether the sets of vertices of A and B are equal.

Parameters

A	The V-polytope

Returns

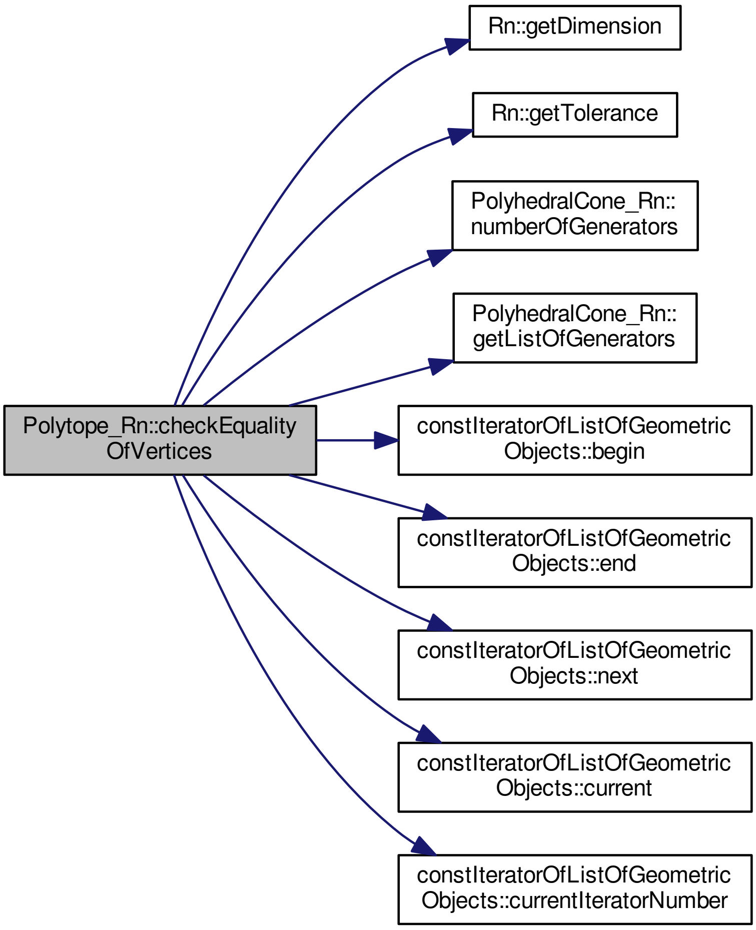
true if

$$\mathcal{V}_A = \mathcal{V}_B$$

, false otherwise.

Definition at line 324 of file Polytope_Rn.cpp.

Here is the call graph for this function:


Rn::getDimension
Rn::getTolerance
PolyhedralCone_Rn::numberOfGenerators
PolyhedralCone_Rn::getListOfGenerators
Polytope_Rn::checkEqualityOfVertices
constIteratorOfListOfGeometricObjects::begin
constIteratorOfListOfGeometricObjects::end
constIteratorOfListOfGeometricObjects::next
constIteratorOfListOfGeometricObjects::current
constIteratorOfListOfGeometricObjects::currentIteratorNumber

4.20.3.3 void Polytope_Rn::createBoundingBox (double *M*) [virtual]

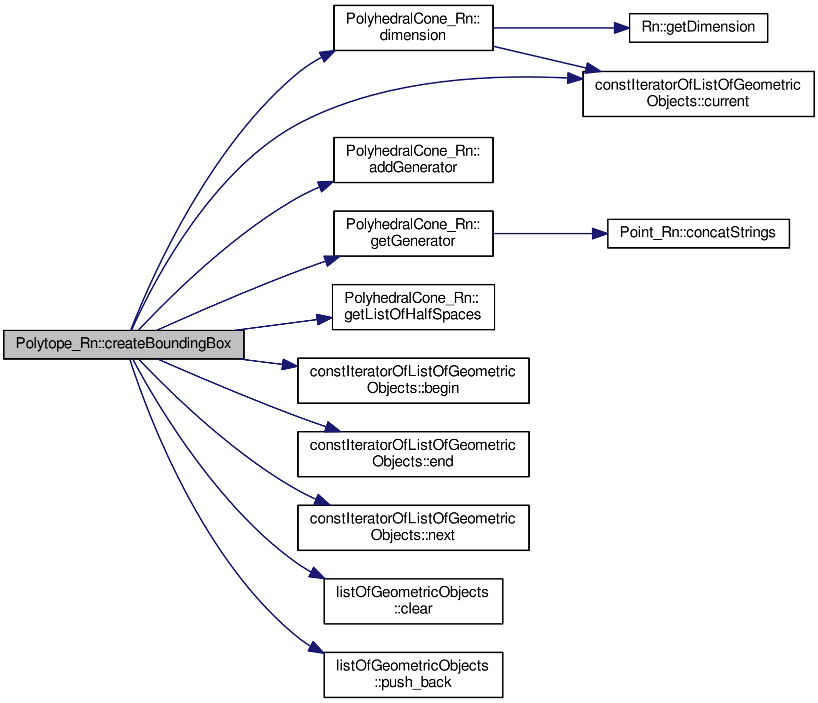
Initialize the truncating algorithm building a M-sized bounding box around the polytope.

When only the polytope facets are given, include it in a huge cube that we are going to truncate.

Reimplemented from PolyhedralCone_Rn.

Definition at line 237 of file Polytope_Rn.cpp.

Here is the call graph for this function:


Polytope_Rn::createBoundingBox
PolyhedralCone_Rn::dimension
Rn::getDimension
constIteratorOfListOfGeometricObjects::current
PolyhedralCone_Rn::addGenerator
PolyhedralCone_Rn::getGenerator
Point_Rn::concatStrings
PolyhedralCone_Rn::getListOfHalfSpaces
constIteratorOfListOfGeometricObjects::begin
constIteratorOfListOfGeometricObjects::end
constIteratorOfListOfGeometricObjects::next
listOfGeometricObjects::clear
listOfGeometricObjects::push_back

4.20.3.4 void Polytope_Rn::createBoundingSimplex (double *M*) [virtual]

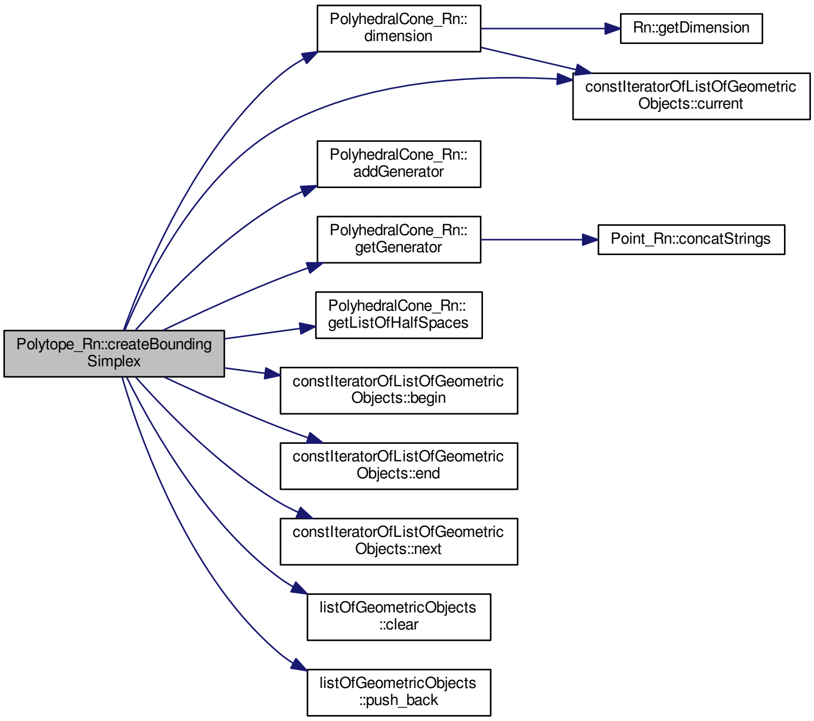
Initialize the truncating algorithm building a M-sized simplex around the polytope.

When only the polytope facets are given, include it in a huge simplex that we are going to truncate.

Reimplemented from PolyhedralCone_Rn.

Definition at line 162 of file Polytope_Rn.cpp.

Here is the call graph for this function:


Polytope_Rn::createBounding Simplex
PolyhedralCone_Rn:: dimension
Rn::getDimension
constIteratorOfListOfGeometric Objects::current
PolyhedralCone_Rn:: addGenerator
PolyhedralCone_Rn:: getGenerator
Point_Rn::concatStrings
PolyhedralCone_Rn:: getListOfHalfSpaces
constIteratorOfListOfGeometric Objects::begin
constIteratorOfListOfGeometric Objects::end
constIteratorOfListOfGeometric Objects::next
listOfGeometricObjects ::clear
listOfGeometricObjects ::push_back

4.20.3.5 void Polytope_Rn::createTruncatedGenerator (const boost::shared_ptr< Generator_Rn_SD > & *in,* const boost::shared_ptr< Generator_Rn_SD > & *out,* boost::shared_ptr< Generator_Rn_SD > *newV,* double *ay,* double *az,* double *b* = `0.`) const [virtual]

This is the intersection vertex in the truncating algorithm, defined by the intersection between an edge and an hyperplane.

This is the intersection vertex in the truncating algorithm. It is defined by the intersection between an edge, i.e. a 1-face, and an hyperplane, i.e. a (n-1)-face.

Reimplemented from PolyhedralCone_Rn.

Definition at line 309 of file Polytope_Rn.cpp.

4.20.3.6 boost::shared_ptr< PolyhedralCone_Rn > Polytope_Rn::getPrimalCone (unsigned int *i*) const throw std::out_of_range)

Return the i-th primal cone C(i) of the polytope A. If A has k vertices then $A = \bigcap_{i=1}^{k} C(i)$ where $C(i) = \bigcap_{i=1}^{l} \bar{H}_i^+$

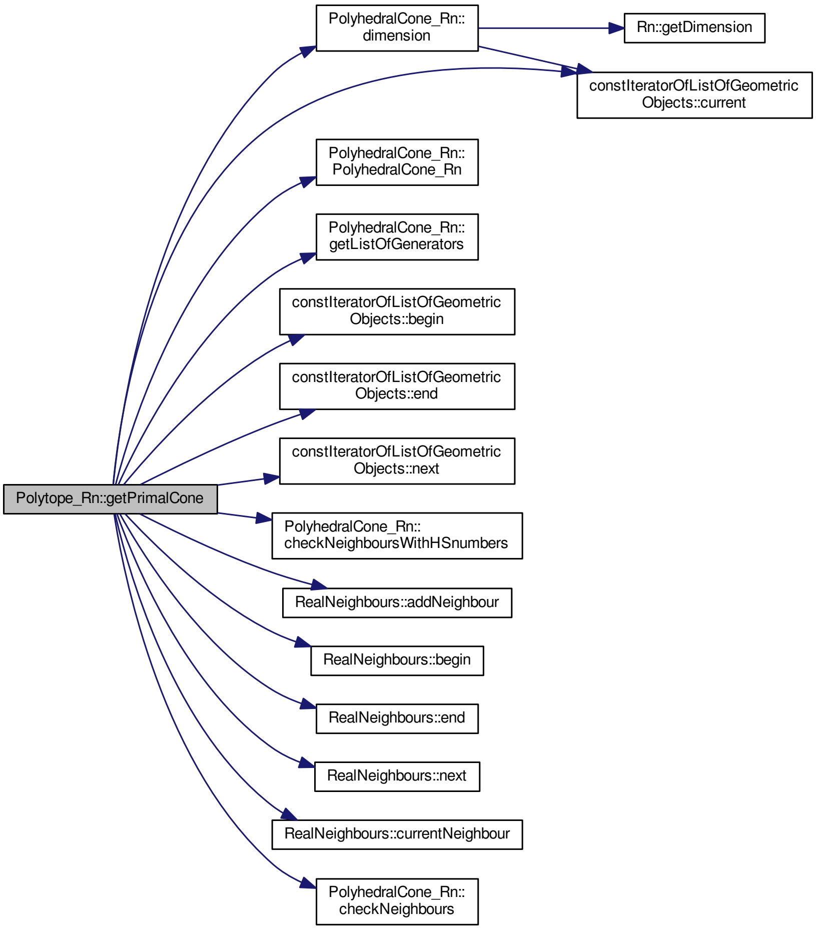
Definition at line 367 of file Polytope_Rn.cpp.

4.20.3.7 boost::shared_ptr< PolyhedralCone_Rn > Polytope_Rn::getPrimalCone (const boost::shared_ptr< Generator_Rn > & *vx*) const

Return the primal cone C(vx) of the polytope A associated to the vertex vx.

Definition at line 404 of file Polytope_Rn.cpp.

Here is the call graph for this function:


PolyhedralCone_Rn::dimension
Rn::getDimension
constIteratorOfListOfGeometricObjects::current
PolyhedralCone_Rn::PolyhedralCone_Rn
PolyhedralCone_Rn::getListOfGenerators
constIteratorOfListOfGeometricObjects::begin
constIteratorOfListOfGeometricObjects::end
constIteratorOfListOfGeometricObjects::next
Polytope_Rn::getPrimalCone
PolyhedralCone_Rn::checkNeighboursWithHSnumbers
RealNeighbours::addNeighbour
RealNeighbours::begin
RealNeighbours::end
RealNeighbours::next
RealNeighbours::currentNeighbour
PolyhedralCone_Rn::checkNeighbours

4.20.3.8 virtual bool Polytope_Rn::isBounded ( ) const [inline], [virtual]

Tell whether this polyhedron is bounded or not, polytopes are bounded.

Reimplemented from PolyhedralCone_Rn.

Definition at line 44 of file Polytope_Rn.h.

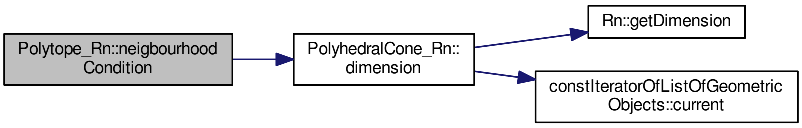
4.20.3.9 virtual unsigned int Polytope_Rn::neigbourhoodCondition ( ) const [inline], [virtual]

Two vertices are neighbours in a polytope <=> they share at least (n-1) facets.

Reimplemented from PolyhedralCone_Rn.

Definition at line 47 of file Polytope_Rn.h.

Here is the call graph for this function:


Polytope_Rn::neigbourhood Condition
PolyhedralCone_Rn:: dimension
Rn::getDimension
constIteratorOfListOfGeometric Objects::current

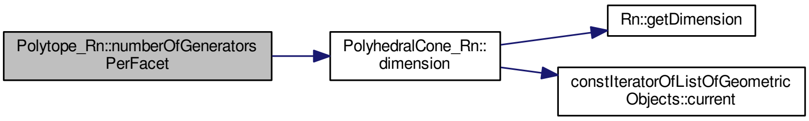
4.20.3.10 virtual unsigned int Polytope_Rn::numberOfGeneratorsPerFacet ( ) const [inline], [virtual]

Each facet in a polytope has got n vertices.

Reimplemented from PolyhedralCone_Rn.

Definition at line 50 of file Polytope_Rn.h.

Here is the call graph for this function:


Polytope_Rn::numberOfGenerators PerFacet
PolyhedralCone_Rn:: dimension
Rn::getDimension
constIteratorOfListOfGeometric Objects::current

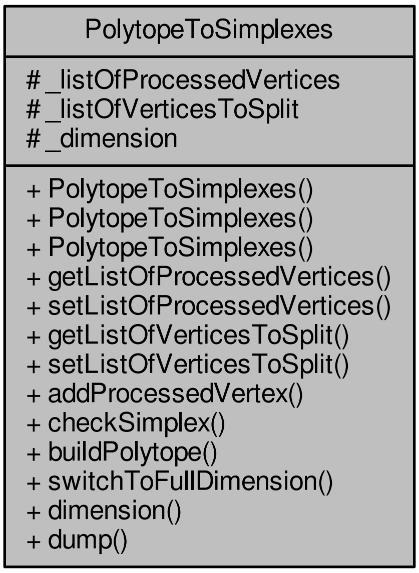
The documentation for this class was generated from the following files:

- /home/vindelos/CPP/I2M/politopix/trunk/Polytope_Rn.h
- /home/vindelos/CPP/I2M/politopix/trunk/Polytope_Rn.cpp

4.21 PolytopeToSimplexes Class Reference

```
#include <VolumeOfPolytopes_Rn.h>
```

Collaboration diagram for PolytopeToSimplexes:



Public Member Functions

- PolytopeToSimplexes (const std::vector< unsigned int > &splitPol, unsigned int dim)
- PolytopeToSimplexes (const std::vector< unsigned int > &vtx2plit, unsigned int apexNb, unsigned int dim)
- PolytopeToSimplexes (const std::vector< unsigned int > &vtx2plit, const std::vector< unsigned int > &prc↩Vtx, unsigned int apexNb, unsigned int dim)
- const std::vector< unsigned int > & getListOfProcessedVertices () const
- void setListOfProcessedVertices (const std::vector< unsigned int > &LPV)
- const std::vector< unsigned int > & getListOfVerticesToSplit () const
- void setListOfVerticesToSplit (const std::vector< unsigned int > &VTS)
- void addProcessedVertex (unsigned int ap)
- bool checkSimplex () const
- std::vector< unsigned int > buildPolytope () const
- void switchToFullDimension ()
- unsigned int dimension () const
- void dump (std::ostream &this_ostream, unsigned int shift=0) const

Protected Attributes

- std::vector< unsigned int > _listOfProcessedVertices

  The ordered list of vertices as we go down in smaller dimensions spaces.
- std::vector< unsigned int > _listOfVerticesToSplit

  The ordered list of vertices to be split in lower dimensions.
- unsigned int _dimension

  The current dimension space we work in.

4.21.1 Detailed Description

Definition at line 42 of file VolumeOfPolytopes_Rn.h.

4.21.2 Constructor & Destructor Documentation

4.21.2.1 PolytopeToSimplexes::PolytopeToSimplexes (const std::vector< unsigned int > & *splitPol,* unsigned int *dim*) [inline]

Definition at line 45 of file VolumeOfPolytopes_Rn.h.

4.21.2.2 PolytopeToSimplexes::PolytopeToSimplexes (const std::vector< unsigned int > & *vtx2plit,* unsigned int *apexNb,* unsigned int *dim*) [inline]

Definition at line 55 of file VolumeOfPolytopes_Rn.h.

4.21.2.3 PolytopeToSimplexes::PolytopeToSimplexes (const std::vector< unsigned int > & *vtx2plit,* const std::vector< unsigned int > & *prcVtx,* unsigned int *apexNb,* unsigned int *dim*) [inline]

Definition at line 61 of file VolumeOfPolytopes_Rn.h.

4.21.3 Member Function Documentation

4.21.3.1 void PolytopeToSimplexes::addProcessedVertex (unsigned int *ap*) [inline]

Definition at line 84 of file VolumeOfPolytopes_Rn.h.

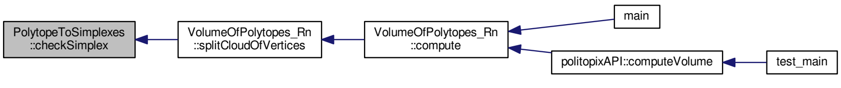
4.21.3.2 std::vector< unsigned int > PolytopeToSimplexes::buildPolytope ( ) const [inline]

Definition at line 94 of file VolumeOfPolytopes_Rn.h.

4.21.3.3 bool PolytopeToSimplexes::checkSimplex ( ) const [inline]

Definition at line 88 of file VolumeOfPolytopes_Rn.h.

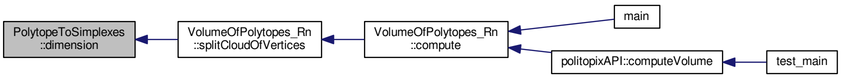
Here is the caller graph for this function:


PolytopeToSimplexes ::checkSimplex
VolumeOfPolytopes_Rn ::splitCloudOfVertices
VolumeOfPolytopes_Rn ::compute
main
politopixAPI::computeVolume
test_main

4.21.3.4 unsigned int PolytopeToSimplexes::dimension ( ) const [inline]

Definition at line 110 of file VolumeOfPolytopes_Rn.h.

Here is the caller graph for this function:


PolytopeToSimplexes ::dimension
VolumeOfPolytopes_Rn ::splitCloudOfVertices
VolumeOfPolytopes_Rn ::compute
main
politopixAPI::computeVolume
test_main

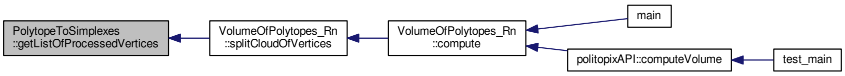
4.21.3.5 void PolytopeToSimplexes::dump (std::ostream & *this_ostream*, unsigned int *shift* = 0) const [inline]

Definition at line 114 of file VolumeOfPolytopes_Rn.h.

4.21.3.6 const std::vector< unsigned int >& PolytopeToSimplexes::getListOfProcessedVertices ( ) const [inline]

Definition at line 68 of file VolumeOfPolytopes_Rn.h.

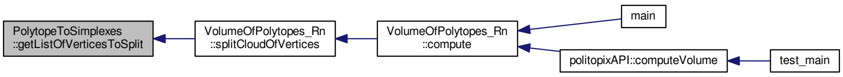
Here is the caller graph for this function:


PolytopeToSimplexes ::getListOfProcessedVertices
VolumeOfPolytopes_Rn ::splitCloudOfVertices
VolumeOfPolytopes_Rn ::compute
main
politopixAPI::computeVolume
test_main

4.21.3.7 const std::vector< unsigned int >& PolytopeToSimplexes::getListOfVerticesToSplit ( ) const [inline]

Definition at line 76 of file VolumeOfPolytopes_Rn.h.

Here is the caller graph for this function:


PolytopeToSimplexes ::getListOfVerticesToSplit
VolumeOfPolytopes_Rn ::splitCloudOfVertices
VolumeOfPolytopes_Rn ::compute
main
politopixAPI::computeVolume
test_main

4.21.3.8 void PolytopeToSimplexes::setListOfProcessedVertices (const std::vector< unsigned int > & *LPV*) [inline]

Definition at line 72 of file VolumeOfPolytopes_Rn.h.

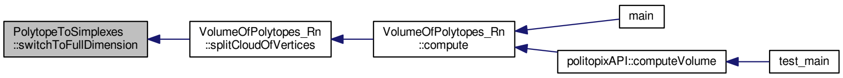
4.21.3.9 void PolytopeToSimplexes::setListOfVerticesToSplit (const std::vector< unsigned int > & *VTS*) [inline]

Definition at line 80 of file VolumeOfPolytopes_Rn.h.

4.21.3.10 void PolytopeToSimplexes::switchToFullDimension ( ) [inline]

Definition at line 104 of file VolumeOfPolytopes_Rn.h.

Here is the caller graph for this function:



4.21.4 Member Data Documentation

4.21.4.1 unsigned int PolytopeToSimplexes::_dimension [protected]

The current dimension space we work in.

Definition at line 131 of file VolumeOfPolytopes_Rn.h.

4.21.4.2 std::vector< unsigned int > PolytopeToSimplexes::_listOfProcessedVertices [protected]

The ordered list of vertices as we go down in smaller dimensions spaces.

Definition at line 127 of file VolumeOfPolytopes_Rn.h.

4.21.4.3 std::vector< unsigned int > PolytopeToSimplexes::_listOfVerticesToSplit [protected]

The ordered list of vertices to be split in lower dimensions.

Definition at line 129 of file VolumeOfPolytopes_Rn.h.

The documentation for this class was generated from the following file:

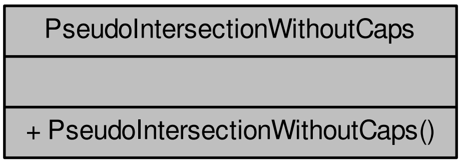
- /home/vindelos/CPP/I2M/politopix/trunk/VolumeOfPolytopes_Rn.h

4.22 PseudoIntersectionWithoutCaps Class Reference

Remove all cap half-spaces and then compute the intersection of two capped polytopes.

`#include <PolyhedralAlgorithms_Rn.h>`

Collaboration diagram for PseudoIntersectionWithoutCaps:


PseudoIntersectionWithoutCaps
+ PseudoIntersectionWithoutCaps()

Public Member Functions

- PseudoIntersectionWithoutCaps (const boost::shared_ptr< Polytope_Rn > &A, const boost::shared_ptr< Polytope_Rn > &B, boost::shared_ptr< Polytope_Rn > &C, const std::set< unsigned int > &firstOperandCaps, const std::set< unsigned int > &secondOperandCaps, std::set< unsigned int > &newCaps, double bb_size=1000.)

    Remove all cap half-spaces and then compute the intersection of two capped polytopes.

4.22.1 Detailed Description

Remove all cap half-spaces and then compute the intersection of two capped polytopes.

Definition at line 281 of file PolyhedralAlgorithms_Rn.h.

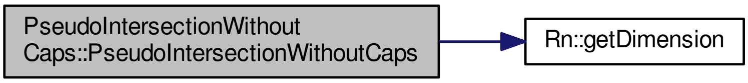
4.22.2 Constructor & Destructor Documentation

4.22.2.1 PseudoIntersectionWithoutCaps::PseudoIntersectionWithoutCaps (const boost::shared_ptr< Polytope_Rn > & *A*, const boost::shared_ptr< Polytope_Rn > & *B*, boost::shared_ptr< Polytope_Rn > & *C*, const std::set< unsigned int > & *firstOperandCaps*, const std::set< unsigned int > & *secondOperandCaps*, std::set< unsigned int > & *newCaps*, double *bb_size* = 1000.)

Remove all cap half-spaces and then compute the intersection of two capped polytopes.

Definition at line 1080 of file PolyhedralAlgorithms_Rn.cpp.

