# The COCONUT API

Version 2.32

## Reference Manual

Hermann Schichl
University of Vienna, Department of Mathematics
A-1090 Wien, Austria
email: Hermann.Schichl@esi.ac.at

Technical Report
October 2003

# Contents



# 1 COCONUT API Namespace Index

## 1.1 COCONUT API Namespace List

Here is a list of all namespaces with brief descriptions:



# 2 COCONUT API Hierarchical Index

## 2.1 COCONUT API Class Hierarchy

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

# 3 COCONUT API Compound Index

## 3.1 COCONUT API Compound List

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

# 4 COCONUT API File Index

## 4.1 COCONUT API File List

Here is a list of all files with brief descriptions:

# 5 COCONUT API Namespace Documentation

## 5.1 std Namespace Reference

## 5.2 vgtl Namespace Reference

# 6 COCONUT API Class Documentation

## 6.1 __docompare_nodes Struct Reference

`#include <model-inline.h>`

### Public Methods

- bool operator() (const model::walker &_a, unsigned int _b) const
- bool operator() (unsigned int _a, const model::walker &_b) const
- bool operator() (const model::walker &_a, const model::walker &_b) const

### 6.1.1 Member Function Documentation

#### 6.1.1.1 bool __docompare_nodes::operator() (const model::walker & *_a*, const model::walker & *_b*) const [inline]

Definition at line 1232 of file model-inline.h.

#### 6.1.1.2 bool __docompare_nodes::operator() (unsigned int _a, const model::walker & _b) const [inline]

Definition at line 1230 of file model-inline.h.

#### 6.1.1.3 bool __docompare_nodes::operator() (const model::walker & _a, unsigned int _b) const [inline]

Definition at line 1228 of file model-inline.h.

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

- model-inline.h

## 6.2 __docompare_variables Struct Reference

`#include <model-inline.h>`

### Public Methods

- bool operator() (const model::walker &_a, unsigned int _b) const
- bool operator() (unsigned int _a, const model::walker &_b) const
- bool operator() (const model::walker &_a, const model::walker &_b) const

### 6.2.1 Member Function Documentation

#### 6.2.1.1 bool __docompare_variables::operator() (const model::walker & _a, const model::walker & _b) const [inline]

Definition at line 1242 of file model-inline.h.

#### 6.2.1.2 bool __docompare_variables::operator() (unsigned int _a, const model::walker & _b) const [inline]

Definition at line 1240 of file model-inline.h.

#### 6.2.1.3 bool __docompare_variables::operator() (const model::walker & _a, unsigned int _b) const [inline]

Definition at line 1238 of file model-inline.h.

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

- model-inline.h

## 6.3 __linalg_cvec< _S > Class Template Reference

`#include <linalg.h>`

### Public Types

- typedef std::vector< _S >::value_type value_type
- typedef std::vector< _S >::reference reference
- typedef std::vector< _S >::pointer pointer
- typedef std::vector< _S >::const_reference const_reference
- typedef std::vector< _S >::const_pointer const_pointer
- typedef std::vector< _S >::size_type size_type