Here is the call graph for this function:


PseudoIntersectionWithout
Caps::PseudoIntersectionWithoutCaps
Rn::getDimension

The documentation for this class was generated from the following files:

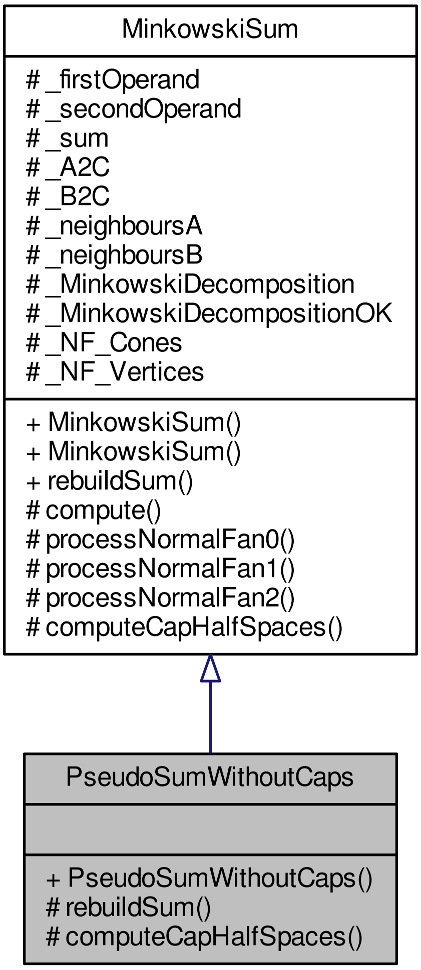
- /home/vindelos/CPP/I2M/politopix/trunk/PolyhedralAlgorithms_Rn.h
- /home/vindelos/CPP/I2M/politopix/trunk/PolyhedralAlgorithms_Rn.cpp

4.23 PseudoSumWithoutCaps Class Reference

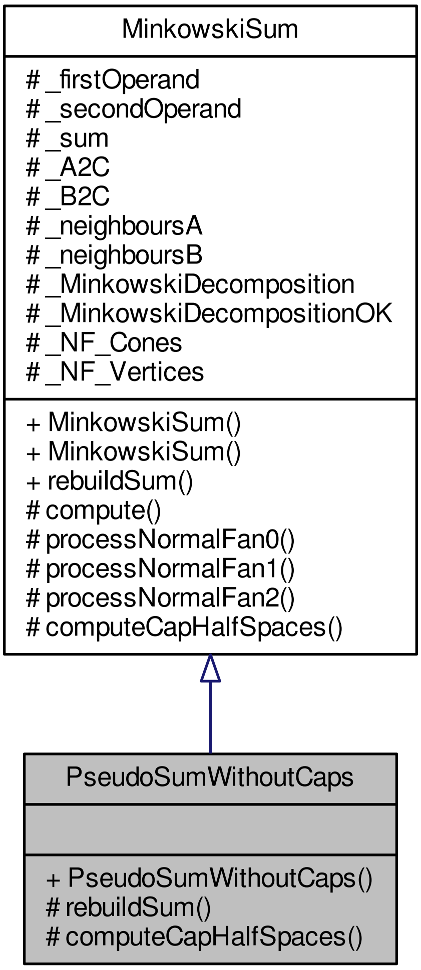
Compute the Minkowski sum of two polytopes and then remove all cap half-spaces to truncate again.

```
#include <PolyhedralAlgorithms_Rn.h>
```

Inheritance diagram for PseudoSumWithoutCaps:



Collaboration diagram for PseudoSumWithoutCaps:



Public Member Functions

- PseudoSumWithoutCaps (const boost::shared_ptr< Polytope_Rn > &A, const boost::shared_ptr< Polytope_Rn > &B, boost::shared_ptr< Polytope_Rn > &C, const std::set< unsigned int > &firstOperandCaps, const std::set< unsigned int > &secondOperandCaps, std::set< unsigned int > &newCaps, double bb_size=1000.)

  Compute the Minkowski sum of two polytopes and then remove all cap half-spaces to truncate again.

Protected Member Functions

- boost::shared_ptr< Polytope_Rn > rebuildSum (const std::set< unsigned int > &firstOperandCaps, const std::set< unsigned int > &secondOperandCaps, std::set< unsigned int > &newCaps, double bb_size=1000.)

  Remove the cap half-spaces stored in sets and then truncate again.
- void computeCapHalfSpaces (const std::set< unsigned int > &firstOperandCaps, const std::set< unsigned int > &secondOperandCaps, std::set< unsigned int > &sumCaps) throw (std::domain_error)

Return the cap half-spaces of the sum in function of the two operands cap half-spaces.

Additional Inherited Members

4.23.1 Detailed Description

Compute the Minkowski sum of two polytopes and then remove all cap half-spaces to truncate again.

Definition at line 244 of file PolyhedralAlgorithms_Rn.h.

4.23.2 Constructor & Destructor Documentation

4.23.2.1 PseudoSumWithoutCaps::PseudoSumWithoutCaps (const boost::shared_ptr< Polytope_Rn > & *A*, const boost::shared_ptr< Polytope_Rn > & *B*, boost::shared_ptr< Polytope_Rn > & *C*, const std::set< unsigned int > & *firstOperandCaps*, const std::set< unsigned int > & *secondOperandCaps*, std::set< unsigned int > & *newCaps*, double *bb_size* = `1000.`) `[inline]`

Compute the Minkowski sum of two polytopes and then remove all cap half-spaces to truncate again.

Definition at line 249 of file PolyhedralAlgorithms_Rn.h.

4.23.3 Member Function Documentation

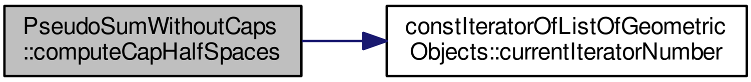
4.23.3.1 void PseudoSumWithoutCaps::computeCapHalfSpaces (const std::set< unsigned int > & *firstOperandCaps*, const std::set< unsigned int > & *secondOperandCaps*, std::set< unsigned int > & *sumCaps*) throw std::domain_error) `[protected]`

Return the cap half-spaces of the sum in function of the two operands cap half-spaces.

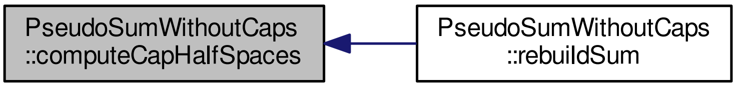
For each facet of the sum F_C, build F_A and F_B such as F_C = F_A + F_B.

Definition at line 825 of file PolyhedralAlgorithms_Rn.cpp.

Here is the call graph for this function:


PseudoSumWithoutCaps ::computeCapHalfSpaces
constIteratorOfListOfGeometric Objects::currentIteratorNumber

Here is the caller graph for this function:


PseudoSumWithoutCaps ::computeCapHalfSpaces
PseudoSumWithoutCaps ::rebuildSum

4.23.3.2 boost::shared_ptr< **Polytope_Rn** > PseudoSumWithoutCaps::rebuildSum (const std::set< unsigned int > & *firstOperandCaps*, const std::set< unsigned int > & *secondOperandCaps*, std::set< unsigned int > & *newCaps*, double *bb_size* = `1000.`) `[protected]`

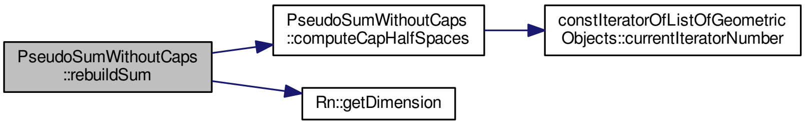
Remove the cap half-spaces stored in sets and then truncate again.

Returns

The new sum

Definition at line 1017 of file PolyhedralAlgorithms_Rn.cpp.

Here is the call graph for this function:


PseudoSumWithoutCaps ::rebuildSum
PseudoSumWithoutCaps ::computeCapHalfSpaces
constIteratorOfListOfGeometric Objects::currentIteratorNumber
Rn::getDimension

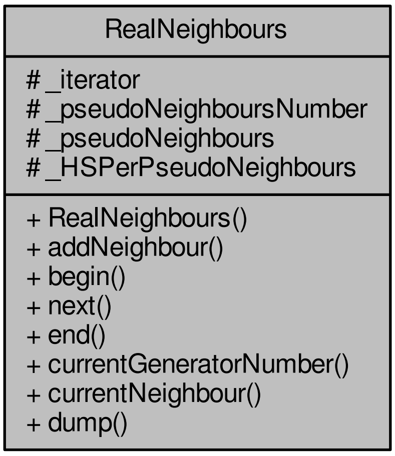
The documentation for this class was generated from the following files:

- /home/vindelos/CPP/I2M/politopix/trunk/PolyhedralAlgorithms_Rn.h
- /home/vindelos/CPP/I2M/politopix/trunk/PolyhedralAlgorithms_Rn.cpp

4.24 RealNeighbours Class Reference

Class dedicated to degeneration processing when looking for neighbors.

Collaboration diagram for RealNeighbours:



Public Member Functions

- RealNeighbours ()
- bool addNeighbour (const std::vector< HalfSpace_Rn * > &commonFacets, const boost::shared_ptr< Generator_Rn > &gen, unsigned int number=0)

  Tell whether a pseudo neighbor is a genuine one comparing set of half-spaces.
- void begin ()

  Iterator function.
- void next ()

  Iterator function.
- bool end ()

  Iterator function.
- unsigned int currentGeneratorNumber ()

  Iterator function.
- boost::shared_ptr< Generator_Rn > currentNeighbour ()

  Iterator function.
- void dump (std::ostream &ofs)

  Display the content on the stream passed as an argument.

Protected Attributes

- unsigned int _iterator

  A runner to iterate through the list of genuine neighbors.
- std::vector< unsigned int > _pseudoNeighboursNumber

  The generator numbers in a global list.
- std::vector< boost::shared_ptr < Generator_Rn > > _pseudoNeighbours

  The generator smart pointer.
- std::vector< std::vector < HalfSpace_Rn * > > _HSPerPseudoNeighbours

  For each generator, store all raw pointers on their corresponding half-spaces.

4.24.1 Detailed Description

Class dedicated to degeneration processing when looking for neighbors.

Definition at line 36 of file Polytope_Rn.cpp.

4.24.2 Constructor & Destructor Documentation

4.24.2.1 RealNeighbours::RealNeighbours ( ) [inline]

Definition at line 39 of file Polytope_Rn.cpp.

4.24.3 Member Function Documentation

4.24.3.1 bool RealNeighbours::addNeighbour (const std::vector< HalfSpace_Rn * > & *commonFacets*, const boost::shared_ptr< Generator_Rn > & *gen*, unsigned int *number* = 0) [inline]

Tell whether a pseudo neighbor is a genuine one comparing set of half-spaces.

Parameters

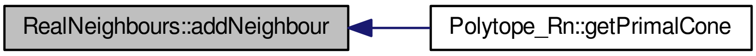
commonFacets	the set of common half-spaces pointers between *this* and *gen*
gen	the other generator candidate to be a genuine neighbor of *this*
number	the other generator number, if provided

Returns

true if SO FAR none match has been found.

Definition at line 46 of file Polytope_Rn.cpp.

Here is the caller graph for this function:

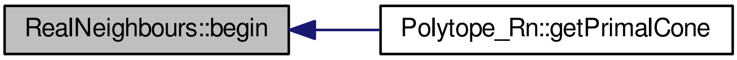


4.24.3.2 void RealNeighbours::begin ( ) [inline]

Iterator function.

Definition at line 79 of file Polytope_Rn.cpp.

Here is the caller graph for this function:



4.24.3.3 unsigned int RealNeighbours::currentGeneratorNumber ( ) [inline]

Iterator function.

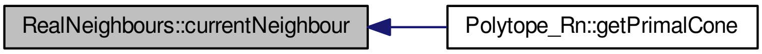
Definition at line 88 of file Polytope_Rn.cpp.

4.24.3.4 boost::shared_ptr<Generator_Rn> RealNeighbours::currentNeighbour ( ) [inline]

Iterator function.

Definition at line 91 of file Polytope_Rn.cpp.

Here is the caller graph for this function:



4.24.3.5 void RealNeighbours::dump (std::ostream & *ofs*) [inline]

Display the content on the stream passed as an argument.

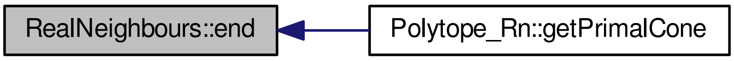
Definition at line 94 of file Polytope_Rn.cpp.

4.24.3.6 bool RealNeighbours::end ( ) [inline]

Iterator function.

Definition at line 85 of file Polytope_Rn.cpp.

Here is the caller graph for this function:

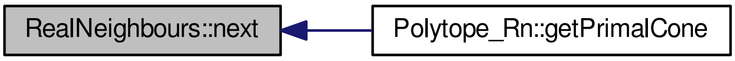


4.24.3.7 void RealNeighbours::next ( ) [inline]

Iterator function.

Definition at line 82 of file Polytope_Rn.cpp.

Here is the caller graph for this function:



4.24.4 Member Data Documentation

4.24.4.1 std::vector< std::vector< HalfSpace_Rn* > > RealNeighbours::_HSPerPseudoNeighbours [protected]

For each generator, store all raw pointers on their corresponding half-spaces.

Definition at line 111 of file Polytope_Rn.cpp.

4.24.4.2 unsigned int RealNeighbours::_iterator [protected]

A runner to iterate through the list of genuine neighbors.

Definition at line 105 of file Polytope_Rn.cpp.

4.24.4.3 std::vector< boost::shared_ptr<Generator_Rn> > RealNeighbours::_pseudoNeighbours [protected]

The generator smart pointer.

Definition at line 109 of file Polytope_Rn.cpp.

4.24.4.4 std::vector< unsigned int > RealNeighbours::_pseudoNeighboursNumber [protected]

The generator numbers in a global list.

Definition at line 107 of file Polytope_Rn.cpp.

The documentation for this class was generated from the following file:

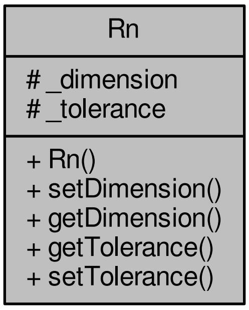
- /home/vindelos/CPP/I2M/politopix/trunk/Polytope_Rn.cpp

4.25 Rn Class Reference

This class stores static function that dispatch the main geometric values we use.

`#include <Rn.h>`

Collaboration diagram for Rn:



Public Member Functions

- Rn ()

Static Public Member Functions

- static polito_EXPORT void setDimension (unsigned int dim)

  Set the dimension for the cartesian space we work in.

- static polito_EXPORT unsigned int getDimension ()

  Return the dimension of the cartesian space we work in.

- static polito_EXPORT double getTolerance ()

  Give the minimum distance between two points.

- static polito_EXPORT void setTolerance (double t)

  Give the minimum distance between two points.

Static Protected Attributes

- static unsigned int _dimension = 6

  Rn dimension.

- static double _tolerance = 1.e-06

  Rn dimension.

4.25.1 Detailed Description

This class stores static function that dispatch the main geometric values we use.

Definition at line 27 of file Rn.h.

4.25.2 Constructor & Destructor Documentation

4.25.2.1 Rn::Rn ( ) [inline]

Definition at line 30 of file Rn.h.

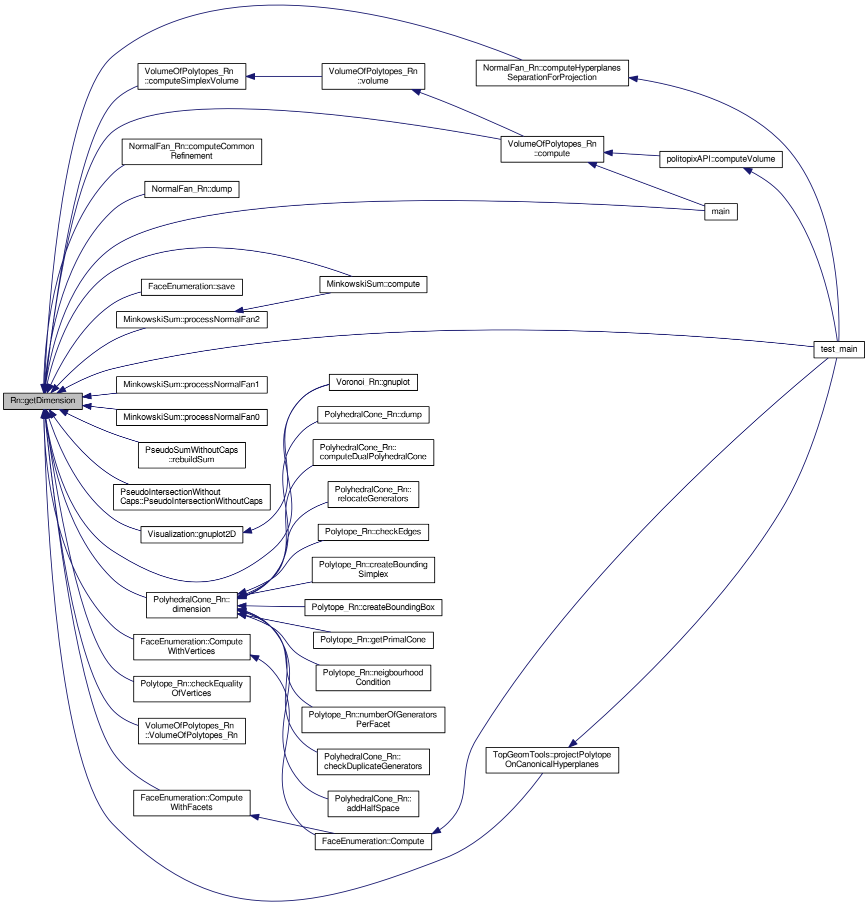
4.25.3 Member Function Documentation

4.25.3.1 unsigned int Rn::getDimension ( ) [static]

Return the dimension of the cartesian space we work in.

Definition at line 29 of file Rn.cpp.

Here is the caller graph for this function:

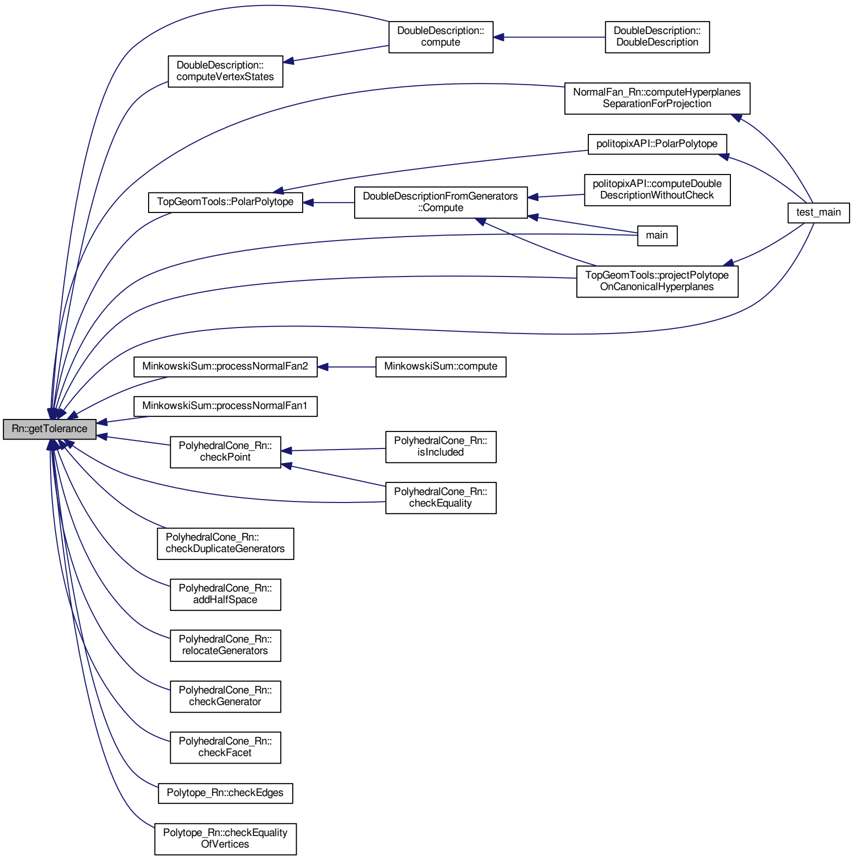

NormalFan_Rn::computeHyperplanes SeparationForProjection
VolumeOfPolytopes_Rn ::computeSimplexVolume
VolumeOfPolytopes_Rn ::volume
NormalFan_Rn::computeCommon Refinement
VolumeOfPolytopes_Rn ::compute
politopixAPI::computeVolume
NormalFan_Rn::dump
main
FaceEnumeration::save
MinkowskiSum::compute
MinkowskiSum::processNormalFan2
test_main
MinkowskiSum::processNormalFan1
Rn::getDimension
MinkowskiSum::processNormalFan0
Voronoi_Rn::gnuplot
PolyhedralCone_Rn::dump
PolyhedralCone_Rn:: computeDualPolyhedralCone
PseudoSumWithoutCaps ::rebuildSum
PolyhedralCone_Rn:: relocateGenerators
PseudoIntersectionWithout Caps::PseudoIntersectionWithoutCaps
Visualization::gnuplot2D
Polytope_Rn::checkEdges
Polytope_Rn::createBounding Simplex
PolyhedralCone_Rn:: dimension
Polytope_Rn::createBoundingBox
Polytope_Rn::getPrimalCone
FaceEnumeration::Compute WithVertices
Polytope_Rn::neigbourhood Condition
Polytope_Rn::checkEquality OfVertices
Polytope_Rn::numberOfGenerators PerFacet
VolumeOfPolytopes_Rn ::VolumeOfPolytopes_Rn
PolyhedralCone_Rn:: checkDuplicateGenerators
TopGeomTools::projectPolytope OnCanonicalHyperplanes
PolyhedralCone_Rn:: addHalfSpace
FaceEnumeration::Compute WithFacets
FaceEnumeration::Compute

4.25.3.2 double Rn::getTolerance ( ) [static]

Give the minimum distance between two points.

Definition at line 31 of file Rn.cpp.

Here is the caller graph for this function:

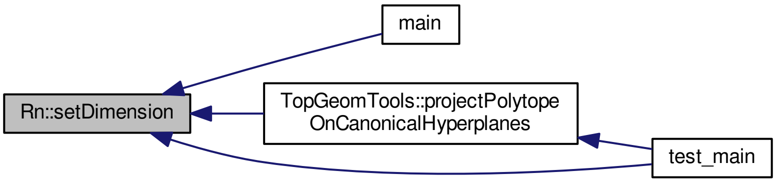

DoubleDescription::compute
DoubleDescription::DoubleDescription
DoubleDescription::computeVertexStates
NormalFan_Rn::computeHyperplanesSeparationForProjection
politopixAPI::PolarPolytope
politopixAPI::computeDoubleDescriptionWithoutCheck
TopGeomTools::PolarPolytope
DoubleDescriptionFromGenerators::Compute
test_main
main
TopGeomTools::projectPolytopeOnCanonicalHyperplanes
MinkowskiSum::processNormalFan2
MinkowskiSum::compute
MinkowskiSum::processNormalFan1
Rn::getTolerance
PolyhedralCone_Rn::checkPoint
PolyhedralCone_Rn::isIncluded
PolyhedralCone_Rn::checkEquality
PolyhedralCone_Rn::checkDuplicateGenerators
PolyhedralCone_Rn::addHalfSpace
PolyhedralCone_Rn::relocateGenerators
PolyhedralCone_Rn::checkGenerator
PolyhedralCone_Rn::checkFacet
Polytope_Rn::checkEdges
Polytope_Rn::checkEqualityOfVertices

4.25.3.3 void Rn::setDimension (unsigned int *dim*) [static]

Set the dimension for the cartesian space we work in.

Definition at line 27 of file Rn.cpp.

Here is the caller graph for this function:

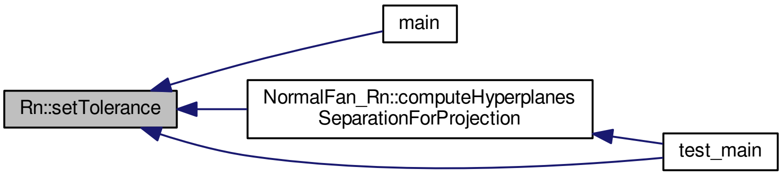


4.25.3.4 void Rn::setTolerance (double *t*) [static]

Give the minimum distance between two points.

Definition at line 33 of file Rn.cpp.

Here is the caller graph for this function:



4.25.4 Member Data Documentation

4.25.4.1 unsigned int Rn::_dimension = 6 [static], [protected]

Rn dimension.

Definition at line 46 of file Rn.h.

4.25.4.2 double Rn::_tolerance = 1.e-06 [static], [protected]

Rn dimension.

Definition at line 48 of file Rn.h.

The documentation for this class was generated from the following files:

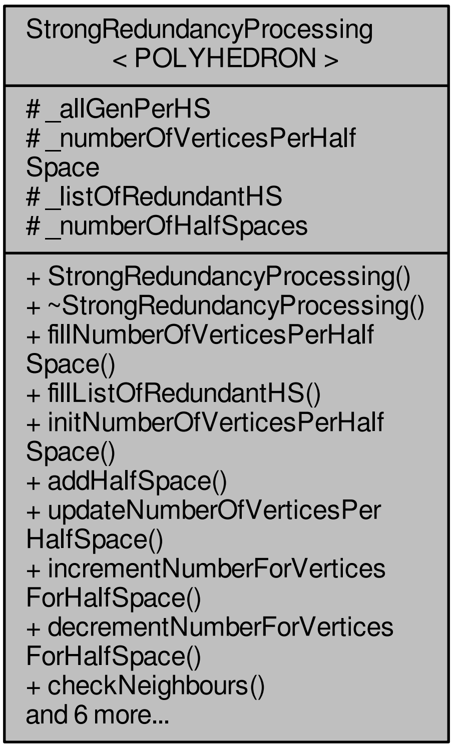
- /home/vindelos/CPP/I2M/politopix/trunk/Rn.h
- /home/vindelos/CPP/I2M/politopix/trunk/Rn.cpp

4.26 StrongRedundancyProcessing< POLYHEDRON > Class Template Reference

This class can be more time-consuming than WeakRedundancyProcessing or NoRedundancyProcessing because it will perform extra checks in the process of intersecting half-spaces. To determine if two vertices are neighbors it will make sure not to count half-spaces marked as redundant.

```
#include <DoubleDescription_Rn.h>
```

Collaboration diagram for StrongRedundancyProcessing< POLYHEDRON >:


StrongRedundancyProcessing
< POLYHEDRON >
# _allGenPerHS
# _numberOfVerticesPerHalf
Space
# _listOfRedundantHS
# _numberOfHalfSpaces
+ StrongRedundancyProcessing()
+ ~StrongRedundancyProcessing()
+ fillNumberOfVerticesPerHalf
Space()
+ fillListOfRedundantHS()
+ initNumberOfVerticesPerHalf
Space()
+ addHalfSpace()
+ updateNumberOfVerticesPer
HalfSpace()
+ incrementNumberForVertices
ForHalfSpace()
+ decrementNumberForVertices
ForHalfSpace()
+ checkNeighbours()
and 6 more...

Public Member Functions

- StrongRedundancyProcessing ()
- virtual ∼StrongRedundancyProcessing ()
- void fillNumberOfVerticesPerHalfSpace (const POLYHEDRON, std::vector< unsigned int > &getNumberOfVerticesPerHalfSpace)
- void fillListOfRedundantHS (const POLYHEDRON, std::vector< unsigned int > &, std::set< unsigned int > &getListOfRedundantHS)
- void initNumberOfVerticesPerHalfSpace (const std::vector< boost::shared_ptr< Generator_Rn_SD > > &LG, unsigned int nbHS)

  Make sure all back pointers from half-spaces to vertices are set.
- void addHalfSpace ()

  Make space for a new half-space.
- void updateNumberOfVerticesPerHalfSpace (unsigned int HS, const std::vector< boost::shared_ptr< Generator_Rn_SD > > &GN_ON)

  The current face must mark all the vertices with state ON.
- void incrementNumberForVerticesForHalfSpace (const boost::shared_ptr< Generator_Rn_SD > &GEN)

  Make sure all the half-spaces belonging to a given generator have their vertices number incremented.
- void decrementNumberForVerticesForHalfSpace (const boost::shared_ptr< Generator_Rn_SD > &GEN)

  Make sure all the half-spaces belonging to a given generator have their vertices number decremented.
- bool checkNeighbours (POLYHEDRON poly, const boost::shared_ptr< Generator_Rn_SD > &genIn, const boost::shared_ptr< Generator_Rn_SD > &genOut, std::vector< unsigned int > &commonFacets)

- virtual void updateListOfRedundantHalfSpaces (unsigned int numberOfGeneratorsPerFacet)
- void fillListOfRedundantHS (const POLYHEDRON poly)
- void unhookRedundantHalfSpaces (POLYHEDRON poly)
- void markHdescription (TrackingOperatorToResult &trackerHdesc, unsigned int truncationStep)
- void dumpListOfRedundantHS (POLYHEDRON poly, std::ostream &this_stream)
- void dumpSD (std::ostream &this_stream)

Protected Attributes

- std::vector< std::set
  < unsigned int > > _allGenPerHS

  Store all raw back pointers to know which vertices belong to a given half-space.
- std::vector< unsigned int > _numberOfVerticesPerHalfSpace

  To know about how many vertices refer to a given half-space.
- std::set< unsigned int > _listOfRedundantHS

  To know whether an half-space has been ticked redundant or not.
- unsigned int _numberOfHalfSpaces

  The total number of processed half-spaces.

4.26.1 Detailed Description

template< class POLYHEDRON >class StrongRedundancyProcessing< POLYHEDRON >

This class can be more time-consuming than WeakRedundancyProcessing or NoRedundancyProcessing because it will perform extra checks in the process of intersecting half-spaces. To determine if two vertices are neighbors it will make sure not to count half-spaces marked as redundant.

Definition at line 514 of file DoubleDescription_Rn.h.

4.26.2 Constructor & Destructor Documentation

4.26.2.1 template< class POLYHEDRON > StrongRedundancyProcessing< POLYHEDRON >::StrongRedundancyProcessing ( ) [inline]

Definition at line 517 of file DoubleDescription_Rn.h.

4.26.2.2 template< class POLYHEDRON > virtual StrongRedundancyProcessing< POLYHEDRON >::~StrongRedundancyProcessing ( ) [inline], [virtual]

Definition at line 519 of file DoubleDescription_Rn.h.

4.26.3 Member Function Documentation

4.26.3.1 template< class POLYHEDRON > void StrongRedundancyProcessing< POLYHEDRON >::addHalfSpace ( ) [inline]

Make space for a new half-space.

Definition at line 576 of file DoubleDescription_Rn.h.

4.26.3.2 template< class POLYHEDRON > bool **StrongRedundancyProcessing**< POLYHEDRON >::checkNeighbours (POLYHEDRON *poly*, const boost::shared_ptr< **Generator_Rn_SD** > & *genIn*, const boost::shared_ptr< **Generator_Rn_SD** > & *genOut*, std::vector< unsigned int > & *commonFacets*) [inline]

Call poly->checkNeighbours() with an extra argument not to count redundant half-spaces in the process of declaring two vertices as neighbors.

Definition at line 611 of file DoubleDescription_Rn.h.

4.26.3.3 template< class POLYHEDRON > void **StrongRedundancyProcessing**< POLYHEDRON >::decrementNumberForVerticesForHalfSpace (const boost::shared_ptr< **Generator_Rn_SD** > & *GEN*) [inline]

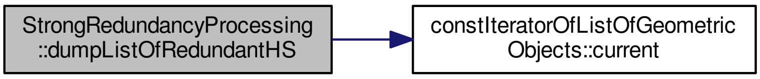
Make sure all the half-spaces belonging to a given generator have their vertices number decremented.

Definition at line 602 of file DoubleDescription_Rn.h.

4.26.3.4 template< class POLYHEDRON > void **StrongRedundancyProcessing**< POLYHEDRON >::dumpListOfRedundantHS (POLYHEDRON *poly*, std::ostream & *this_stream*) [inline]

Definition at line 743 of file DoubleDescription_Rn.h.

Here is the call graph for this function:


StrongRedundancyProcessing ::dumpListOfRedundantHS
constIteratorOfListOfGeometric Objects::current

4.26.3.5 template< class POLYHEDRON > void **StrongRedundancyProcessing**< POLYHEDRON >::dumpSD (std::ostream & *this_stream*) [inline]

Definition at line 764 of file DoubleDescription_Rn.h.

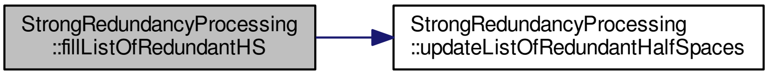
4.26.3.6 template< class POLYHEDRON > void **StrongRedundancyProcessing**< POLYHEDRON >::fillListOfRedundantHS (const POLYHEDRON , std::vector< unsigned int > & , std::set< unsigned int > & *getListOfRedundantHS*) [inline]

Definition at line 527 of file DoubleDescription_Rn.h.

4.26.3.7 template< class POLYHEDRON > void **StrongRedundancyProcessing**< POLYHEDRON >::fillListOfRedundantHS (const POLYHEDRON *poly*) [inline]

Definition at line 673 of file DoubleDescription_Rn.h.

Here is the call graph for this function:


StrongRedundancyProcessing ::fillListOfRedundantHS
StrongRedundancyProcessing ::updateListOfRedundantHalfSpaces

4.26.3.8 template< class POLYHEDRON > void **StrongRedundancyProcessing**< POLYHEDRON >::fillNumberOfVerticesPerHalfSpace (const POLYHEDRON , std::vector< unsigned int > & *getNumberOfVerticesPerHalfSpace*) `[inline]`

Definition at line 521 of file DoubleDescription_Rn.h.

4.26.3.9 template< class POLYHEDRON > void **StrongRedundancyProcessing**< POLYHEDRON >::incrementNumberForVerticesForHalfSpace (const boost::shared_ptr< **Generator_Rn_SD** > & *GEN*) `[inline]`

Make sure all the half-spaces belonging to a given generator have their vertices number incremented.

Definition at line 594 of file DoubleDescription_Rn.h.

4.26.3.10 template< class POLYHEDRON > void **StrongRedundancyProcessing**< POLYHEDRON >::initNumberOfVerticesPerHalfSpace (const std::vector< boost::shared_ptr< **Generator_Rn_SD** > > & *LG*, unsigned int *nbHS*) `[inline]`

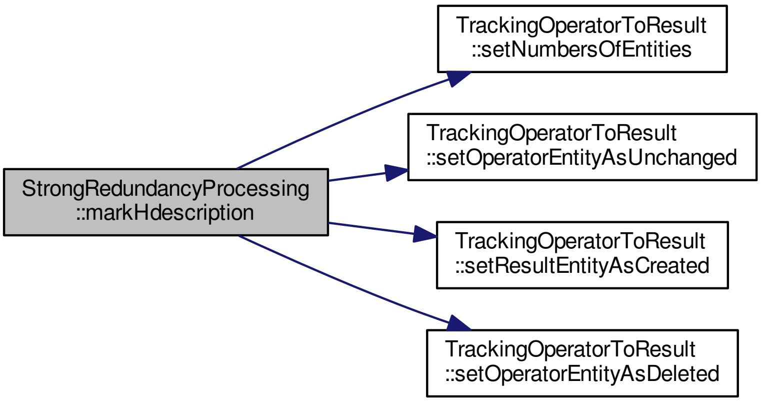
Make sure all back pointers from half-spaces to vertices are set.

Definition at line 535 of file DoubleDescription_Rn.h.

4.26.3.11 template< class POLYHEDRON > void **StrongRedundancyProcessing**< POLYHEDRON >::markHdescription (**TrackingOperatorToResult** & *trackerHdesc*, unsigned int *truncationStep*) `[inline]`

Definition at line 731 of file DoubleDescription_Rn.h.

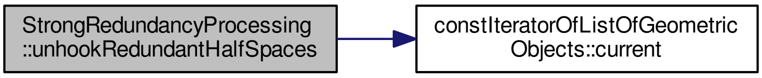
Here is the call graph for this function:


StrongRedundancyProcessing ::markHdescription
TrackingOperatorToResult ::setNumbersOfEntities
TrackingOperatorToResult ::setOperatorEntityAsUnchanged
TrackingOperatorToResult ::setResultEntityAsCreated
TrackingOperatorToResult ::setOperatorEntityAsDeleted

4.26.3.12 template<class POLYHEDRON > void StrongRedundancyProcessing< POLYHEDRON >::unhookRedundantHalfSpaces (POLYHEDRON *poly*) [inline]

Definition at line 675 of file DoubleDescription_Rn.h.

Here is the call graph for this function:

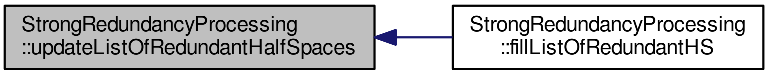

StrongRedundancyProcessing ::unhookRedundantHalfSpaces
constIteratorOfListOfGeometric Objects::current

4.26.3.13 template<class POLYHEDRON > virtual void StrongRedundancyProcessing< POLYHEDRON >::updateListOfRedundantHalfSpaces (unsigned int *numberOfGeneratorsPerFacet*) [inline], [virtual]

Inspect all half-spaces, find their lists of generators and numbers of generators to know whether they are redundant or not.

Definition at line 628 of file DoubleDescription_Rn.h.

Here is the caller graph for this function:


StrongRedundancyProcessing
::updateListOfRedundantHalfSpaces
StrongRedundancyProcessing
::fillListOfRedundantHS

4.26.3.14 template< class POLYHEDRON > void **StrongRedundancyProcessing**< POLYHEDRON >::updateNumberOfVerticesPerHalfSpace (unsigned int *HS*, const std::vector< boost::shared_ptr< **Generator_Rn_SD** > > & *GN_ON*) [inline]

The current face must mark all the vertices with state ON.

Definition at line 584 of file DoubleDescription_Rn.h.

4.26.4 Member Data Documentation

4.26.4.1 template< class POLYHEDRON > std::vector< std::set< unsigned int > > **StrongRedundancyProcessing**< POLYHEDRON >::_allGenPerHS [protected]

Store all raw back pointers to know which vertices belong to a given half-space.

Definition at line 786 of file DoubleDescription_Rn.h.

4.26.4.2 template< class POLYHEDRON > std::set< unsigned int > **StrongRedundancyProcessing**< POLYHEDRON >::_listOfRedundantHS [protected]

To know whether an half-space has been ticked redundant or not.

Definition at line 790 of file DoubleDescription_Rn.h.

4.26.4.3 template< class POLYHEDRON > unsigned int **StrongRedundancyProcessing**< POLYHEDRON >::_numberOfHalfSpaces [protected]

The total number of processed half-spaces.

Definition at line 792 of file DoubleDescription_Rn.h.

4.26.4.4 template< class POLYHEDRON > std::vector< unsigned int > **StrongRedundancyProcessing**< POLYHEDRON >::_numberOfVerticesPerHalfSpace [protected]

To know about how many vertices refer to a given half-space.

Definition at line 788 of file DoubleDescription_Rn.h.

The documentation for this class was generated from the following file:

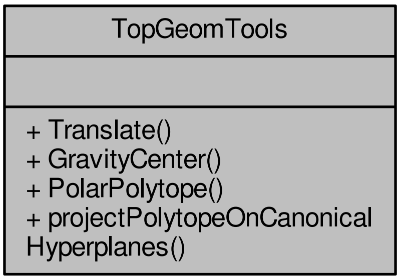
- /home/vindelos/CPP/I2M/politopix/trunk/DoubleDescription_Rn.h

4.27 TopGeomTools Class Reference

Basic tools for topology and geometry: translations, polarity, ...

```
#include <PolyhedralAlgorithms_Rn.h>
```

Collaboration diagram for TopGeomTools:



Static Public Member Functions

- static int Translate (boost::shared_ptr< Polytope_Rn > &pol, const boost::numeric::ublas::vector< double > &v2t)

  Translate a polytope or polyhedral cone by the given vector.
- static int GravityCenter (boost::shared_ptr< Polytope_Rn > &pol, boost::numeric::ublas::vector< double > &gravity_center)

  Translate a polytope or polyhedral cone by the given vector.
- static int PolarPolytope (const boost::shared_ptr< Polytope_Rn > &original_pol, boost::shared_ptr< Polytope_Rn > &polar_pol, bool forceComputation=true, double bb_size=1000.) throw (invalid_argument)

  Compute the polar polytope.
- static int projectPolytopeOnCanonicalHyperplanes (const std::set< unsigned int > &listOfHyperplanes, const boost::shared_ptr< Polytope_Rn > &original_pol, boost::shared_ptr< Polytope_Rn > &proj_pol) throw (invalid_argument)

  Compute the projection of a polytope on the intersection of canonical hyperplanes of the shape $x_i = 0$

4.27.1 Detailed Description

Basic tools for topology and geometry: translations, polarity, ...

Definition at line 300 of file PolyhedralAlgorithms_Rn.h.

4.27.2 Member Function Documentation

4.27.2.1 int TopGeomTools::GravityCenter (boost::shared_ptr< **Polytope_Rn** > & *pol,* boost::numeric::ublas::vector< double > & *gravity_center*) `[static]`

Translate a polytope or polyhedral cone by the given vector.

Parameters

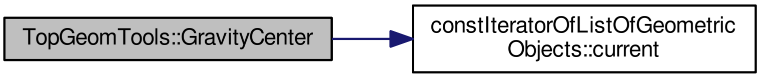
pol	The corresponding polytope
v2t	The translation vector

Returns

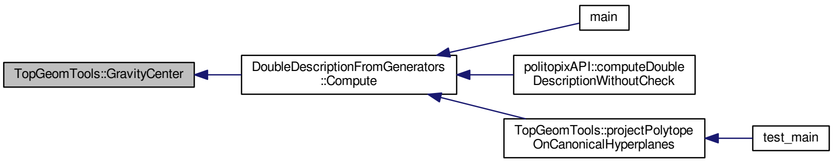
TEST_OK or 0 if the process was successful, TEST_KO or -1 if something went wrong.

Definition at line 1158 of file PolyhedralAlgorithms_Rn.cpp.

Here is the call graph for this function:


TopGeomTools::GravityCenter
constIteratorOfListOfGeometric Objects::current

Here is the caller graph for this function:


TopGeomTools::GravityCenter
DoubleDescriptionFromGenerators ::Compute
main
politopixAPI::computeDouble DescriptionWithoutCheck
TopGeomTools::projectPolytope OnCanonicalHyperplanes
test_main

4.27.2.2 int TopGeomTools::PolarPolytope (const boost::shared_ptr< **Polytope_Rn** > & *original_pol,* boost::shared_ptr< **Polytope_Rn** > & *polar_pol,* bool *forceComputation* = `true`, double *bb_size* = `1000.`) throw invalid_argument) `[static]`

Compute the polar polytope.

Parameters

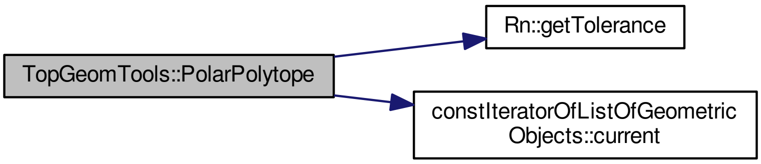
original_pol	The input polytope
polar_pol	The polar polytope

Returns

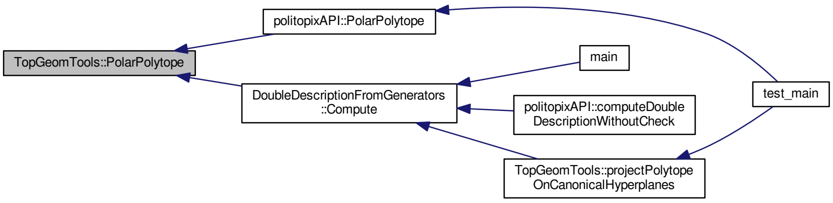
TEST_OK or 0 if the process was successful, TEST_KO or -1 if something went wrong.

Definition at line 1173 of file PolyhedralAlgorithms_Rn.cpp.

Here is the call graph for this function:


TopGeomTools::PolarPolytope
Rn::getTolerance
constIteratorOfListOfGeometricObjects::current

Here is the caller graph for this function:


TopGeomTools::PolarPolytope
politopixAPI::PolarPolytope
main
DoubleDescriptionFromGenerators::Compute
politopixAPI::computeDoubleDescriptionWithoutCheck
test_main
TopGeomTools::projectPolytopeOnCanonicalHyperplanes

4.27.2.3 int TopGeomTools::projectPolytopeOnCanonicalHyperplanes (const std::set< unsigned int > & *listOfHyperplanes*, const boost::shared_ptr< Polytope_Rn > & *original_pol*, boost::shared_ptr< Polytope_Rn > & *proj_pol*) throw invalid_argument) [static]

Compute the projection of a polytope on the intersection of canonical hyperplanes of the shape *x_i = 0*

Parameters

listOf↩ Hyperplanes	The set of indices describing the canonical hyperplanes $i \in listOfHyperplanes \Leftrightarrow x_i = 0$
original_pol	The input polytope
proj_pol	The projected polytope

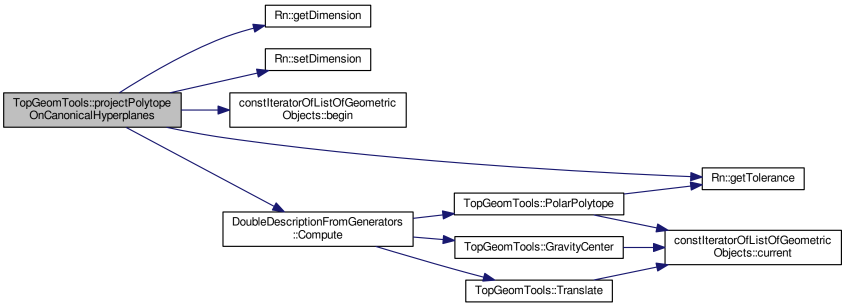
Returns

TEST_OK or 0 if the process was successful, TEST_KO or -1 if something went wrong.

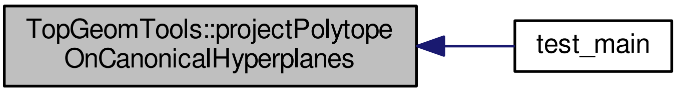
@

Definition at line 1263 of file PolyhedralAlgorithms_Rn.cpp.

Here is the call graph for this function:


Rn::getDimension
Rn::setDimension
TopGeomTools::projectPolytope OnCanonicalHyperplanes
constIteratorOfListOfGeometric Objects::begin
Rn::getTolerance
TopGeomTools::PolarPolytope
DoubleDescriptionFromGenerators ::Compute
TopGeomTools::GravityCenter
constIteratorOfListOfGeometric Objects::current
TopGeomTools::Translate

Here is the caller graph for this function:



4.27.2.4 int TopGeomTools::Translate (boost::shared_ptr< **Polytope_Rn** > & *pol*, const boost::numeric::ublas::vector< double > & *v2t*) [static]

Translate a polytope or polyhedral cone by the given vector.

Parameters

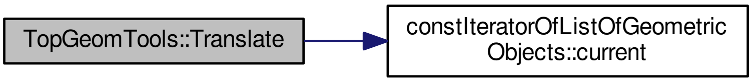
pol	The corresponding polytope
v2t	The translation vector

Returns

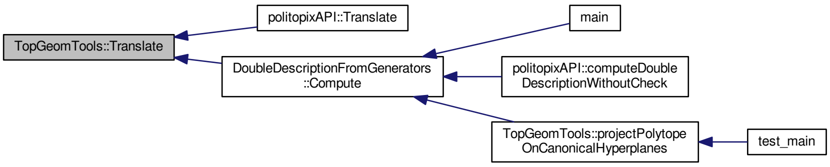
TEST_OK or 0 if the process was successful, TEST_KO or -1 if something went wrong.

Definition at line 1138 of file PolyhedralAlgorithms_Rn.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:


TopGeomTools::Translate
politopixAPI::Translate
main
DoubleDescriptionFromGenerators ::Compute
politopixAPI::computeDouble DescriptionWithoutCheck
TopGeomTools::projectPolytope OnCanonicalHyperplanes
test_main

The documentation for this class was generated from the following files:

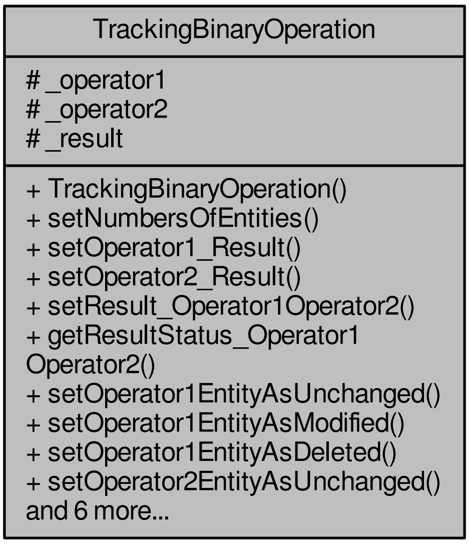
- /home/vindelos/CPP/I2M/politopix/trunk/PolyhedralAlgorithms_Rn.h
- /home/vindelos/CPP/I2M/politopix/trunk/PolyhedralAlgorithms_Rn.cpp

4.28 TrackingBinaryOperation Class Reference

This class stores static function that dispatch the main geometric values we use.

```
#include <Tracking.h>
```

Collaboration diagram for TrackingBinaryOperation:


TrackingBinaryOperation
# _operator1
# _operator2
# _result
+ TrackingBinaryOperation()
+ setNumbersOfEntities()
+ setOperator1_Result()
+ setOperator2_Result()
+ setResult_Operator1Operator2()
+ getResultStatus_Operator1
Operator2()
+ setOperator1EntityAsUnchanged()
+ setOperator1EntityAsModified()
+ setOperator1EntityAsDeleted()
+ setOperator2EntityAsUnchanged()
and 6 more...

Public Member Functions

- TrackingBinaryOperation ()
- void setNumbersOfEntities (unsigned int nbEntBefore1, unsigned int nbEntBefore2, unsigned int nbEntAfter)
- void setOperator1_Result (unsigned int nbOp1, int nbRes)

  Set the link between the nbOp1-th entity of operator1 and the nbRes-th entity of the result.
- void setOperator2_Result (unsigned int nbOp2, int nbRes)

  Set the link between the nbOp2-th entity of operator2 and the nbRes-th entity of the result.
- void setResult_Operator1Operator2 (unsigned int nbRes, int nbOp1, int nbOp2)

  Set the link between the nbRes-th entity of the result and the nbOp1-th entity of operator1 and the nbOp2-th entity of operator2.
- const std::vector< std::pair
  < StatusAfter,
  operator1_operator2 > > & getResultStatus_Operator1Operator2 () const
- void setOperator1EntityAsUnchanged (unsigned int nbOp1)

  Mark Operator_UNCHANGED the nbOp1-th entity of operator1.
- void setOperator1EntityAsModified (unsigned int nbOp1)

  Mark Operator_MODIFIED the nbOp1-th entity of operator1.
- void setOperator1EntityAsDeleted (unsigned int nbOp1)

  Mark Operator_DELETED the nbOp1-th entity of operator1.
- void setOperator2EntityAsUnchanged (unsigned int nbOp2)

  Mark Operator_UNCHANGED the nbOp2-th entity of operator2.
- void setOperator2EntityAsModified (unsigned int nbOp2)

  Mark Operator_MODIFIED the nbOp2-th entity of operator2.
- void setOperator2EntityAsDeleted (unsigned int nbOp2)

  Mark Operator_DELETED the nbOp2-th entity of operator2.
- void setResultEntityAsUnchanged (unsigned int nbRes)

Mark Result_UNCHANGED the nbRes-th entity of the result.
- void setResultEntityAsModified (unsigned int nbRes)

  Mark Result_MODIFIED the nbRes-th entity of the result.
- void setResultEntityAsUnknown (unsigned int nbRes)

  Mark Result_UNKNOWN the nbRes-th entity of the result.
- void setResultEntityAsCreated (unsigned int nbRes)

  Mark Result_CREATED the nbRes-th entity of the result.

Protected Attributes

- std::vector< std::pair
  < StatusBefore, int > > _operator1

  List of the entities before the given operation with their status and the entity number it is connected to after the operation (if need be).
- std::vector< std::pair
  < StatusBefore, int > > _operator2

  List of the entities before the given operation with their status and the entity number it is connected to after the operation (if need be).
- std::vector< std::pair
  < StatusAfter,
  operator1_operator2 > > _result

  List of the entities after the given operation with their status and the entity numbers it is connected to before the operation (if need be).

4.28.1 Detailed Description

This class stores static function that dispatch the main geometric values we use.

Definition at line 110 of file Tracking.h.

4.28.2 Constructor & Destructor Documentation

4.28.2.1 TrackingBinaryOperation::TrackingBinaryOperation ( ) [inline]

Definition at line 114 of file Tracking.h.

4.28.3 Member Function Documentation

4.28.3.1 const std::vector< std::pair< StatusAfter, operator1_operator2 > >& TrackingBinaryOperation::getResultStatus_Operator1Operator2 ( ) const [inline]

Definition at line 149 of file Tracking.h.

4.28.3.2 void TrackingBinaryOperation::setNumbersOfEntities (unsigned int *nbEntBefore1,* unsigned int *nbEntBefore2,* unsigned int *nbEntAfter*) [inline]

Definition at line 116 of file Tracking.h.

4.28.3.3 void TrackingBinaryOperation::setOperator1_Result (unsigned int *nbOp1,* int *nbRes*) [inline]

Set the link between the nbOp1-th entity of operator1 and the nbRes-th entity of the result.

Definition at line 137 of file Tracking.h.

4.28.3.4 void TrackingBinaryOperation::setOperator1EntityAsDeleted (unsigned int *nbOp1*) [inline]

Mark Operator_DELETED the nbOp1-th entity of operator1.

Definition at line 158 of file Tracking.h.

4.28.3.5 void TrackingBinaryOperation::setOperator1EntityAsModified (unsigned int *nbOp1*) [inline]

Mark Operator_MODIFIED the nbOp1-th entity of operator1.

Definition at line 155 of file Tracking.h.

4.28.3.6 void TrackingBinaryOperation::setOperator1EntityAsUnchanged (unsigned int *nbOp1*) [inline]

Mark Operator_UNCHANGED the nbOp1-th entity of operator1.

Definition at line 152 of file Tracking.h.

4.28.3.7 void TrackingBinaryOperation::setOperator2_Result (unsigned int *nbOp2,* int *nbRes*) [inline]

Set the link between the nbOp2-th entity of operator2 and the nbRes-th entity of the result.

Definition at line 140 of file Tracking.h.

4.28.3.8 void TrackingBinaryOperation::setOperator2EntityAsDeleted (unsigned int *nbOp2*) [inline]

Mark Operator_DELETED the nbOp2-th entity of operator2.

Definition at line 167 of file Tracking.h.

4.28.3.9 void TrackingBinaryOperation::setOperator2EntityAsModified (unsigned int *nbOp2*) [inline]

Mark Operator_MODIFIED the nbOp2-th entity of operator2.

Definition at line 164 of file Tracking.h.

4.28.3.10 void TrackingBinaryOperation::setOperator2EntityAsUnchanged (unsigned int *nbOp2*) [inline]

Mark Operator_UNCHANGED the nbOp2-th entity of operator2.

Definition at line 161 of file Tracking.h.

4.28.3.11 void TrackingBinaryOperation::setResult_Operator1Operator2 (unsigned int *nbRes,* int *nbOp1,* int *nbOp2*) [inline]

Set the link between the nbRes-th entity of the result and the nbOp1-th entity of operator1 and the nbOp2-th entity of operator2.

Definition at line 143 of file Tracking.h.

4.28.3.12 void TrackingBinaryOperation::setResultEntityAsCreated (unsigned int *nbRes*) [inline]

Mark Result_CREATED the nbRes-th entity of the result.

Definition at line 179 of file Tracking.h.

4.28.3.13 void TrackingBinaryOperation::setResultEntityAsModified (unsigned int *nbRes*) `[inline]`

Mark Result_MODIFIED the nbRes-th entity of the result.

Definition at line 173 of file Tracking.h.

4.28.3.14 void TrackingBinaryOperation::setResultEntityAsUnchanged (unsigned int *nbRes*) `[inline]`

Mark Result_UNCHANGED the nbRes-th entity of the result.

Definition at line 170 of file Tracking.h.

4.28.3.15 void TrackingBinaryOperation::setResultEntityAsUnknown (unsigned int *nbRes*) `[inline]`

Mark Result_UNKNOWN the nbRes-th entity of the result.

Definition at line 176 of file Tracking.h.

4.28.4 Member Data Documentation

4.28.4.1 std::vector< std::pair< StatusBefore, int > > TrackingBinaryOperation::_operator1 `[protected]`

List of the entities before the given operation with their status and the entity number it is connected to after the operation (if need be).

Definition at line 184 of file Tracking.h.

4.28.4.2 std::vector< std::pair< StatusBefore, int > > TrackingBinaryOperation::_operator2 `[protected]`

List of the entities before the given operation with their status and the entity number it is connected to after the operation (if need be).

Definition at line 186 of file Tracking.h.

4.28.4.3 std::vector< std::pair< StatusAfter, operator1_operator2 > > TrackingBinaryOperation::_result `[protected]`

List of the entities after the given operation with their status and the entity numbers it is connected to before the operation (if need be).

Definition at line 188 of file Tracking.h.

The documentation for this class was generated from the following file:

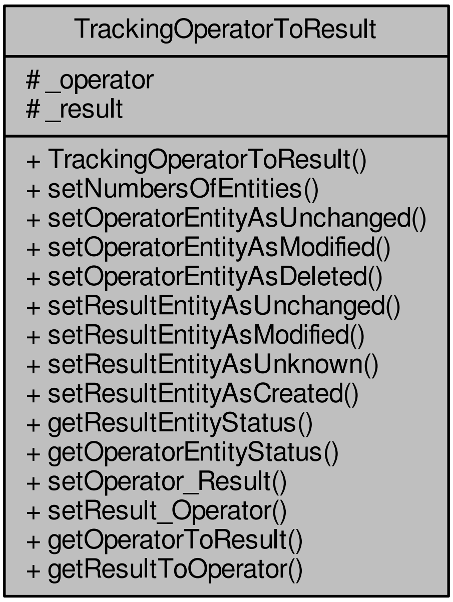
- /home/vindelos/CPP/I2M/politopix/trunk/Tracking.h

4.29 TrackingOperatorToResult Class Reference

This class stores static function that dispatch the main geometric values we use.

`#include <Tracking.h>`

Collaboration diagram for TrackingOperatorToResult:


TrackingOperatorToResult
# _operator
# _result
+ TrackingOperatorToResult()
+ setNumbersOfEntities()
+ setOperatorEntityAsUnchanged()
+ setOperatorEntityAsModified()
+ setOperatorEntityAsDeleted()
+ setResultEntityAsUnchanged()
+ setResultEntityAsModified()
+ setResultEntityAsUnknown()
+ setResultEntityAsCreated()
+ getResultEntityStatus()
+ getOperatorEntityStatus()
+ setOperator_Result()
+ setResult_Operator()
+ getOperatorToResult()
+ getResultToOperator()

Public Member Functions

- TrackingOperatorToResult ()
- void setNumbersOfEntities (unsigned int nbEntBefore, unsigned int nbEntAfter)
- void setOperatorEntityAsUnchanged (unsigned int nb)

  Mark as Operator_UNCHANGED the nb-th entity before the operation.
- void setOperatorEntityAsModified (unsigned int nb)

  Mark as Operator_MODIFIED the nb-th entity before the operation.
- void setOperatorEntityAsDeleted (unsigned int nb)

  Mark as Operator_DELETED the nb-th entity before the operation.
- void setResultEntityAsUnchanged (unsigned int nb)

  Mark as Result_UNCHANGED the nb-th entity before the operation.
- void setResultEntityAsModified (unsigned int nb)

  Mark as Result_MODIFIED the nb-th entity after the operation.
- void setResultEntityAsUnknown (unsigned int nb)

  Mark as Result_UNKNOWN the nb-th entity after the operation.
- void setResultEntityAsCreated (unsigned int nb)

  Mark as Result_CREATED the nb-th entity after the operation.
- StatusAfter getResultEntityStatus (unsigned int nb)

  Get the nb-th entity status after the operation.
- StatusBefore getOperatorEntityStatus (unsigned int nb)

  Get the nb-th entity status after the operation.
- void setOperator_Result (unsigned int nb, int nbRes)

  Set the link between the nb-th entity of operator1 and the nbRes-th entity of the result.
- void setResult_Operator (unsigned int nbRes, int nb)

Set the link between the nb-th entity of operator1 and the nbRes-th entity of the result.

- const std::vector< std::pair
  < StatusBefore, int > > & getOperatorToResult () const
- const std::vector< std::pair
  < StatusAfter, int > > & getResultToOperator () const

Protected Attributes

- std::vector< std::pair
  < StatusBefore, int > > _operator

  List of the entities before the given operation with their status and the entity number it is connected to after the operation (if need be).
- std::vector< std::pair
  < StatusAfter, int > > _result

  List of the entities after the given operation with their status and the entity number it is connected to before the operation (if need be).

4.29.1 Detailed Description

This class stores static function that dispatch the main geometric values we use.

Definition at line 41 of file Tracking.h.

4.29.2 Constructor & Destructor Documentation

4.29.2.1 TrackingOperatorToResult::TrackingOperatorToResult ( ) [inline]

Definition at line 45 of file Tracking.h.

4.29.3 Member Function Documentation

4.29.3.1 StatusBefore TrackingOperatorToResult::getOperatorEntityStatus (unsigned int *nb*) [inline]

Get the nb-th entity status after the operation.

Definition at line 82 of file Tracking.h.

4.29.3.2 const std::vector< std::pair< StatusBefore, int > >& TrackingOperatorToResult::getOperatorToResult ( ) const [inline]

Definition at line 92 of file Tracking.h.

4.29.3.3 StatusAfter TrackingOperatorToResult::getResultEntityStatus (unsigned int *nb*) [inline]

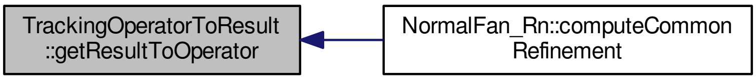
Get the nb-th entity status after the operation.

Definition at line 79 of file Tracking.h.

4.29.3.4 const std::vector< std::pair< StatusAfter, int > >& TrackingOperatorToResult::getResultToOperator ( ) const [inline]

Definition at line 95 of file Tracking.h.

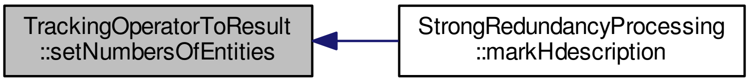
Here is the caller graph for this function:


TrackingOperatorToResult ::getResultToOperator
NormalFan_Rn::computeCommon Refinement

4.29.3.5 void TrackingOperatorToResult::setNumbersOfEntities (unsigned int *nbEntBefore,* unsigned int *nbEntAfter*) [inline]

Definition at line 47 of file Tracking.h.

Here is the caller graph for this function:

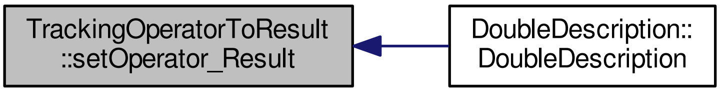

TrackingOperatorToResult ::setNumbersOfEntities
StrongRedundancyProcessing ::markHdescription

4.29.3.6 void TrackingOperatorToResult::setOperator_Result (unsigned int *nb,* int *nbRes*) [inline]

Set the link between the nb-th entity of operator1 and the nbRes-th entity of the result.

Definition at line 86 of file Tracking.h.

Here is the caller graph for this function:

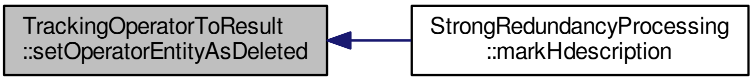

TrackingOperatorToResult ::setOperator_Result
DoubleDescription:: DoubleDescription

4.29.3.7 void TrackingOperatorToResult::setOperatorEntityAsDeleted (unsigned int *nb*) [inline]

Mark as Operator_DELETED the nb-th entity before the operation.

Definition at line 64 of file Tracking.h.

Here is the caller graph for this function:

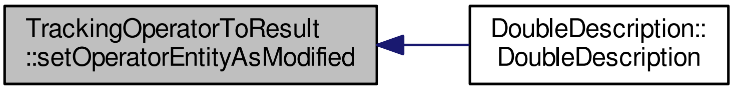


4.29.3.8 void TrackingOperatorToResult::setOperatorEntityAsModified (unsigned int *nb*) [inline]

Mark as Operator_MODIFIED the nb-th entity before the operation.

Definition at line 61 of file Tracking.h.

Here is the caller graph for this function:

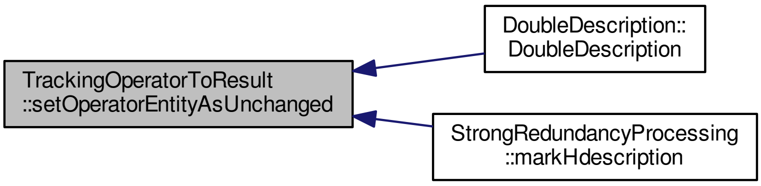


4.29.3.9 void TrackingOperatorToResult::setOperatorEntityAsUnchanged (unsigned int *nb*) [inline]

Mark as Operator_UNCHANGED the nb-th entity before the operation.

Definition at line 58 of file Tracking.h.

Here is the caller graph for this function:

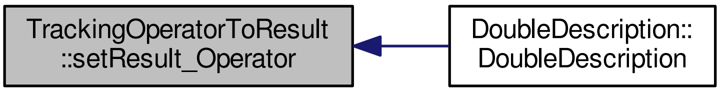


4.29.3.10 void TrackingOperatorToResult::setResult_Operator (unsigned int *nbRes,* int *nb*) [inline]

Set the link between the nb-th entity of operator1 and the nbRes-th entity of the result.

Definition at line 89 of file Tracking.h.

Here is the caller graph for this function:

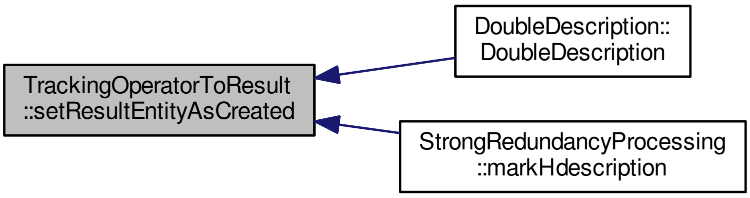


4.29.3.11 void TrackingOperatorToResult::setResultEntityAsCreated (unsigned int *nb*) [inline]

Mark as Result_CREATED the nb-th entity after the operation.

Definition at line 76 of file Tracking.h.

Here is the caller graph for this function:

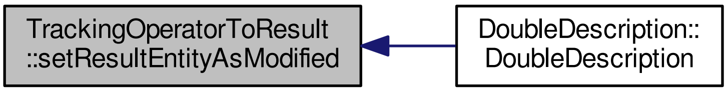


4.29.3.12 void TrackingOperatorToResult::setResultEntityAsModified (unsigned int *nb*) [inline]

Mark as Result_MODIFIED the nb-th entity after the operation.

Definition at line 70 of file Tracking.h.

Here is the caller graph for this function:

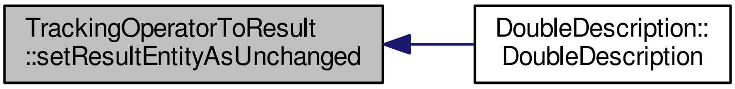


4.29.3.13 void TrackingOperatorToResult::setResultEntityAsUnchanged (unsigned int *nb*) [inline]

Mark as Result_UNCHANGED the nb-th entity before the operation.

Definition at line 67 of file Tracking.h.

Here is the caller graph for this function:



4.29.3.14 void TrackingOperatorToResult::setResultEntityAsUnknown (unsigned int *nb*) [inline]

Mark as Result_UNKNOWN the nb-th entity after the operation.

Definition at line 73 of file Tracking.h.

4.29.4 Member Data Documentation

4.29.4.1 std::vector< std::pair< StatusBefore, int > > TrackingOperatorToResult::_operator [protected]

List of the entities before the given operation with their status and the entity number it is connected to after the operation (if need be).

Definition at line 100 of file Tracking.h.

4.29.4.2 std::vector< std::pair< StatusAfter, int > > TrackingOperatorToResult::_result [protected]

List of the entities after the given operation with their status and the entity number it is connected to before the operation (if need be).

Definition at line 102 of file Tracking.h.

The documentation for this class was generated from the following file:

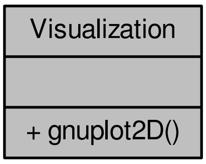
- /home/vindelos/CPP/I2M/politopix/trunk/Tracking.h

4.30 Visualization Class Reference

2D visualization tools

`#include <PolyhedralAlgorithms_Rn.h>`

Collaboration diagram for Visualization:



Static Public Member Functions

- static void gnuplot2D (const boost::shared_ptr< Polytope_Rn > &polygon, const std::string &name, double col, std::ostream &out) throw (std::domain_error)

  Provide the drwing of polygon under the gnuplot format.

4.30.1 Detailed Description

2D visualization tools

Definition at line 355 of file PolyhedralAlgorithms_Rn.h.

4.30.2 Member Function Documentation

4.30.2.1 void Visualization::gnuplot2D (const boost::shared_ptr< **Polytope_Rn** > & *polygon,* const std::string & *name,* double *col,* std::ostream & *out*) throw std::domain_error) [static]

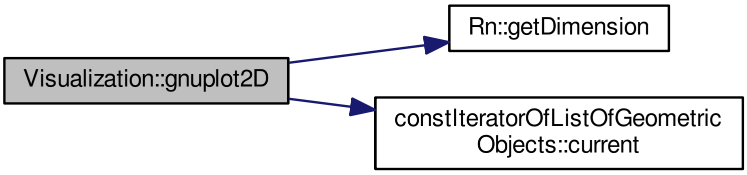
Provide the drwing of polygon under the gnuplot format.

Parameters

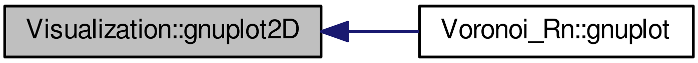
pol	The input 2D polygon
name	The polygon unique name
col	A number in [0,1] coding the color

Definition at line 1360 of file PolyhedralAlgorithms_Rn.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



The documentation for this class was generated from the following files:

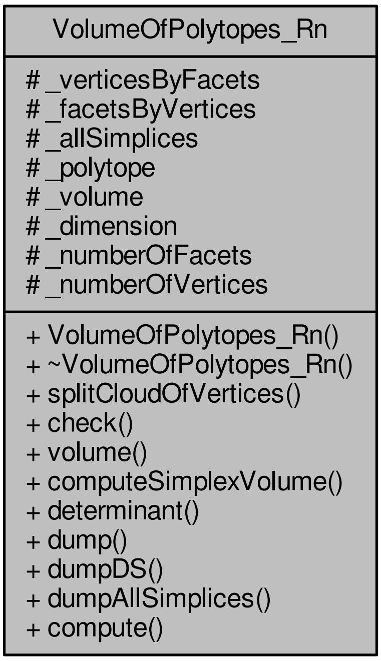
- /home/vindelos/CPP/I2M/politopix/trunk/PolyhedralAlgorithms_Rn.h
- /home/vindelos/CPP/I2M/politopix/trunk/PolyhedralAlgorithms_Rn.cpp

4.31 VolumeOfPolytopes_Rn Class Reference

Split a polytope into simplices to compute its volume.
Two Algorithms for Determining Volumes of Convex Polyhedra (1979) by **Jacques Cohen** and **Timothy Hickey**
Journal of the ACM (JACM) JACM Homepage archive
Volume 26 Issue 3, july 1979
Pages 401-414

```
#include <VolumeOfPolytopes_Rn.h>
```

Collaboration diagram for VolumeOfPolytopes_Rn:



Public Member Functions

- VolumeOfPolytopes_Rn (const boost::shared_ptr< Polytope_Rn > P)

  Constructor.
- ∼VolumeOfPolytopes_Rn ()
- void splitCloudOfVertices (unsigned int DIM)
- void check () const throw (std::domain_error)
- double volume ()

  Sum the volumes of all simplices partitionning the polytope.
- double computeSimplexVolume (const std::set< boost::shared_ptr< Generator_Rn > > &listOfSimplexVertices) const
- double determinant (boost::numeric::ublas::matrix< double > a) const
- void dump (std::ostream &this_ostream)
- void dumpDS (std::ostream &this_ostream) const
- void dumpAllSimplices (std::ostream &this_ostream) const

Static Public Member Functions

- static double compute (const boost::shared_ptr< Polytope_Rn > P)

  Return the volume of the given polytope P.

Protected Attributes

- std::vector< std::vector
  < unsigned int > > _verticesByFacets

  The ordered list of all vertices stored by facets.
- std::vector< std::vector
  < unsigned int > > _facetsByVertices

  The ordered list of all facets stored by vertices.
- std::vector< PolytopeToSimplexes > _allSimplices

  List to store all the simplices partitioning the polytope.
- boost::shared_ptr< Polytope_Rn > _polytope

  The current polytope we are working on.
- double _volume

  The volume of the polytope.
- unsigned int _dimension

  As the algorithm goes down in lower dimensions, we want to store the starting space dimension.
- unsigned int _numberOfFacets

  The number of facets of the current polytope.
- unsigned int _numberOfVertices

  The number of vertices of the current polytope.

4.31.1 Detailed Description

Split a polytope into simplices to compute its volume.
Two Algorithms for Determining Volumes of Convex Polyhedra (1979) by **Jacques Cohen** and **Timothy Hickey**
Journal of the ACM (JACM) JACM Homepage archive
Volume 26 Issue 3, july 1979
Pages 401-414

Definition at line 139 of file VolumeOfPolytopes_Rn.h.

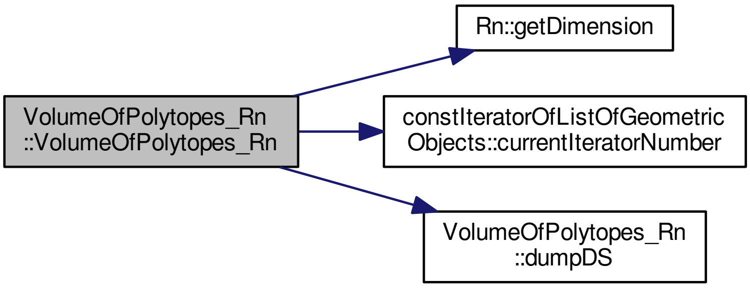
4.31.2 Constructor & Destructor Documentation

4.31.2.1 VolumeOfPolytopes_Rn::VolumeOfPolytopes_Rn (const boost::shared_ptr< **Polytope_Rn** > *P*)

Constructor.

Definition at line 26 of file VolumeOfPolytopes_Rn.cpp.

Here is the call graph for this function:


Rn::getDimension
VolumeOfPolytopes_Rn ::VolumeOfPolytopes_Rn
constIteratorOfListOfGeometric Objects::currentIteratorNumber
VolumeOfPolytopes_Rn ::dumpDS

4.31.2.2 VolumeOfPolytopes_Rn::∼VolumeOfPolytopes_Rn ( ) [inline]

Definition at line 146 of file VolumeOfPolytopes_Rn.h.

4.31.3 Member Function Documentation

4.31.3.1 void VolumeOfPolytopes_Rn::check ( ) const throw std::domain_error)

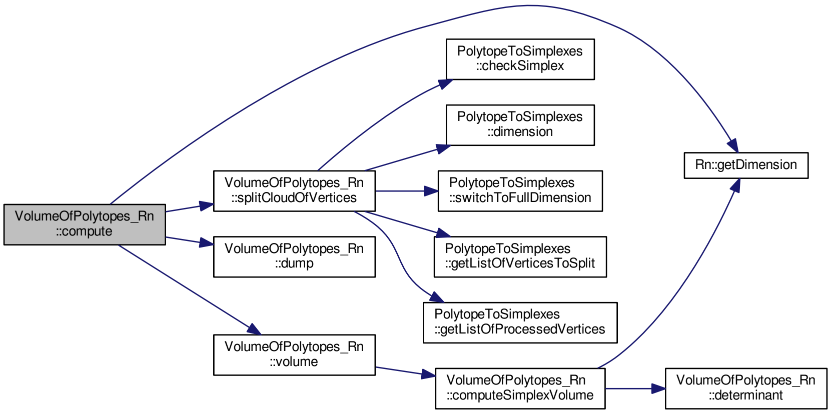
Definition at line 74 of file VolumeOfPolytopes_Rn.cpp.

4.31.3.2 static double VolumeOfPolytopes_Rn::compute (const boost::shared_ptr< **Polytope_Rn** > *P*) [inline], [static]

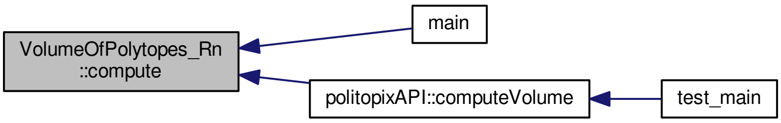
Return the volume of the given polytope P.

Definition at line 153 of file VolumeOfPolytopes_Rn.h.

Here is the call graph for this function:


PolytopeToSimplexes ::checkSimplex
PolytopeToSimplexes ::dimension
Rn::getDimension
VolumeOfPolytopes_Rn ::splitCloudOfVertices
PolytopeToSimplexes ::switchToFullDimension
VolumeOfPolytopes_Rn ::compute
VolumeOfPolytopes_Rn ::dump
PolytopeToSimplexes ::getListOfVerticesToSplit
PolytopeToSimplexes ::getListOfProcessedVertices
VolumeOfPolytopes_Rn ::volume
VolumeOfPolytopes_Rn ::computeSimplexVolume
VolumeOfPolytopes_Rn ::determinant

Here is the caller graph for this function:

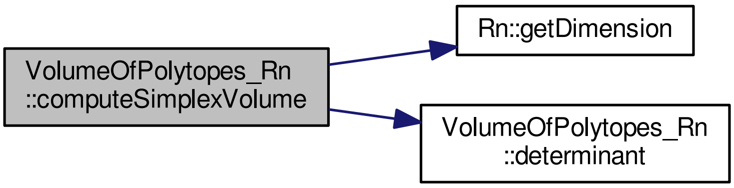


4.31.3.3 double VolumeOfPolytopes_Rn::computeSimplexVolume (const std::set< boost::shared_ptr< Generator_Rn > > & *listOfSimplexVertices*) const

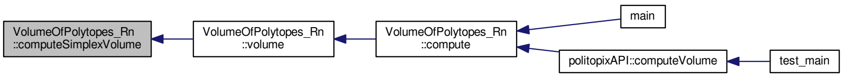
Compute the volume of a simplex making use of the following formula : $Vol\big(conv(v_0,...,v_k)\big) = \frac{|det(v_1-v_0,...,v_k-v_0)|}{n!}$

Definition at line 228 of file VolumeOfPolytopes_Rn.cpp.

Here is the call graph for this function:


VolumeOfPolytopes_Rn ::computeSimplexVolume
Rn::getDimension
VolumeOfPolytopes_Rn ::determinant

Here is the caller graph for this function:

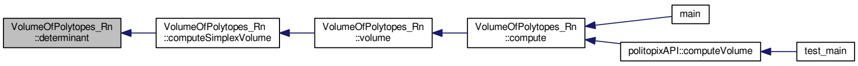

VolumeOfPolytopes_Rn ::computeSimplexVolume
VolumeOfPolytopes_Rn ::volume
VolumeOfPolytopes_Rn ::compute
main
politopixAPI::computeVolume
test_main

4.31.3.4 double VolumeOfPolytopes_Rn::determinant (boost::numeric::ublas::matrix< double > *a*) const

Called by computeSimplexVolume() to compute a square matrix determinant. As we run it on small matrices we just use the minors method.

Definition at line 247 of file VolumeOfPolytopes_Rn.cpp.

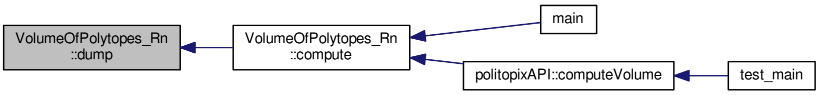
Here is the caller graph for this function:


VolumeOfPolytopes_Rn ::determinant
VolumeOfPolytopes_Rn ::computeSimplexVolume
VolumeOfPolytopes_Rn ::volume
VolumeOfPolytopes_Rn ::compute
main
politopixAPI::computeVolume
test_main

4.31.3.5 void VolumeOfPolytopes_Rn::dump (std::ostream & *this_ostream*) [inline]

Definition at line 200 of file VolumeOfPolytopes_Rn.h.

Here is the caller graph for this function:


VolumeOfPolytopes_Rn ::dump
VolumeOfPolytopes_Rn ::compute
main
politopixAPI::computeVolume
test_main

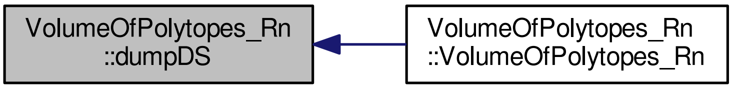
4.31.3.6 void VolumeOfPolytopes_Rn::dumpAllSimplices (std::ostream & *this_ostream*) const [inline]

Definition at line 236 of file VolumeOfPolytopes_Rn.h.

4.31.3.7 void VolumeOfPolytopes_Rn::dumpDS (std::ostream & *this_ostream*) const [inline]

Definition at line 216 of file VolumeOfPolytopes_Rn.h.

Here is the caller graph for this function:


VolumeOfPolytopes_Rn ::dumpDS
VolumeOfPolytopes_Rn ::VolumeOfPolytopes_Rn

4.31.3.8 void VolumeOfPolytopes_Rn::splitCloudOfVertices (unsigned int *DIM*)

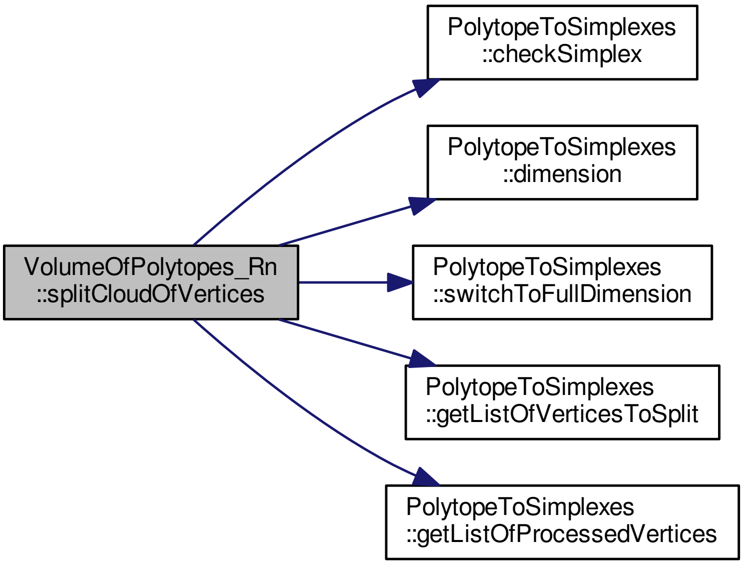
Build all simplices to partition the polytope.

Parameters

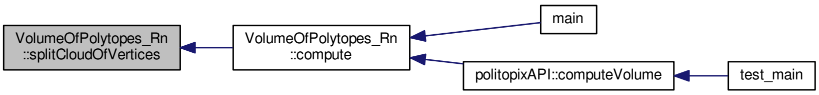
DIM	the dimension of the space we work in

Definition at line 81 of file VolumeOfPolytopes_Rn.cpp.

Here is the call graph for this function:


VolumeOfPolytopes_Rn ::splitCloudOfVertices
PolytopeToSimplexes ::checkSimplex
PolytopeToSimplexes ::dimension
PolytopeToSimplexes ::switchToFullDimension
PolytopeToSimplexes ::getListOfVerticesToSplit
PolytopeToSimplexes ::getListOfProcessedVertices

Here is the caller graph for this function:

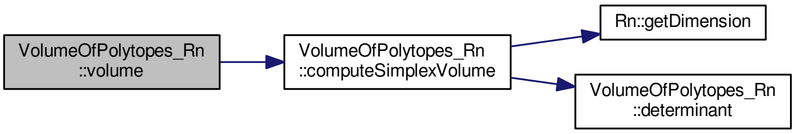

VolumeOfPolytopes_Rn ::splitCloudOfVertices
VolumeOfPolytopes_Rn ::compute
main
politopixAPI::computeVolume
test_main

4.31.3.9 double VolumeOfPolytopes_Rn::volume ( )

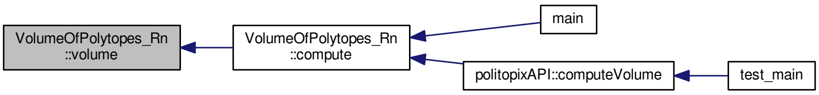
Sum the volumes of all simplices partitionning the polytope.

Definition at line 194 of file VolumeOfPolytopes_Rn.cpp.

Here is the call graph for this function:


VolumeOfPolytopes_Rn ::volume
VolumeOfPolytopes_Rn ::computeSimplexVolume
Rn::getDimension
VolumeOfPolytopes_Rn ::determinant

Here is the caller graph for this function:


VolumeOfPolytopes_Rn ::volume
VolumeOfPolytopes_Rn ::compute
main
politopixAPI::computeVolume
test_main

4.31.4 Member Data Documentation

4.31.4.1 std::vector< PolytopeToSimplexes > VolumeOfPolytopes_Rn::_allSimplices [protected]

List to store all the simplices partitioning the polytope.

Definition at line 251 of file VolumeOfPolytopes_Rn.h.

4.31.4.2 unsigned int VolumeOfPolytopes_Rn::_dimension [protected]

As the algorithm goes down in lower dimensions, we want to store the starting space dimension.

Definition at line 257 of file VolumeOfPolytopes_Rn.h.

4.31.4.3 std::vector< std::vector< unsigned int > > VolumeOfPolytopes_Rn::_facetsByVertices [protected]

The ordered list of all facets stored by vertices.

Definition at line 249 of file VolumeOfPolytopes_Rn.h.

4.31.4.4 unsigned int VolumeOfPolytopes_Rn::_numberOfFacets [protected]

The number of facets of the current polytope.

Definition at line 259 of file VolumeOfPolytopes_Rn.h.

4.31.4.5 unsigned int VolumeOfPolytopes_Rn::_numberOfVertices [protected]

The number of vertices of the current polytope.

Definition at line 261 of file VolumeOfPolytopes_Rn.h.

4.31.4.6 boost::shared_ptr<Polytope_Rn> VolumeOfPolytopes_Rn::_polytope [protected]

The current polytope we are working on.

Definition at line 253 of file VolumeOfPolytopes_Rn.h.

4.31.4.7 std::vector< std::vector< unsigned int > > VolumeOfPolytopes_Rn::_verticesByFacets [protected]

The ordered list of all vertices stored by facets.

Definition at line 247 of file VolumeOfPolytopes_Rn.h.

4.31.4.8 double VolumeOfPolytopes_Rn::_volume [protected]

The volume of the polytope.

Definition at line 255 of file VolumeOfPolytopes_Rn.h.

The documentation for this class was generated from the following files:

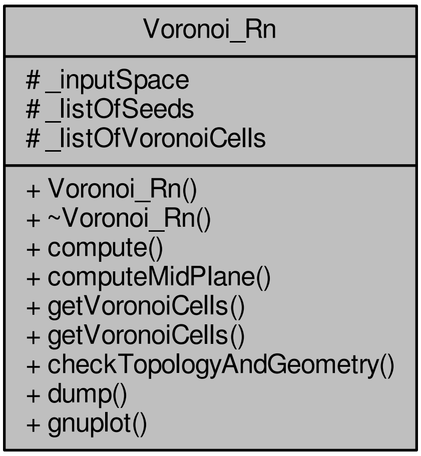
- /home/vindelos/CPP/I2M/politopix/trunk/VolumeOfPolytopes_Rn.h
- /home/vindelos/CPP/I2M/politopix/trunk/VolumeOfPolytopes_Rn.cpp

4.32 Voronoi_Rn Class Reference

Compute a n-dimensional Voronoi diagram. It is a partitioning of a space into regions based on distance to points. Both the space and the list of points are provided as input.

`#include <Voronoi_Rn.h>`

Collaboration diagram for Voronoi_Rn:



Public Member Functions

- Voronoi_Rn (const boost::shared_ptr< Polytope_Rn > &inputSpace, const std::vector< Point_Rn > &list←OfPoints)

  Constructor.
- ∼Voronoi_Rn ()

  Destructor.
- bool compute () throw (std::length_error)

  Run the whole algorithm.
- boost::shared_ptr< HalfSpace_Rn > computeMidPlane (std::vector< Point_Rn >::const_iterator seed1, std::vector< Point_Rn >::const_iterator seed2)

  Compute the half-space containing seed1, in between seed1 and seed2, according to the growing seed property.
- const std::vector
  < boost::shared_ptr
  < Polytope_Rn > > & getVoronoiCells () const
- std::vector< boost::shared_ptr
  < Polytope_Rn > > getVoronoiCells ()
- bool checkTopologyAndGeometry () const throw (std::domain_error)
- void dump (std::ostream &out) const

  Dump the cell structure on the given output.
- void gnuplot (std::ostream &out) const throw (std::domain_error)

Protected Attributes

- const boost::shared_ptr
  < Polytope_Rn > & _inputSpace

  The original space to be divided.
- const std::vector< Point_Rn > & _listOfSeeds

  The list of input points.
- std::vector< boost::shared_ptr
  < Polytope_Rn > > _listOfVoronoiCells

  The list of polytopes partitioning the whole space.

4.32.1 Detailed Description

Compute a n-dimensional Voronoi diagram. It is a partitioning of a space into regions based on distance to points. Both the space and the list of points are provided as input.

Definition at line 38 of file Voronoi_Rn.h.

4.32.2 Constructor & Destructor Documentation

4.32.2.1 Voronoi_Rn::Voronoi_Rn (const boost::shared_ptr< **Polytope_Rn** > & *inputSpace,* const std::vector< **Point_Rn** > & *listOfPoints*)

Constructor.

Constructor

Definition at line 32 of file Voronoi_Rn.cpp.

4.32.2.2 Voronoi_Rn::∼Voronoi_Rn ( ) [inline]

Destructor.

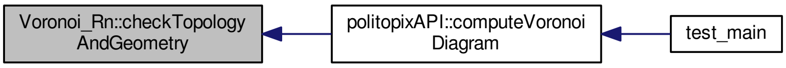
Definition at line 48 of file Voronoi_Rn.h.

4.32.3 Member Function Documentation

4.32.3.1 bool Voronoi_Rn::checkTopologyAndGeometry ( ) const throw std::domain_error)

Definition at line 139 of file Voronoi_Rn.cpp.

Here is the caller graph for this function:

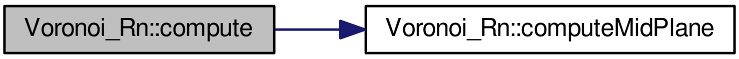

Voronoi_Rn::checkTopology AndGeometry
politopixAPI::computeVoronoi Diagram
test_main

4.32.3.2 bool Voronoi_Rn::compute ( ) throw std::length_error)

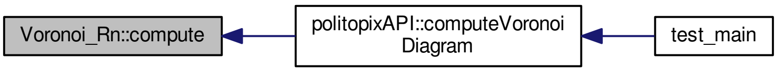
Run the whole algorithm.

Definition at line 36 of file Voronoi_Rn.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

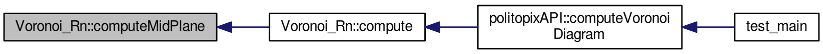


4.32.3.3 boost::shared_ptr< HalfSpace_Rn > Voronoi_Rn::computeMidPlane (std::vector< Point_Rn >::const_iterator *seed1,* std::vector< Point_Rn >::const_iterator *seed2*)

Compute the half-space containing seed1, in between seed1 and seed2, according to the growing seed property.

Definition at line 122 of file Voronoi_Rn.cpp.

Here is the caller graph for this function:


Voronoi_Rn::computeMidPlane
Voronoi_Rn::compute
politopixAPI::computeVoronoi Diagram
test_main

4.32.3.4 void Voronoi_Rn::dump (std::ostream & *out*) const

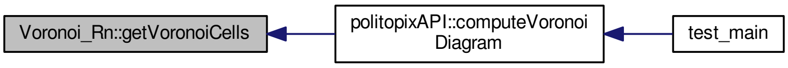
Dump the cell structure on the given output.

Definition at line 144 of file Voronoi_Rn.cpp.

4.32.3.5 const std::vector< boost::shared_ptr<Polytope_Rn> >& Voronoi_Rn::getVoronoiCells ( ) const [inline]

Definition at line 56 of file Voronoi_Rn.h.

Here is the caller graph for this function:


Voronoi_Rn::getVoronoiCells
politopixAPI::computeVoronoi Diagram
test_main

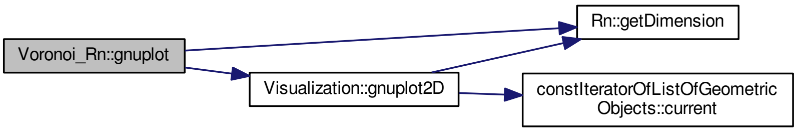
4.32.3.6 std::vector< boost::shared_ptr<Polytope_Rn> > Voronoi_Rn::getVoronoiCells ( ) [inline]

Definition at line 57 of file Voronoi_Rn.h.

4.32.3.7 void Voronoi_Rn::gnuplot (std::ostream & *out*) const throw std::domain_error)

Definition at line 164 of file Voronoi_Rn.cpp.

Here is the call graph for this function:



4.32.4 Member Data Documentation

4.32.4.1 const boost::shared_ptr<Polytope_Rn>& Voronoi_Rn::_inputSpace [protected]

The original space to be divided.

Definition at line 69 of file Voronoi_Rn.h.

4.32.4.2 const std::vector< Point_Rn >& Voronoi_Rn::_listOfSeeds [protected]

The list of input points.

Definition at line 71 of file Voronoi_Rn.h.

4.32.4.3 std::vector< boost::shared_ptr<Polytope_Rn> > Voronoi_Rn::_listOfVoronoiCells [protected]

The list of polytopes partitioning the whole space.

Definition at line 73 of file Voronoi_Rn.h.

The documentation for this class was generated from the following files:

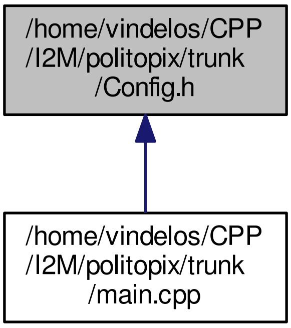
- /home/vindelos/CPP/I2M/politopix/trunk/Voronoi_Rn.h
- /home/vindelos/CPP/I2M/politopix/trunk/Voronoi_Rn.cpp

Chapter 5

File Documentation

5.1 /home/vindelos/CPP/I2M/politopix/trunk/Config.h File Reference

This graph shows which files directly or indirectly include this file:



Macros

- #define politopix_VERSION_MAJOR 4
- #define politopix_VERSION_MINOR 0
- #define politopix_VERSION_PATCH 0

5.1.1 Macro Definition Documentation

5.1.1.1 #define politopix_VERSION_MAJOR 4

Definition at line 2 of file Config.h.

5.1.1.2 #define politopix_VERSION_MINOR 0

Definition at line 3 of file Config.h.

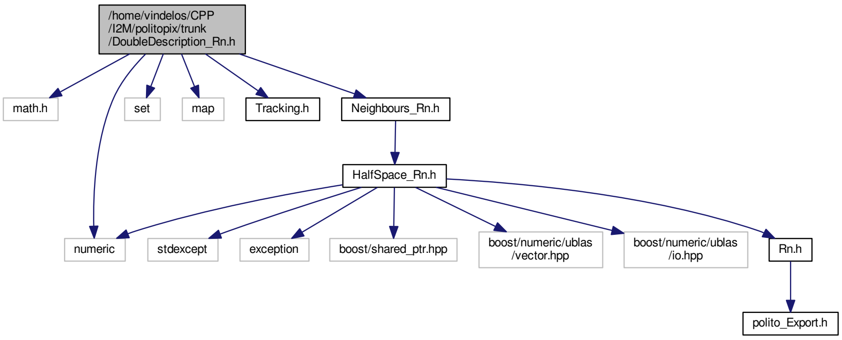
5.1.1.3 #define politopix_VERSION_PATCH 0

Definition at line 4 of file Config.h.

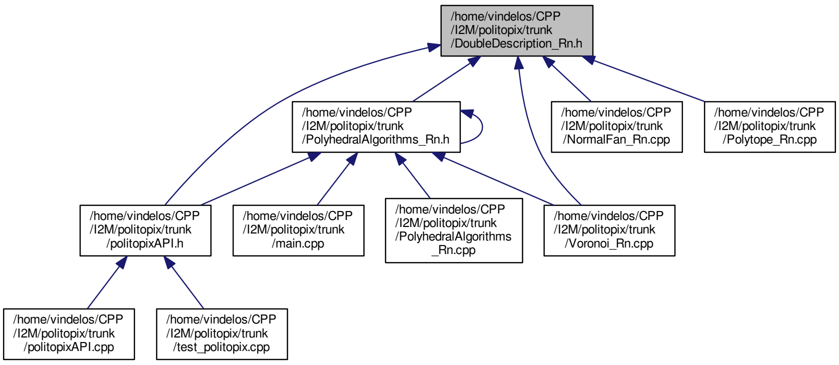
5.2 /home/vindelos/CPP/I2M/politopix/trunk/DoubleDescription_Rn.h File Reference

```
#include <math.h>
#include <numeric>
#include <set>
#include <map>
#include "Tracking.h"
#include "Neighbours_Rn.h"
```

Include dependency graph for DoubleDescription_Rn.h:



This graph shows which files directly or indirectly include this file:



Classes

- class DoubleDescription< POLYHEDRON, ITERATOR, REDUNDANCY_PROCESSING >

  The algorithm implemented here is an incremental algorithm as mentioned in How Good are Convex Hull Algorithms? (1997) by ***David Avis*** *and* ***David Bremner****. Specific and efficient implementations can be found in The double description method revisited (1996) written by* ***Komei Fukuda*** *and* ***Alain Prodon*** *.*
  Incremental algorithms for the vertex enumeration problem compute the vertex description by intersecting the defining

half-spaces sequentially. An initial simplex is constructed from a subset of n+1 half-spaces and its vertices and 1-skeleton are computed. Additional half-spaces are introduced sequentially and the vertex description and 1-skeleton are updated at each stage. Essentially such an update amounts to identifying and removing all vertices that are not contained in the new half-space, introducing new vertices for all intersections between edges and the bounding hyperplane of the new half-space, and generating the new edges between these new vertices.
This algorithm can be instantiated by polytopes or polyhedral cones, and as a second argument can be instantiated by iterators such as minindex, lexmin, lexmax.

- class NoRedundancyProcessing< POLYHEDRON >

  Makes the assumption we do not need to process redundant half-spaces in a specific way.

- class StrongRedundancyProcessing< POLYHEDRON >

  This class can be more time-consuming than WeakRedundancyProcessing or NoRedundancyProcessing because it will perform extra checks in the process of intersecting half-spaces. To determine if two vertices are neighbors it will make sure not to count half-spaces marked as redundant.

5.2.1 Detailed Description

Author

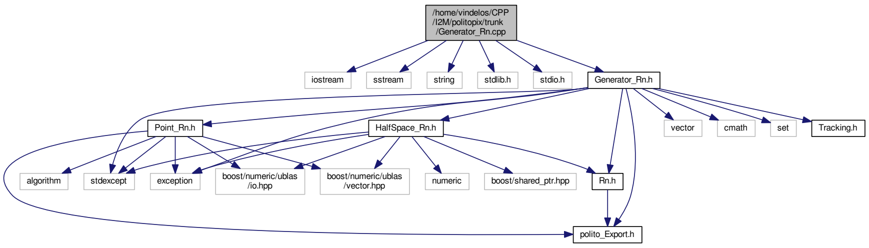
Delos Vincent (vincent.delos@i2m.u-bordeaux1.fr)

Definition in file DoubleDescription_Rn.h.

5.3 /home/vindelos/CPP/I2M/politopix/trunk/Generator_Rn.cpp File Reference

```
#include <iostream>
#include <sstream>
#include <string>
#include <stdlib.h>
#include <stdio.h>
#include "Generator_Rn.h"
```

Include dependency graph for Generator_Rn.cpp:



5.3.1 Detailed Description

Author

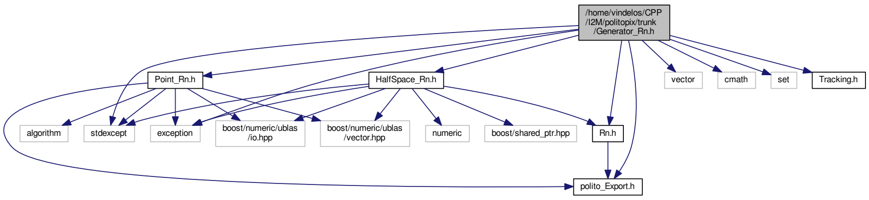
Delos Vincent (v.delos@i2m.u-bordeaux1.fr)

Definition in file Generator_Rn.cpp.

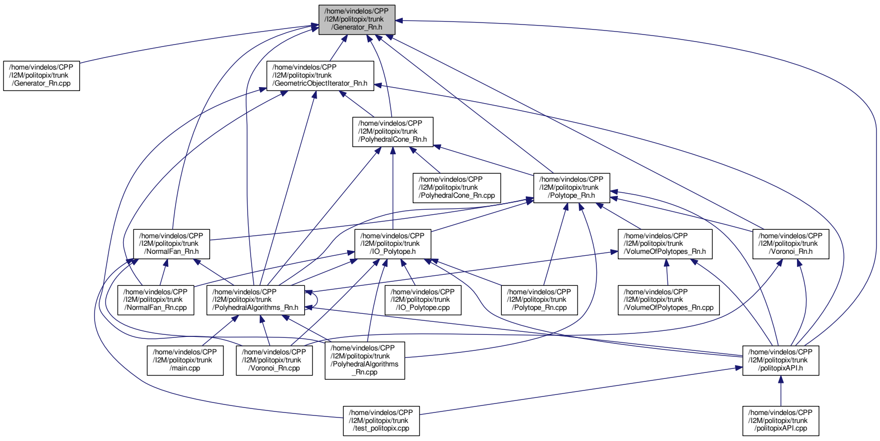
5.4 /home/vindelos/CPP/I2M/politopix/trunk/Generator_Rn.h File Reference

```
#include <stdexcept>
#include <exception>
#include <vector>
#include <cmath>
#include <set>
#include "polito_Export.h"
#include "Rn.h"
#include "Tracking.h"
#include "Point_Rn.h"
#include "HalfSpace_Rn.h"
```

Include dependency graph for Generator_Rn.h:



This graph shows which files directly or indirectly include this file:



Classes

- class Generator_Rn

  A n-coordinates generator, which can be a vertex or an edge whether it is contained by a polytope or a polyhedral cone. It contains all of its support facets.

- class Generator_Rn_SD

  A n-coordinates generator for internal data structure. It can be a vertex or an edge whether it is embedded in a polytope or a polyhedral cone. It contains all of its support facets.

5.4.1 Detailed Description

Author

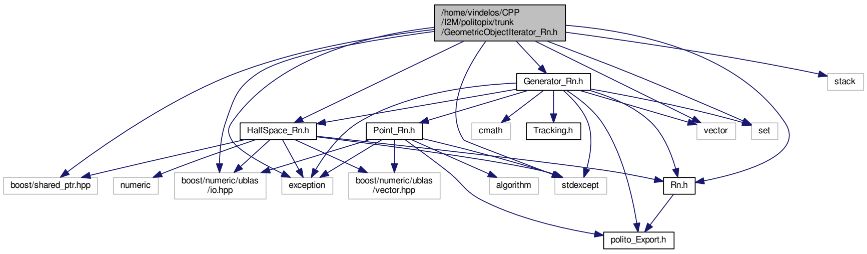
Delos Vincent (v.delos@i2m.u-bordeaux1.fr)

Definition in file Generator_Rn.h.

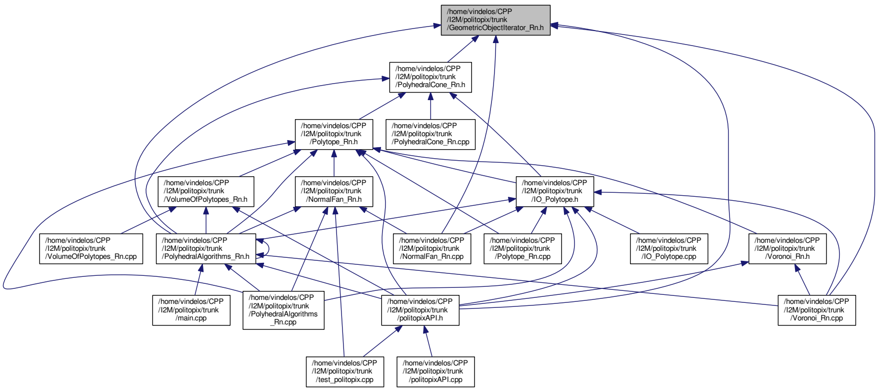
5.5 /home/vindelos/CPP/I2M/politopix/trunk/GeometricObjectIterator_Rn.h File Reference

```
#include <boost/numeric/ublas/io.hpp>
#include <boost/shared_ptr.hpp>
#include <stdexcept>
#include <exception>
#include <vector>
#include <stack>
#include <set>
#include "Generator_Rn.h"
#include "HalfSpace_Rn.h"
#include "Rn.h"
```

Include dependency graph for GeometricObjectIterator_Rn.h:



This graph shows which files directly or indirectly include this file:



Classes

- class listOfGeometricObjects< GEOMETRIC_OBJECT >

  This class is designed to contain the list of all generators or half-spaces representing a polytope or a polyhedral cone.

- class constIteratorOfListOfGeometricObjects< GEOMETRIC_OBJECT >

  This class is designed to run the list of all geometric objects representing a polytope.
- class lexIteratorOfListOfGeometricObjects< GEOMETRIC_OBJECT >

  Insert the half-spaces in the list in a lexicographically order, whether min or max.
- class lexminIteratorOfListOfGeometricObjects< GEOMETRIC_OBJECT >

  Insert the half-spaces in the list in lexicographically increasing order.
- class lexmaxIteratorOfListOfGeometricObjects< GEOMETRIC_OBJECT >

  Insert the half-spaces in the list in lexicographically decreasing order.

5.5.1 Detailed Description

Author

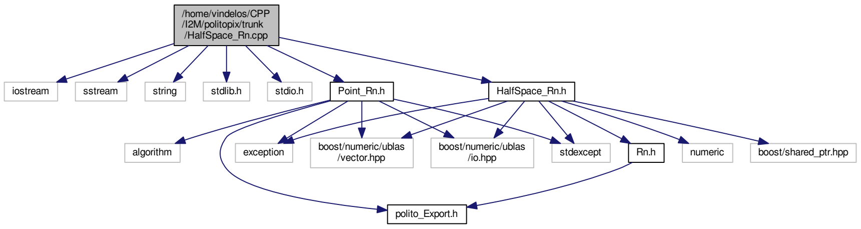
Delos Vincent (`v.delos@i2m.u-bordeaux1.fr`)

Definition in file GeometricObjectIterator_Rn.h.

5.6 /home/vindelos/CPP/I2M/politopix/trunk/HalfSpace_Rn.cpp File Reference

```
#include <iostream>
#include <sstream>
#include <string>
#include <stdlib.h>
#include <stdio.h>
#include "Point_Rn.h"
#include "HalfSpace_Rn.h"
```

Include dependency graph for HalfSpace_Rn.cpp:



5.6.1 Detailed Description

Author

Delos Vincent (`v.delos@i2m.u-bordeaux1.fr`)

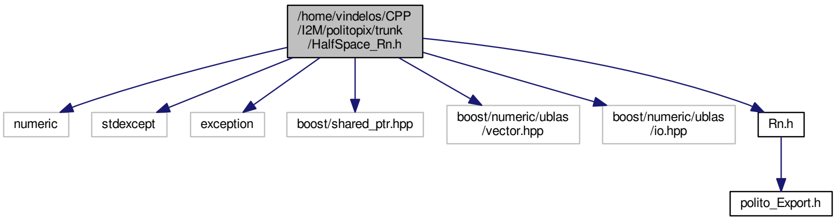
Definition in file HalfSpace_Rn.cpp.

5.7 /home/vindelos/CPP/I2M/politopix/trunk/HalfSpace_Rn.h File Reference

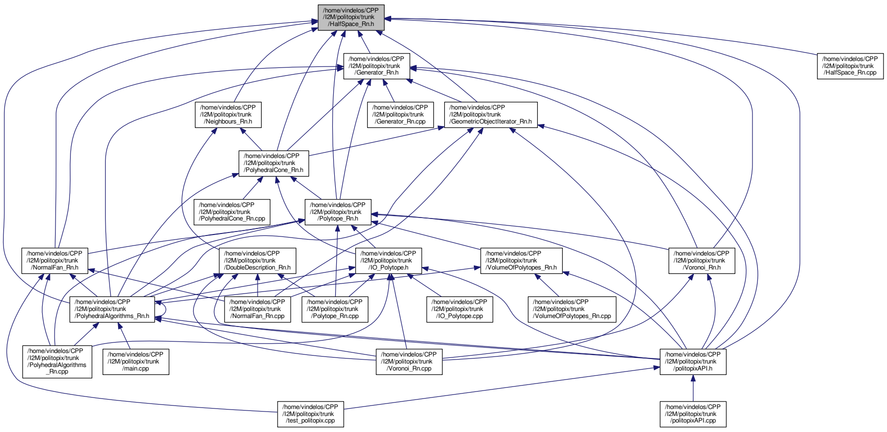
```
#include <numeric>
```

```
#include <stdexcept>
#include <exception>
#include <boost/shared_ptr.hpp>
#include <boost/numeric/ublas/vector.hpp>
#include <boost/numeric/ublas/io.hpp>
#include "Rn.h"
```

Include dependency graph for HalfSpace_Rn.h:



This graph shows which files directly or indirectly include this file:



Classes

- class HalfSpace_Rn

  A half-space whose frontier is a linear (n-1) dimension space.
  _constant + _coefficients[0].x1 + ... + _coefficients[n-1].xn >= 0.

5.7.1 Detailed Description

Author

Delos Vincent (`v.delos@i2m.u-bordeaux1.fr`)

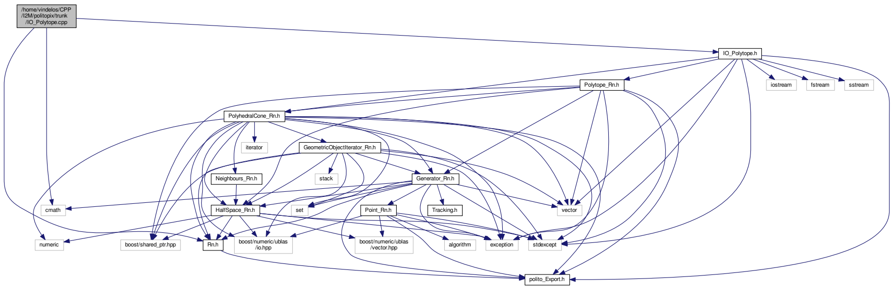
Definition in file HalfSpace_Rn.h.

5.8 /home/vindelos/CPP/I2M/politopix/trunk/IO_Polytope.cpp File Reference

```
#include <cmath>
```

```
#include "Rn.h"
#include "IO_Polytope.h"
```

Include dependency graph for IO_Polytope.cpp:



5.8.1 Detailed Description

Author

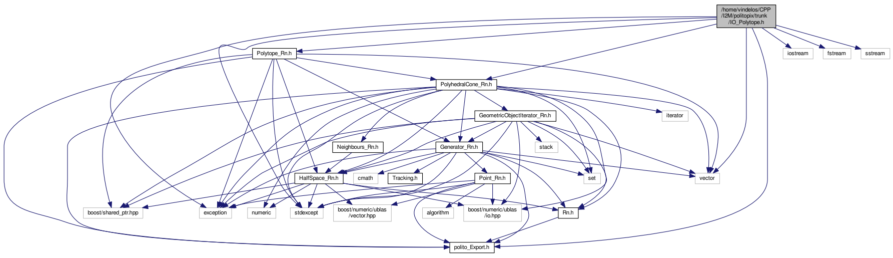
Delos Vincent (v.delos@i2m.u-bordeaux1.fr)

Definition in file IO_Polytope.cpp.

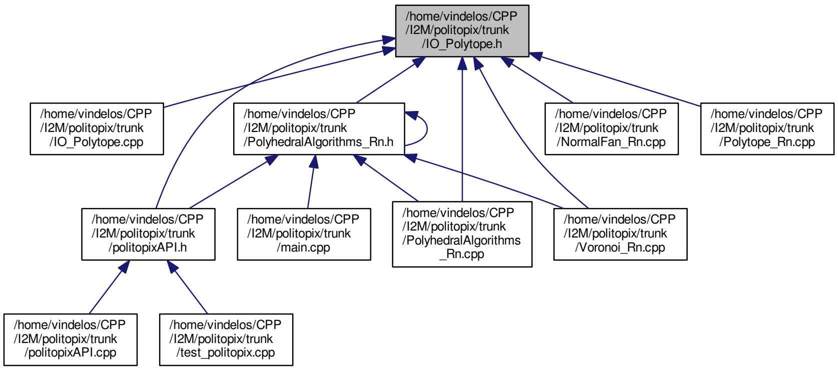
5.9 /home/vindelos/CPP/I2M/politopix/trunk/IO_Polytope.h File Reference

```
#include <stdexcept>
#include <exception>
#include <iostream>
#include <fstream>
#include <sstream>
#include <vector>
#include "polito_Export.h"
#include "Polytope_Rn.h"
#include "PolyhedralCone_Rn.h"
```

Include dependency graph for IO_Polytope.h:



This graph shows which files directly or indirectly include this file:


/home/vindelos/CPP/I2M/politopix/trunk/IO_Polytope.h
/home/vindelos/CPP/I2M/politopix/trunk/IO_Polytope.cpp
/home/vindelos/CPP/I2M/politopix/trunk/PolyhedralAlgorithms_Rn.h
/home/vindelos/CPP/I2M/politopix/trunk/NormalFan_Rn.cpp
/home/vindelos/CPP/I2M/politopix/trunk/Polytope_Rn.cpp
/home/vindelos/CPP/I2M/politopix/trunk/politopixAPI.h
/home/vindelos/CPP/I2M/politopix/trunk/main.cpp
/home/vindelos/CPP/I2M/politopix/trunk/PolyhedralAlgorithms_Rn.cpp
/home/vindelos/CPP/I2M/politopix/trunk/Voronoi_Rn.cpp
/home/vindelos/CPP/I2M/politopix/trunk/politopixAPI.cpp
/home/vindelos/CPP/I2M/politopix/trunk/test_politopix.cpp

Classes

- class IO_Polytope

  Read/write polytopes.
  The way we store polytopes :
  1st line : comments = "# Dimension NumberOfHalfspaces NumberOfGenerators"
  2nd line : cartesian_space_dimension number_of_facets number_of_generators
  3rd line : comments = "# HALFSPACES : a0 + a1.x1 + ... + an.xn >= 0."
  4th line : a00 a10 ... an0
  k-th line : a0k a1k ... ank
  (k+1)-th line : comments = "# GENERATORS : V = (v1, ..., vn)"
  (k+2)-th line : v11 ... v1n
  l-th line : vl1 ... vln
  (l+1)-th line : comments = "# FACETS PER GENERATOR : {Fi1, Fi2, ...}"
  (l+2)-th line the neigh : Fr ... Fs
  m-th line : Fs ... Ft
  If (number_of_vertices == 0) then compute the vertices from the
  facets including the polytope into a huge cube containing it.
  In this case the blocks "GENERATORS" and "FACETS PER GENERATOR" are ignored.

5.9.1 Detailed Description

Author

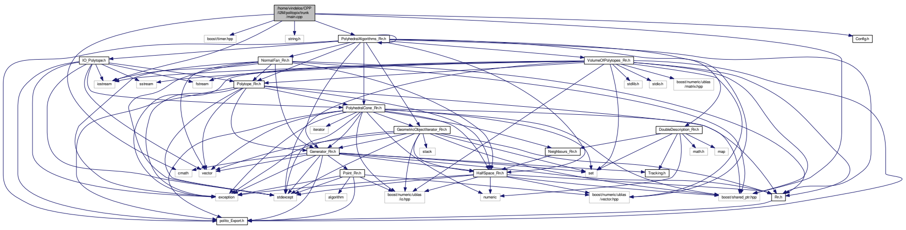
Delos Vincent (v.delos@i2m.u-bordeaux1.fr)

Definition in file IO_Polytope.h.

5.10 /home/vindelos/CPP/I2M/politopix/trunk/main.cpp File Reference

```
#include <boost/shared_ptr.hpp>
#include <boost/timer.hpp>
#include <iostream>
#include <string.h>
#include <cmath>
#include "PolyhedralAlgorithms_Rn.h"
#include "Config.h"
```

Include dependency graph for main.cpp:



Functions

- int main (int argc, char ∗argv[ ])

5.10.1 Detailed Description

Author

Delos Vincent (v.delos@i2m.u-bordeaux1.fr)

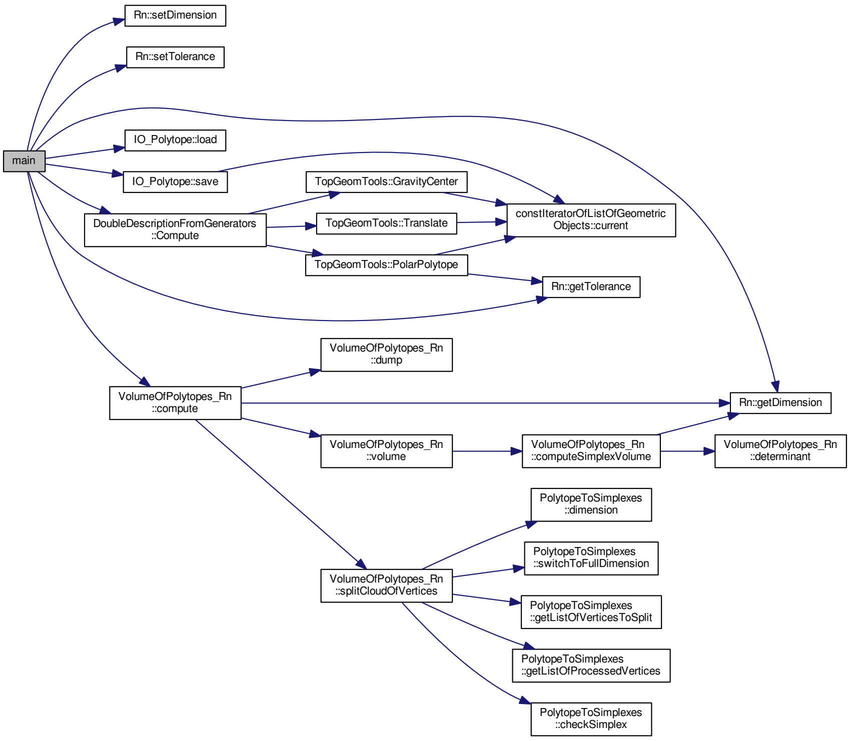
Definition in file main.cpp.

5.10.2 Function Documentation

5.10.2.1 int main (int *argc,* char ∗ *argv[ ]*)

Definition at line 59 of file main.cpp.

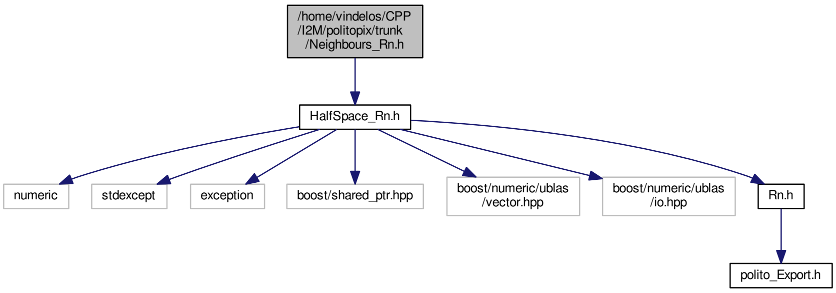
Here is the call graph for this function:



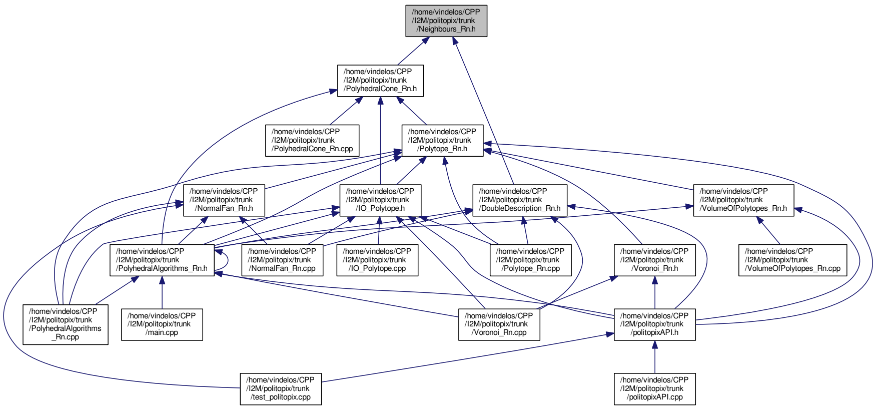
5.11 /home/vindelos/CPP/I2M/politopix/trunk/Neighbours_Rn.h File Reference

```
#include "HalfSpace_Rn.h"
```

Include dependency graph for Neighbours_Rn.h:



This graph shows which files directly or indirectly include this file:


/home/vindelos/CPP /I2M/politopix/trunk /Neighbours_Rn.h
/home/vindelos/CPP /I2M/politopix/trunk /PolyhedralCone_Rn.h
/home/vindelos/CPP /I2M/politopix/trunk /PolyhedralCone_Rn.cpp
/home/vindelos/CPP /I2M/politopix/trunk /Polytope_Rn.h
/home/vindelos/CPP /I2M/politopix/trunk /NormalFan_Rn.h
/home/vindelos/CPP /I2M/politopix/trunk /IO_Polytope.h
/home/vindelos/CPP /I2M/politopix/trunk /DoubleDescription_Rn.h
/home/vindelos/CPP /I2M/politopix/trunk /VolumeOfPolytopes_Rn.h
/home/vindelos/CPP /I2M/politopix/trunk /PolyhedralAlgorithms_Rn.h
/home/vindelos/CPP /I2M/politopix/trunk /NormalFan_Rn.cpp
/home/vindelos/CPP /I2M/politopix/trunk /IO_Polytope.cpp
/home/vindelos/CPP /I2M/politopix/trunk /Polytope_Rn.cpp
/home/vindelos/CPP /I2M/politopix/trunk /Voronoi_Rn.h
/home/vindelos/CPP /I2M/politopix/trunk /VolumeOfPolytopes_Rn.cpp
/home/vindelos/CPP /I2M/politopix/trunk /PolyhedralAlgorithms _Rn.cpp
/home/vindelos/CPP /I2M/politopix/trunk /main.cpp
/home/vindelos/CPP /I2M/politopix/trunk /Voronoi_Rn.cpp
/home/vindelos/CPP /I2M/politopix/trunk /politopixAPI.h
/home/vindelos/CPP /I2M/politopix/trunk /test_politopix.cpp
/home/vindelos/CPP /I2M/politopix/trunk /politopixAPI.cpp

Classes

- class Neighbours_Rn

  Class dedicated to degeneration processing when looking for neighbours.
  Let A be a polytope of $\mathbb{R}^n, A = H_1^+ \cap H_2^+ \cap ... H_{n-1}^+ \cap ... H_k^+$ *where k>n.*
  Let [v_i, v_j] be a segment of two vertices of A such as:

5.11.1 Detailed Description

Author

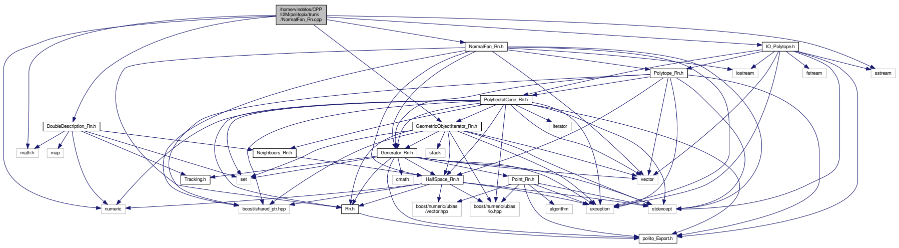
Delos Vincent (v.delos@i2m.u-bordeaux1.fr)

Definition in file Neighbours_Rn.h.

5.12 /home/vindelos/CPP/I2M/politopix/trunk/NormalFan_Rn.cpp File Reference

```
#include "GeometricObjectIterator_Rn.h"
#include "DoubleDescription_Rn.h"
#include "NormalFan_Rn.h"
#include "IO_Polytope.h"
#include <math.h>
#include <numeric>
#include <sstream>
```

Include dependency graph for NormalFan_Rn.cpp:



5.12.1 Detailed Description

Author

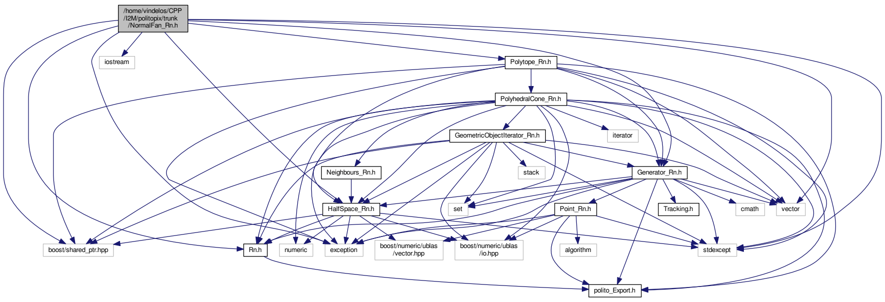
Delos Vincent (v.delos@i2m.u-bordeaux1.fr)

Definition in file NormalFan_Rn.cpp.

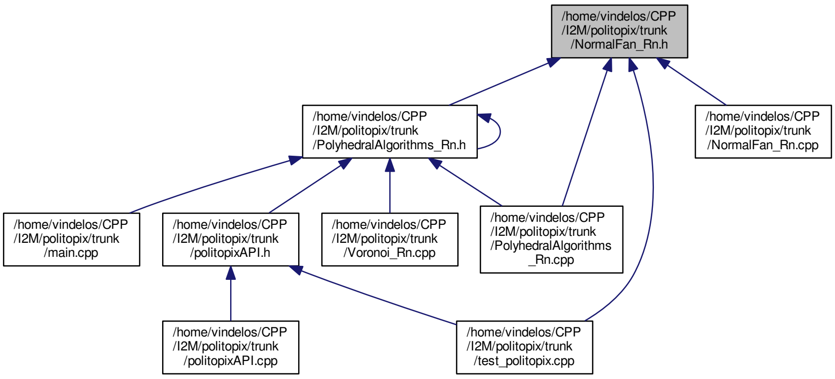
5.13 /home/vindelos/CPP/I2M/politopix/trunk/NormalFan_Rn.h File Reference

```
#include <boost/shared_ptr.hpp>
#include <stdexcept>
#include <exception>
#include <iostream>
#include <vector>
#include "Rn.h"
#include "Generator_Rn.h"
#include "HalfSpace_Rn.h"
#include "Polytope_Rn.h"
```

Include dependency graph for NormalFan_Rn.h:



This graph shows which files directly or indirectly include this file:


/home/vindelos/CPP/I2M/politopix/trunk/NormalFan_Rn.h
/home/vindelos/CPP/I2M/politopix/trunk/PolyhedralAlgorithms_Rn.h
/home/vindelos/CPP/I2M/politopix/trunk/NormalFan_Rn.cpp
/home/vindelos/CPP/I2M/politopix/trunk/main.cpp
/home/vindelos/CPP/I2M/politopix/trunk/politopixAPI.h
/home/vindelos/CPP/I2M/politopix/trunk/Voronoi_Rn.cpp
/home/vindelos/CPP/I2M/politopix/trunk/PolyhedralAlgorithms_Rn.cpp
/home/vindelos/CPP/I2M/politopix/trunk/politopixAPI.cpp
/home/vindelos/CPP/I2M/politopix/trunk/test_politopix.cpp

Classes

- class NormalFan_Rn

  Model a normal fan.

5.13.1 Detailed Description

Author

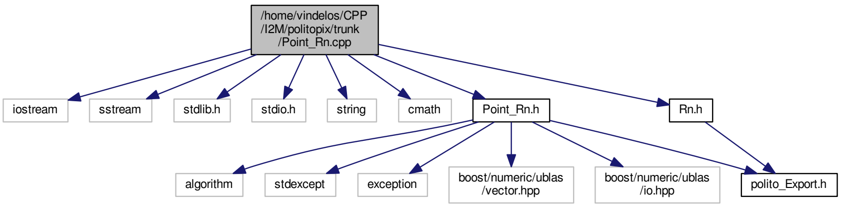
Delos Vincent (v.delos@i2m.u-bordeaux1.fr)

Definition in file NormalFan_Rn.h.

5.14 /home/vindelos/CPP/I2M/politopix/trunk/Point_Rn.cpp File Reference

```
#include <iostream>
#include <sstream>
#include <stdlib.h>
#include <stdio.h>
#include <string>
#include <cmath>
#include "Point_Rn.h"
#include "Rn.h"
```

Include dependency graph for Point_Rn.cpp:



5.14.1 Detailed Description

Author

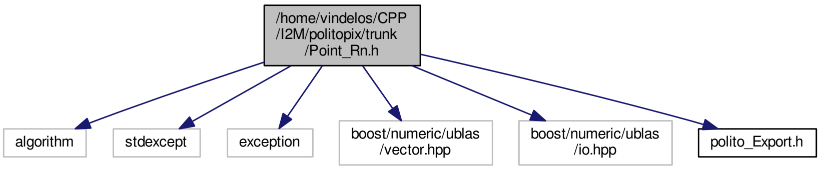
Delos Vincent (v.delos@i2m.u-bordeaux1.fr)

Definition in file Point_Rn.cpp.

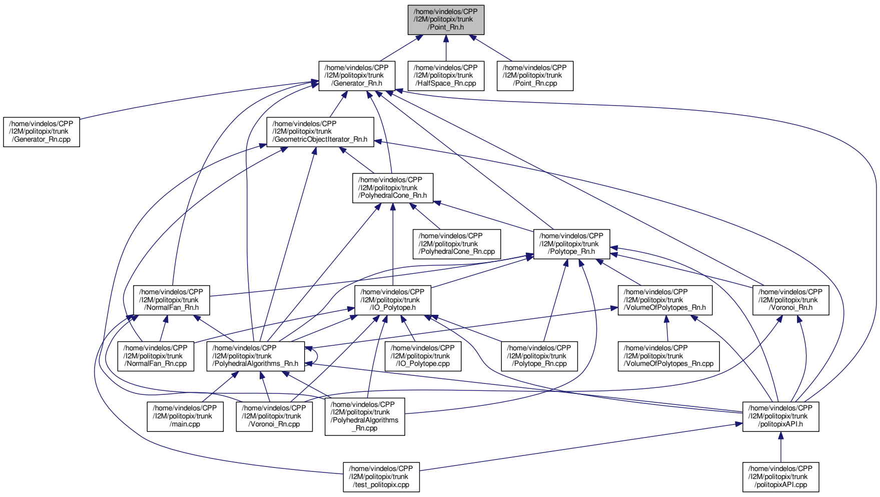
5.15 /home/vindelos/CPP/I2M/politopix/trunk/Point_Rn.h File Reference

```
#include <algorithm>
#include <stdexcept>
#include <exception>
#include <boost/numeric/ublas/vector.hpp>
#include <boost/numeric/ublas/io.hpp>
#include "polito_Export.h"
```

Include dependency graph for Point_Rn.h:


/home/vindelos/CPP/I2M/politopix/trunk/Point_Rn.h
algorithm
stdexcept
exception
boost/numeric/ublas/vector.hpp
boost/numeric/ublas/io.hpp
polito_Export.h

This graph shows which files directly or indirectly include this file:


/home/vindelos/CPP/I2M/politopix/trunk/Point_Rn.h
/home/vindelos/CPP/I2M/politopix/trunk/Generator_Rn.h
/home/vindelos/CPP/I2M/politopix/trunk/HalfSpace_Rn.cpp
/home/vindelos/CPP/I2M/politopix/trunk/Point_Rn.cpp
/home/vindelos/CPP/I2M/politopix/trunk/Generator_Rn.cpp
/home/vindelos/CPP/I2M/politopix/trunk/GeometricObjectIterator_Rn.h
/home/vindelos/CPP/I2M/politopix/trunk/PolyhedralCone_Rn.h
/home/vindelos/CPP/I2M/politopix/trunk/PolyhedralCone_Rn.cpp
/home/vindelos/CPP/I2M/politopix/trunk/Polytope_Rn.h
/home/vindelos/CPP/I2M/politopix/trunk/NormalFan_Rn.h
/home/vindelos/CPP/I2M/politopix/trunk/IO_Polytope.h
/home/vindelos/CPP/I2M/politopix/trunk/VolumeOfPolytopes_Rn.h
/home/vindelos/CPP/I2M/politopix/trunk/Voronoi_Rn.h
/home/vindelos/CPP/I2M/politopix/trunk/NormalFan_Rn.cpp
/home/vindelos/CPP/I2M/politopix/trunk/PolyhedralAlgorithms_Rn.h
/home/vindelos/CPP/I2M/politopix/trunk/IO_Polytope.cpp
/home/vindelos/CPP/I2M/politopix/trunk/Polytope_Rn.cpp
/home/vindelos/CPP/I2M/politopix/trunk/VolumeOfPolytopes_Rn.cpp
/home/vindelos/CPP/I2M/politopix/trunk/main.cpp
/home/vindelos/CPP/I2M/politopix/trunk/Voronoi_Rn.cpp
/home/vindelos/CPP/I2M/politopix/trunk/PolyhedralAlgorithms_Rn.cpp
/home/vindelos/CPP/I2M/politopix/trunk/politopixAPI.h
/home/vindelos/CPP/I2M/politopix/trunk/test_politopix.cpp
/home/vindelos/CPP/I2M/politopix/trunk/politopixAPI.cpp

Classes

- class Point_Rn

  Creation of a n-coordinate geometric point designed to be shared by its neighbour faces.

5.15.1 Detailed Description

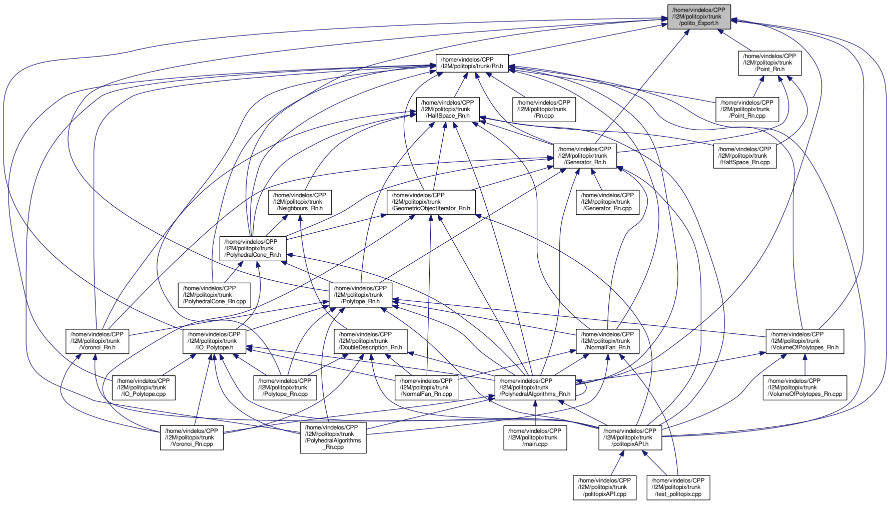
Author

Delos Vincent (v.delos@i2m.u-bordeaux1.fr)

Definition in file Point_Rn.h.

5.16 /home/vindelos/CPP/I2M/politopix/trunk/polito_Export.h File Reference

This graph shows which files directly or indirectly include this file:


/home/vindelos/CPP /I2M/politopix/trunk /polito_Export.h
/home/vindelos/CPP /I2M/politopix/trunk/Rn.h
/home/vindelos/CPP /I2M/politopix/trunk /Point_Rn.h
/home/vindelos/CPP /I2M/politopix/trunk /HalfSpace_Rn.h
/home/vindelos/CPP /I2M/politopix/trunk /Rn.cpp
/home/vindelos/CPP /I2M/politopix/trunk /Point_Rn.cpp
/home/vindelos/CPP /I2M/politopix/trunk /Generator_Rn.h
/home/vindelos/CPP /I2M/politopix/trunk /HalfSpace_Rn.cpp
/home/vindelos/CPP /I2M/politopix/trunk /Neighbours_Rn.h
/home/vindelos/CPP /I2M/politopix/trunk /GeometricObjectIterator_Rn.h
/home/vindelos/CPP /I2M/politopix/trunk /Generator_Rn.cpp
/home/vindelos/CPP /I2M/politopix/trunk /PolyhedralCone_Rn.h
/home/vindelos/CPP /I2M/politopix/trunk /PolyhedralCone_Rn.cpp
/home/vindelos/CPP /I2M/politopix/trunk /Polytope_Rn.h
/home/vindelos/CPP /I2M/politopix/trunk /Voronoi_Rn.h
/home/vindelos/CPP /I2M/politopix/trunk /IO_Polytope.h
/home/vindelos/CPP /I2M/politopix/trunk /DoubleDescription_Rn.h
/home/vindelos/CPP /I2M/politopix/trunk /NormalFan_Rn.h
/home/vindelos/CPP /I2M/politopix/trunk /VolumeOfPolytopes_Rn.h
/home/vindelos/CPP /I2M/politopix/trunk /IO_Polytope.cpp
/home/vindelos/CPP /I2M/politopix/trunk /Polytope_Rn.cpp
/home/vindelos/CPP /I2M/politopix/trunk /NormalFan_Rn.cpp
/home/vindelos/CPP /I2M/politopix/trunk /PolyhedralAlgorithms_Rn.h
/home/vindelos/CPP /I2M/politopix/trunk /VolumeOfPolytopes_Rn.cpp
/home/vindelos/CPP /I2M/politopix/trunk /Voronoi_Rn.cpp
/home/vindelos/CPP /I2M/politopix/trunk /PolyhedralAlgorithms _Rn.cpp
/home/vindelos/CPP /I2M/politopix/trunk /main.cpp
/home/vindelos/CPP /I2M/politopix/trunk /politopixAPI.h
/home/vindelos/CPP /I2M/politopix/trunk /politopixAPI.cpp
/home/vindelos/CPP /I2M/politopix/trunk /test_politopix.cpp

Macros

- #define polito_EXPORT __attribute__((visibility("default")))
- #define POLITO_NO_EXPORT __attribute__((visibility("hidden")))
- #define POLITO_DEPRECATED __attribute__ ((__deprecated__))
- #define POLITO_DEPRECATED_EXPORT polito_EXPORT __attribute__ ((__deprecated__))
- #define POLITO_DEPRECATED_NO_EXPORT POLITO_NO_EXPORT __attribute__ ((__deprecated__))
- #define DEFINE_NO_DEPRECATED 0

5.16.1 Macro Definition Documentation

5.16.1.1 #define DEFINE_NO_DEPRECATED 0

Definition at line 30 of file polito_Export.h.

5.16.1.2 #define POLITO_DEPRECATED __attribute__ ((__deprecated__))

Definition at line 25 of file polito_Export.h.

5.16.1.3 #define POLITO_DEPRECATED_EXPORT polito_EXPORT __attribute__ ((__deprecated__))

Definition at line 26 of file polito_Export.h.

5.16.1.4 #define POLITO_DEPRECATED_NO_EXPORT POLITO_NO_EXPORT __attribute__ ((__deprecated__))

Definition at line 27 of file polito_Export.h.

5.16.1.5 #define polito_EXPORT __attribute__((visibility("default")))

Definition at line 15 of file polito_Export.h.

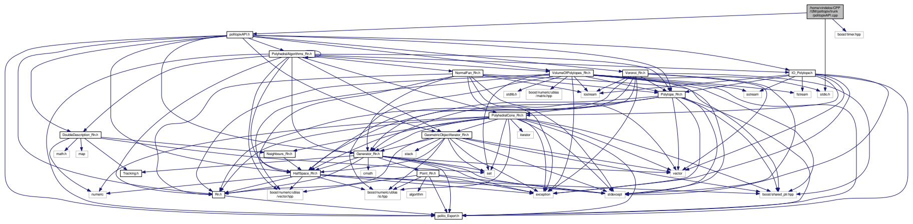
5.16.1.6 #define POLITO_NO_EXPORT __attribute__((visibility("hidden")))

Definition at line 20 of file polito_Export.h.

5.17 /home/vindelos/CPP/I2M/politopix/trunk/politopixAPI.cpp File Reference

```
#include <stdio.h>
#include <boost/timer.hpp>
#include "politopixAPI.h"
```

Include dependency graph for politopixAPI.cpp:



5.17.1 Detailed Description

Author

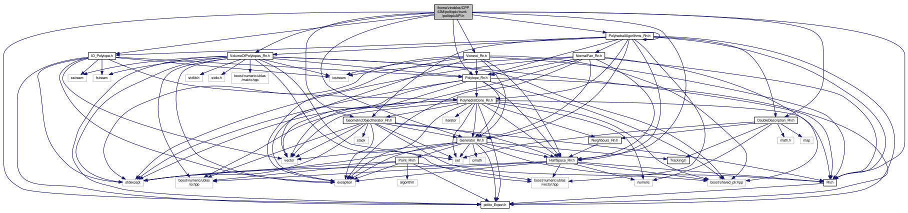
Delos Vincent (v.delos@i2m.u-bordeaux1.fr)

Definition in file politopixAPI.cpp.

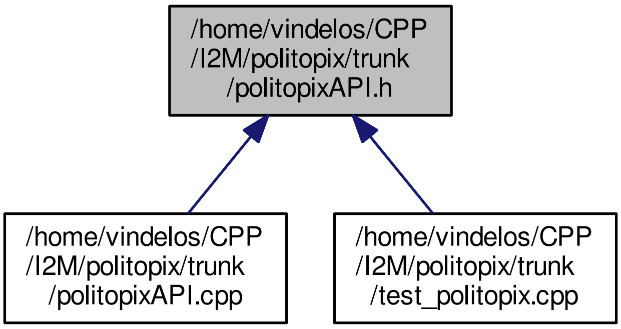
5.18 /home/vindelos/CPP/I2M/politopix/trunk/politopixAPI.h File Reference

```
#include "polito_Export.h"
#include "Rn.h"
#include "Voronoi_Rn.h"
#include "Polytope_Rn.h"
#include "IO_Polytope.h"
#include "Generator_Rn.h"
#include "HalfSpace_Rn.h"
#include "VolumeOfPolytopes_Rn.h"
#include "DoubleDescription_Rn.h"
#include "PolyhedralAlgorithms_Rn.h"
#include "GeometricObjectIterator_Rn.h"
```

Include dependency graph for politopixAPI.h:



This graph shows which files directly or indirectly include this file:



Classes

- class politopixAPI

Macros

- #define TEST_OK 0
- #define TEST_KO -1

5.18.1 Detailed Description

Author

Delos Vincent (`v.delos@i2m.u-bordeaux1.fr`)

Definition in file politopixAPI.h.

5.18.2 Macro Definition Documentation

5.18.2.1 #define TEST_KO -1

Definition at line 39 of file politopixAPI.h.

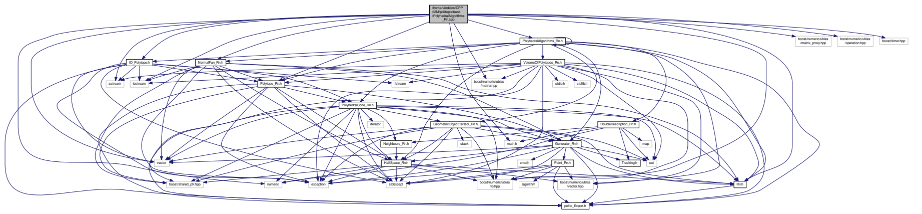
5.18.2.2 #define TEST_OK 0

Definition at line 38 of file politopixAPI.h.

5.19 /home/vindelos/CPP/I2M/politopix/trunk/PolyhedralAlgorithms_Rn.cpp File Reference

```
#include <iostream>
#include <sstream>
#include <vector>
#include <math.h>
#include <numeric>
#include <boost/numeric/ublas/matrix.hpp>
#include <boost/numeric/ublas/matrix_proxy.hpp>
#include <boost/numeric/ublas/operation.hpp>
#include <boost/numeric/ublas/io.hpp>
#include <boost/timer.hpp>
#include "Rn.h"
#include "Polytope_Rn.h"
#include "IO_Polytope.h"
#include "NormalFan_Rn.h"
#include "PolyhedralAlgorithms_Rn.h"
```

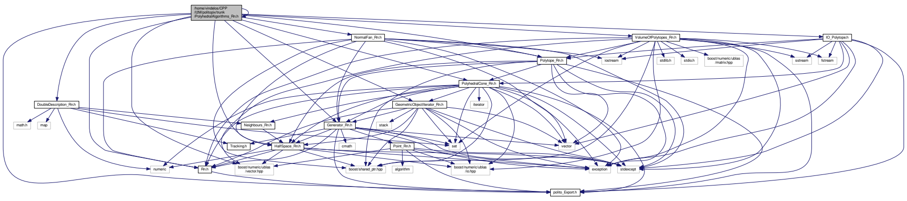
Include dependency graph for PolyhedralAlgorithms_Rn.cpp:



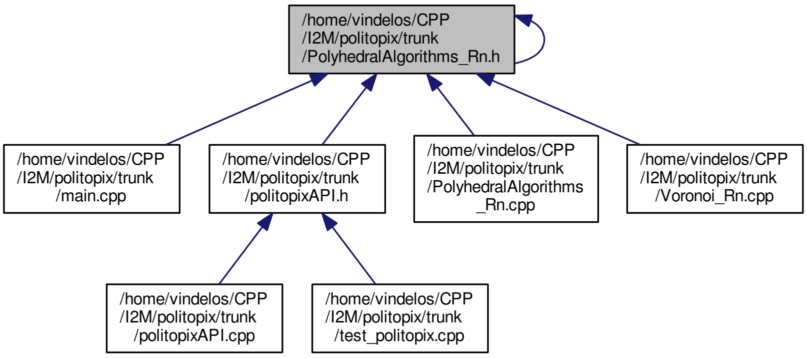
5.20 /home/vindelos/CPP/I2M/politopix/trunk/PolyhedralAlgorithms_Rn.h File Reference

```
#include <boost/numeric/ublas/vector.hpp>
#include "polito_Export.h"
#include "Rn.h"
#include "Polytope_Rn.h"
#include "IO_Polytope.h"
#include "Generator_Rn.h"
#include "NormalFan_Rn.h"
#include "HalfSpace_Rn.h"
#include "PolyhedralCone_Rn.h"
#include "VolumeOfPolytopes_Rn.h"
#include "DoubleDescription_Rn.h"
#include "PolyhedralAlgorithms_Rn.h"
#include "GeometricObjectIterator_Rn.h"
```

Include dependency graph for PolyhedralAlgorithms_Rn.h:



This graph shows which files directly or indirectly include this file:



Classes

- class FaceEnumeration

    Combinatorial face enumeration for polytopes.

- class MinkowskiSum

    Compute the Minkowski sum of two polytopes.

- class PseudoSumWithoutCaps

    Compute the Minkowski sum of two polytopes and then remove all cap half-spaces to truncate again.

- class PseudoIntersectionWithoutCaps

    Remove all cap half-spaces and then compute the intersection of two capped polytopes.

- class TopGeomTools

    Basic tools for topology and geometry: translations, polarity, ...

- class DoubleDescriptionFromGenerators

    Compute the V-description from the H-description.

- class Visualization

    2D visualization tools

Typedefs

- typedef std::vector< unsigned int > ListOfFaces

5.20.1 Typedef Documentation

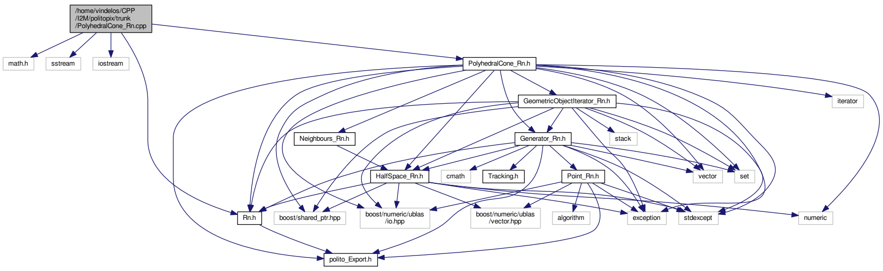
5.20.1.1 typedef std::vector< unsigned int > ListOfFaces

Definition at line 40 of file PolyhedralAlgorithms_Rn.h.

5.21 /home/vindelos/CPP/I2M/politopix/trunk/PolyhedralCone_Rn.cpp File Reference

```
#include <math.h>
#include <sstream>
#include <iostream>
#include "Rn.h"
#include "PolyhedralCone_Rn.h"
```

Include dependency graph for PolyhedralCone_Rn.cpp:



5.21.1 Detailed Description

Author

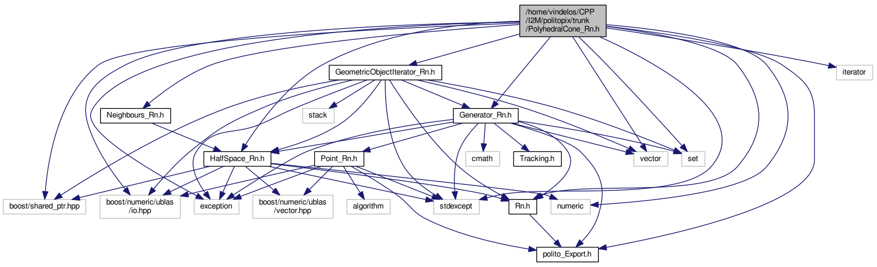
Delos Vincent (v.delos@i2m.u-bordeaux1.fr)

Definition in file PolyhedralCone_Rn.cpp.

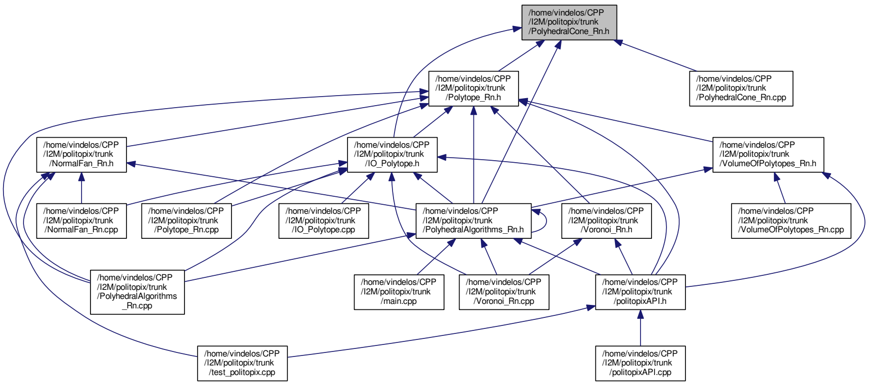
5.22 /home/vindelos/CPP/I2M/politopix/trunk/PolyhedralCone_Rn.h File Reference

```
#include <boost/numeric/ublas/io.hpp>
#include <boost/shared_ptr.hpp>
#include <stdexcept>
#include <exception>
#include <iterator>
#include <numeric>
#include <vector>
#include <set>
#include "polito_Export.h"
#include "GeometricObjectIterator_Rn.h"
#include "Neighbours_Rn.h"
#include "Generator_Rn.h"
#include "HalfSpace_Rn.h"
#include "Rn.h"
```

Include dependency graph for PolyhedralCone_Rn.h:



This graph shows which files directly or indirectly include this file:



Classes

- class PolyhedralCone_Rn

  Model a polyhedral cone using its two equivalent definitions : the convex hull and the half-space intersection. We store its edges in _listOfHS and the positive combination of these vectors generates the polyhedral cone.

5.22.1 Detailed Description

Author

Delos Vincent (`v.delos@i2m.u-bordeaux1.fr`)

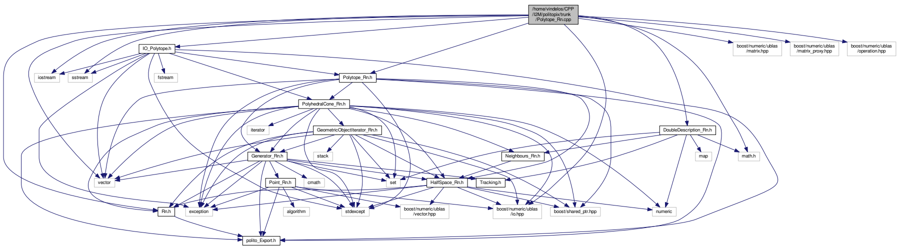
Definition in file PolyhedralCone_Rn.h.

5.23 /home/vindelos/CPP/I2M/politopix/trunk/Polytope_Rn.cpp File Reference

```
#include <iostream>
```

```
#include <sstream>
#include <vector>
#include <math.h>
#include <boost/numeric/ublas/matrix.hpp>
#include <boost/numeric/ublas/matrix_proxy.hpp>
#include <boost/numeric/ublas/operation.hpp>
#include <boost/numeric/ublas/io.hpp>
#include "Rn.h"
#include "Polytope_Rn.h"
#include "IO_Polytope.h"
#include "DoubleDescription_Rn.h"
```

Include dependency graph for Polytope_Rn.cpp:



Classes

- class RealNeighbours

Class dedicated to degeneration processing when looking for neighbors.

5.23.1 Detailed Description

Author

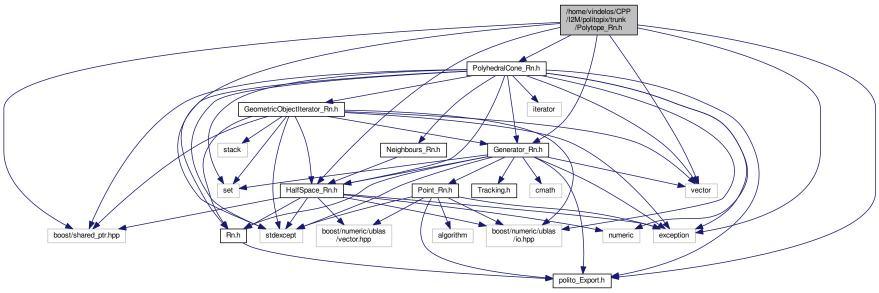
Delos Vincent (`v.delos@i2m.u-bordeaux1.fr`)

Definition in file Polytope_Rn.cpp.

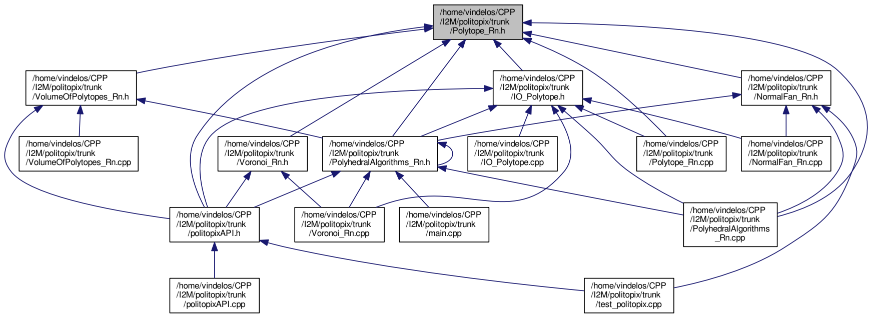
5.24 /home/vindelos/CPP/I2M/politopix/trunk/Polytope_Rn.h File Reference

```
#include <boost/shared_ptr.hpp>
#include <stdexcept>
#include <exception>
#include <vector>
#include "polito_Export.h"
#include "PolyhedralCone_Rn.h"
#include "Generator_Rn.h"
#include "HalfSpace_Rn.h"
```

Include dependency graph for Polytope_Rn.h:



This graph shows which files directly or indirectly include this file:



Classes

- class Polytope_Rn

  Model a polytope using its two equivalent definitions : the convex hull and the half-space intersection.

5.24.1 Detailed Description

Author

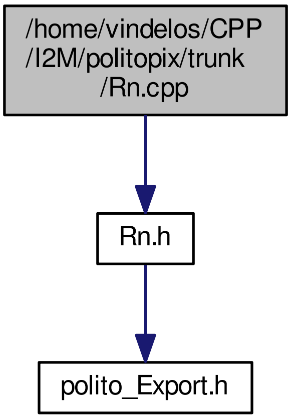
Delos Vincent (v.delos@i2m.u-bordeaux1.fr)

Definition in file Polytope_Rn.h.

5.25 /home/vindelos/CPP/I2M/politopix/trunk/Rn.cpp File Reference

```
#include "Rn.h"
```

Include dependency graph for Rn.cpp:



5.25.1 Detailed Description

Author

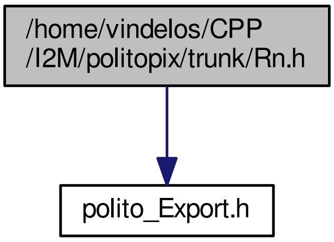
Delos Vincent (v.delos@i2m.u-bordeaux1.fr)

Definition in file Rn.cpp.

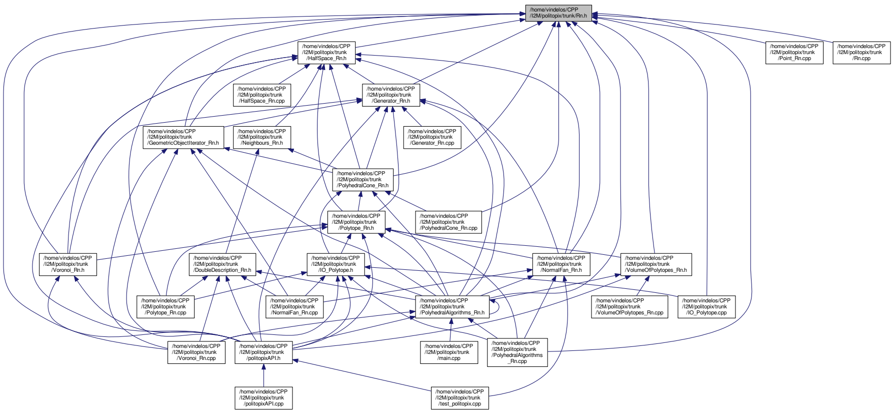
5.26 /home/vindelos/CPP/I2M/politopix/trunk/Rn.h File Reference

```
#include "polito_Export.h"
```

Include dependency graph for Rn.h:



This graph shows which files directly or indirectly include this file:


/home/vindelos/CPP /I2M/politopix/trunk/Rn.h
/home/vindelos/CPP /I2M/politopix/trunk /HalfSpace_Rn.h
/home/vindelos/CPP /I2M/politopix/trunk /Point_Rn.cpp
/home/vindelos/CPP /I2M/politopix/trunk /Rn.cpp
/home/vindelos/CPP /I2M/politopix/trunk /HalfSpace_Rn.cpp
/home/vindelos/CPP /I2M/politopix/trunk /Generator_Rn.h
/home/vindelos/CPP /I2M/politopix/trunk /GeometricObjectIterator_Rn.h
/home/vindelos/CPP /I2M/politopix/trunk /Neighbours_Rn.h
/home/vindelos/CPP /I2M/politopix/trunk /Generator_Rn.cpp
/home/vindelos/CPP /I2M/politopix/trunk /PolyhedralCone_Rn.h
/home/vindelos/CPP /I2M/politopix/trunk /Polytope_Rn.h
/home/vindelos/CPP /I2M/politopix/trunk /PolyhedralCone_Rn.cpp
/home/vindelos/CPP /I2M/politopix/trunk /Voronoi_Rn.h
/home/vindelos/CPP /I2M/politopix/trunk /DoubleDescription_Rn.h
/home/vindelos/CPP /I2M/politopix/trunk /IO_Polytope.h
/home/vindelos/CPP /I2M/politopix/trunk /NormalFan_Rn.h
/home/vindelos/CPP /I2M/politopix/trunk /VolumeOfPolytopes_Rn.h
/home/vindelos/CPP /I2M/politopix/trunk /Polytope_Rn.cpp
/home/vindelos/CPP /I2M/politopix/trunk /NormalFan_Rn.cpp
/home/vindelos/CPP /I2M/politopix/trunk /PolyhedralAlgorithms_Rn.h
/home/vindelos/CPP /I2M/politopix/trunk /VolumeOfPolytopes_Rn.cpp
/home/vindelos/CPP /I2M/politopix/trunk /IO_Polytope.cpp
/home/vindelos/CPP /I2M/politopix/trunk /Voronoi_Rn.cpp
/home/vindelos/CPP /I2M/politopix/trunk /politopixAPI.h
/home/vindelos/CPP /I2M/politopix/trunk /main.cpp
/home/vindelos/CPP /I2M/politopix/trunk /PolyhedralAlgorithms _Rn.cpp
/home/vindelos/CPP /I2M/politopix/trunk /politopixAPI.cpp
/home/vindelos/CPP /I2M/politopix/trunk /test_politopix.cpp

Classes

- class Rn

  This class stores static function that dispatch the main geometric values we use.

5.26.1 Detailed Description

Author

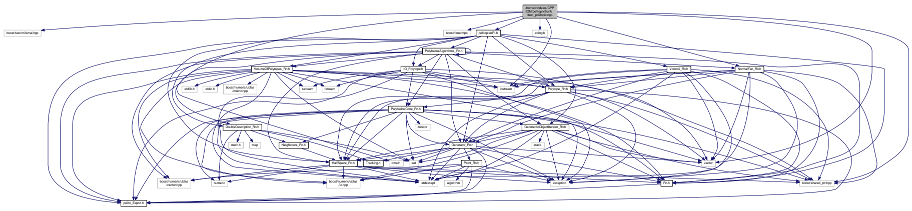
Delos Vincent (v.delos@i2m.u-bordeaux1.fr)

Definition in file Rn.h.

5.27 /home/vindelos/CPP/I2M/politopix/trunk/test_politopix.cpp File Reference

```
#include <boost/test/minimal.hpp>
#include <boost/shared_ptr.hpp>
#include <boost/timer.hpp>
#include <iostream>
#include <string.h>
#include <vector>
#include <cmath>
#include "politopixAPI.h"
#include "NormalFan_Rn.h"
```

Include dependency graph for test_politopix.cpp:



Functions

- int test_main (int argc, char ∗argv[])

5.27.1 Detailed Description

Author

Delos Vincent (v.delos@i2m.u-bordeaux1.fr)

Definition in file test_politopix.cpp.

5.27.2 Function Documentation

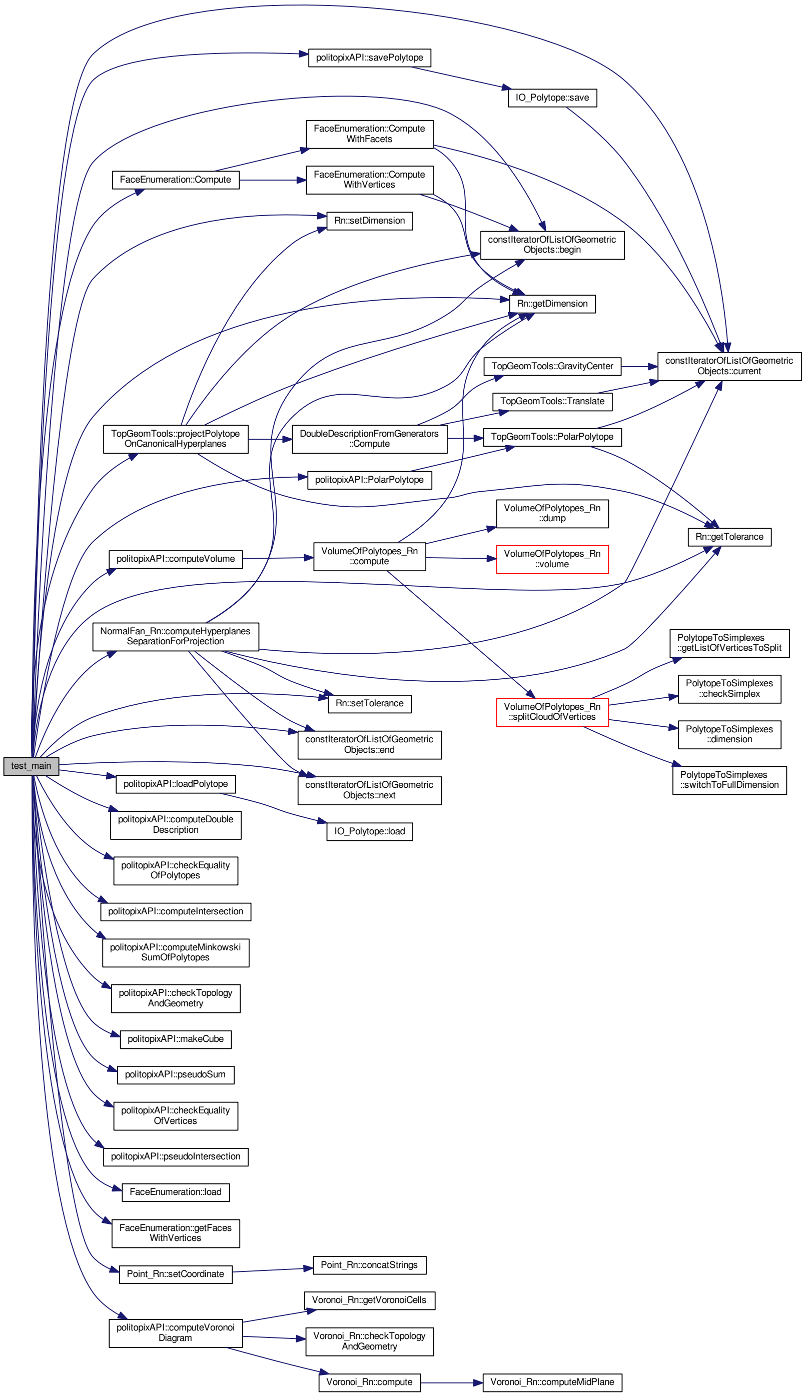
5.27.2.1 int test_main (int *argc,* char ∗ *argv[]*)

PSEUDO SUM METHOD ///

FULL METHOD ///

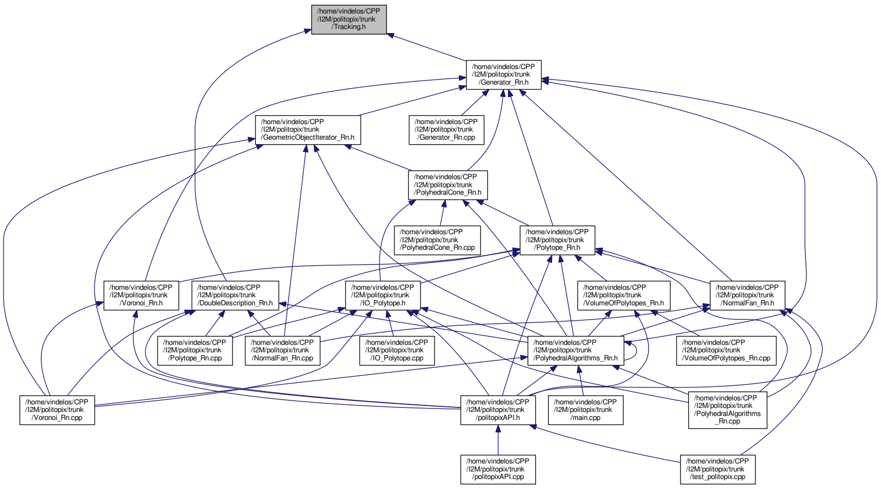
Definition at line 38 of file test_politopix.cpp.

Here is the call graph for this function:


test_main
politopixAPI::savePolytope
IO_Polytope::save
FaceEnumeration::Compute WithFacets
FaceEnumeration::Compute
FaceEnumeration::Compute WithVertices
Rn::setDimension
constIteratorOfListOfGeometric Objects::begin
Rn::getDimension
TopGeomTools::GravityCenter
constIteratorOfListOfGeometric Objects::current
TopGeomTools::Translate
TopGeomTools::projectPolytope OnCanonicalHyperplanes
DoubleDescriptionFromGenerators ::Compute
TopGeomTools::PolarPolytope
politopixAPI::PolarPolytope
VolumeOfPolytopes_Rn ::dump
Rn::getTolerance
politopixAPI::computeVolume
VolumeOfPolytopes_Rn ::compute
VolumeOfPolytopes_Rn ::volume
NormalFan_Rn::computeHyperplanes SeparationForProjection
PolytopeToSimplexes ::getListOfVerticesToSplit
PolytopeToSimplexes ::checkSimplex
Rn::setTolerance
VolumeOfPolytopes_Rn ::splitCloudOfVertices
PolytopeToSimplexes ::dimension
constIteratorOfListOfGeometric Objects::end
PolytopeToSimplexes ::switchToFullDimension
politopixAPI::loadPolytope
constIteratorOfListOfGeometric Objects::next
politopixAPI::computeDouble Description
IO_Polytope::load
politopixAPI::checkEquality OfPolytopes
politopixAPI::computeIntersection
politopixAPI::computeMinkowski SumOfPolytopes
politopixAPI::checkTopology AndGeometry
politopixAPI::makeCube
politopixAPI::pseudoSum
politopixAPI::checkEquality OfVertices
politopixAPI::pseudoIntersection
FaceEnumeration::load
FaceEnumeration::getFaces WithVertices
Point_Rn::concatStrings
Point_Rn::setCoordinate
Voronoi_Rn::getVoronoiCells
politopixAPI::computeVoronoi Diagram
Voronoi_Rn::checkTopology AndGeometry
Voronoi_Rn::compute
Voronoi_Rn::computeMidPlane

5.28 /home/vindelos/CPP/I2M/politopix/trunk/Tracking.h File Reference

This graph shows which files directly or indirectly include this file:


/home/vindelos/CPP /I2M/politopix/trunk /Tracking.h
/home/vindelos/CPP /I2M/politopix/trunk /Generator_Rn.h
/home/vindelos/CPP /I2M/politopix/trunk /GeometricObjectIterator_Rn.h
/home/vindelos/CPP /I2M/politopix/trunk /Generator_Rn.cpp
/home/vindelos/CPP /I2M/politopix/trunk /PolyhedralCone_Rn.h
/home/vindelos/CPP /I2M/politopix/trunk /PolyhedralCone_Rn.cpp
/home/vindelos/CPP /I2M/politopix/trunk /Polytope_Rn.h
/home/vindelos/CPP /I2M/politopix/trunk /Voronoi_Rn.h
/home/vindelos/CPP /I2M/politopix/trunk /DoubleDescription_Rn.h
/home/vindelos/CPP /I2M/politopix/trunk /IO_Polytope.h
/home/vindelos/CPP /I2M/politopix/trunk /VolumeOfPolytopes_Rn.h
/home/vindelos/CPP /I2M/politopix/trunk /NormalFan_Rn.h
/home/vindelos/CPP /I2M/politopix/trunk /Polytope_Rn.cpp
/home/vindelos/CPP /I2M/politopix/trunk /NormalFan_Rn.cpp
/home/vindelos/CPP /I2M/politopix/trunk /IO_Polytope.cpp
/home/vindelos/CPP /I2M/politopix/trunk /PolyhedralAlgorithms_Rn.h
/home/vindelos/CPP /I2M/politopix/trunk /VolumeOfPolytopes_Rn.cpp
/home/vindelos/CPP /I2M/politopix/trunk /Voronoi_Rn.cpp
/home/vindelos/CPP /I2M/politopix/trunk /politopixAPI.h
/home/vindelos/CPP /I2M/politopix/trunk /main.cpp
/home/vindelos/CPP /I2M/politopix/trunk /PolyhedralAlgorithms _Rn.cpp
/home/vindelos/CPP /I2M/politopix/trunk /politopixAPI.cpp
/home/vindelos/CPP /I2M/politopix/trunk /test_politopix.cpp

Classes

- class TrackingOperatorToResult

  This class stores static function that dispatch the main geometric values we use.
- class TrackingBinaryOperation

  This class stores static function that dispatch the main geometric values we use.

Typedefs

- typedef std::pair< int, int > operator1_operator2

Enumerations

- enum StatusBefore { Operator_UNCHANGED = 0, Operator_MODIFIED = 1, Operator_DELETED = 2, Operator_UNKNOWN = 4 }
- enum StatusAfter { Result_UNCHANGED = 0, Result_MODIFIED = 1, Result_CREATED = 2, Result_UNKNOWN = 4 }

5.28.1 Detailed Description

Author

Delos Vincent (`v.delos@i2m.u-bordeaux1.fr`)

Definition in file Tracking.h.

5.28.2 Typedef Documentation

5.28.2.1 typedef std::pair< int, int > operator1_operator2

Definition at line 107 of file Tracking.h.

5.28.3 Enumeration Type Documentation

5.28.3.1 enum StatusAfter

Enumerator

Result_UNCHANGED

Result_MODIFIED

Result_CREATED

Result_UNKNOWN

Definition at line 32 of file Tracking.h.

5.28.3.2 enum StatusBefore

Enumerator

Operator_UNCHANGED

Operator_MODIFIED

Operator_DELETED

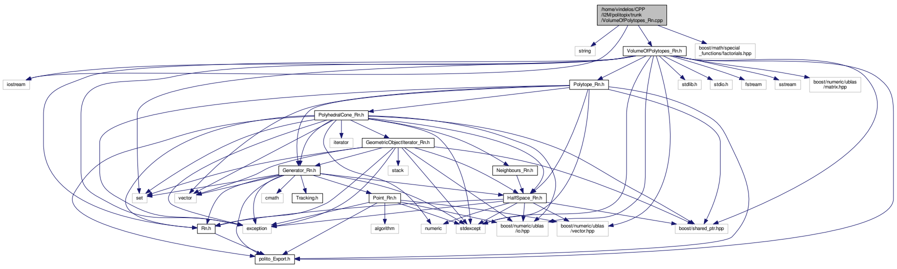
Operator_UNKNOWN

Definition at line 25 of file Tracking.h.

5.29 /home/vindelos/CPP/I2M/politopix/trunk/VolumeOfPolytopes_Rn.cpp File Reference

```
#include <string>
#include <iostream>
#include <boost/math/special_functions/factorials.hpp>
#include "VolumeOfPolytopes_Rn.h"
```

Include dependency graph for VolumeOfPolytopes_Rn.cpp:



5.29.1 Detailed Description

Author

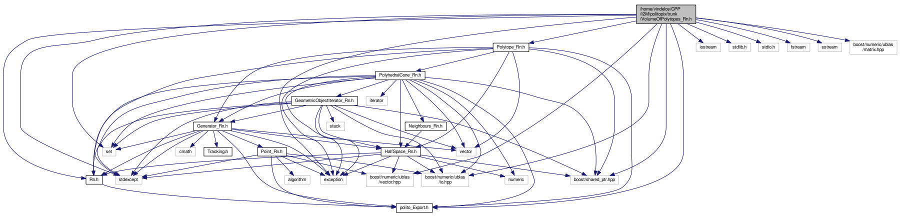
Delos Vincent (v.delos@i2m.u-bordeaux1.fr)

Definition in file VolumeOfPolytopes_Rn.cpp.

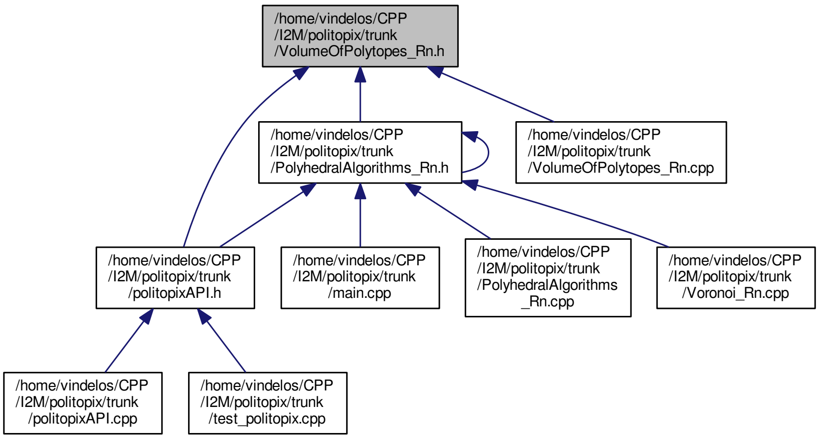
5.30 /home/vindelos/CPP/I2M/politopix/trunk/VolumeOfPolytopes_Rn.h File Reference

```
#include <stdexcept>
#include <exception>
#include <iostream>
#include <stdlib.h>
#include <stdio.h>
#include <fstream>
#include <sstream>
#include <set>
#include <boost/shared_ptr.hpp>
#include <boost/numeric/ublas/io.hpp>
#include <boost/numeric/ublas/vector.hpp>
#include <boost/numeric/ublas/matrix.hpp>
#include "polito_Export.h"
#include "Polytope_Rn.h"
#include "Rn.h"
```

Include dependency graph for VolumeOfPolytopes_Rn.h:



This graph shows which files directly or indirectly include this file:



Classes

- class PolytopeToSimplexes
- class VolumeOfPolytopes_Rn

    Split a polytope into simplices to compute its volume.
    *Two Algorithms for Determining Volumes of Convex Polyhedra (1979) by **Jacques Cohen** and **Timothy Hickey** Journal of the ACM (JACM) JACM Homepage archive*

Volume 26 Issue 3, july 1979
Pages 401-414

5.30.1 Detailed Description

Author

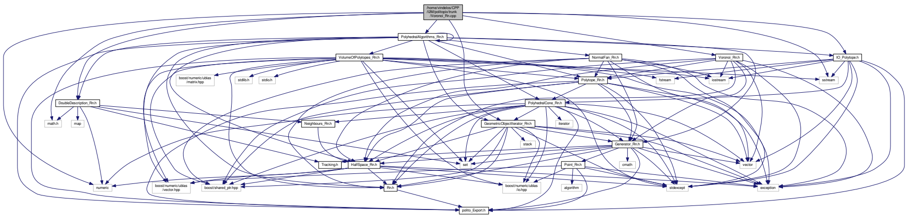
Delos Vincent (v.delos@i2m.u-bordeaux1.fr)

Definition in file VolumeOfPolytopes_Rn.h.

5.31 /home/vindelos/CPP/I2M/politopix/trunk/Voronoi_Rn.cpp File Reference

```
#include "GeometricObjectIterator_Rn.h"
#include "PolyhedralAlgorithms_Rn.h"
#include "DoubleDescription_Rn.h"
#include "Voronoi_Rn.h"
#include "IO_Polytope.h"
#include <math.h>
#include <numeric>
#include <sstream>
```

Include dependency graph for Voronoi_Rn.cpp:



5.31.1 Detailed Description

Author

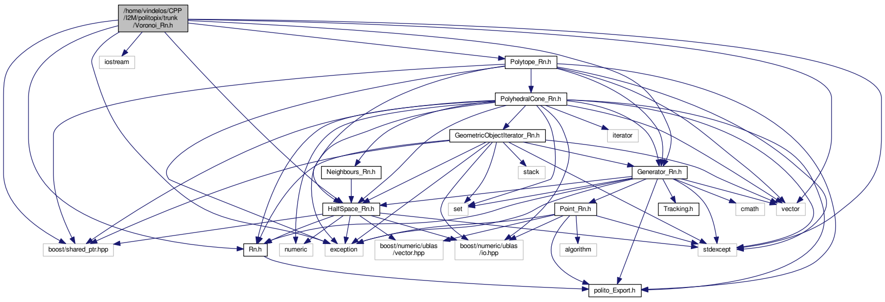
Delos Vincent (v.delos@i2m.u-bordeaux1.fr)

Definition in file Voronoi_Rn.cpp.

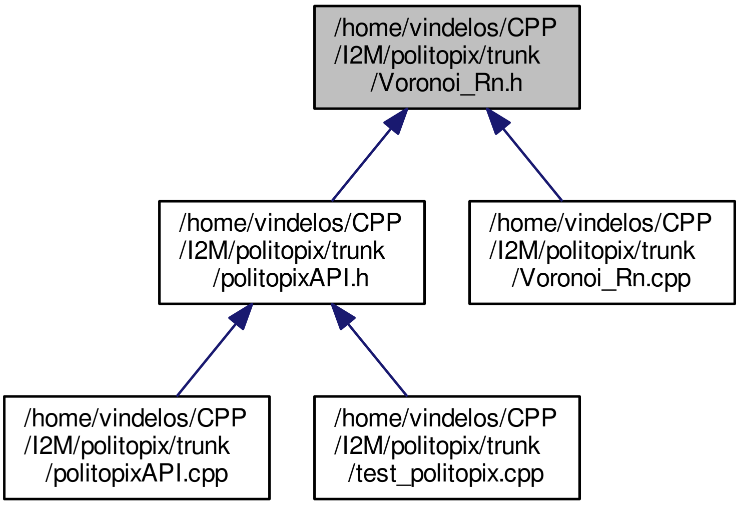
5.32 /home/vindelos/CPP/I2M/politopix/trunk/Voronoi_Rn.h File Reference

```
#include <boost/shared_ptr.hpp>
#include <stdexcept>
#include <exception>
#include <iostream>
#include <vector>
#include "Rn.h"
#include "Generator_Rn.h"
#include "HalfSpace_Rn.h"
#include "Polytope_Rn.h"
```

Include dependency graph for Voronoi_Rn.h:



This graph shows which files directly or indirectly include this file:



Classes

- class Voronoi_Rn

    Compute a n-dimensional Voronoi diagram. It is a partitioning of a space into regions based on distance to points. Both the space and the list of points are provided as input.

5.32.1 Detailed Description

Author

Delos Vincent (`v.delos@i2m.u-bordeaux1.fr`)

Definition in file Voronoi_Rn.h.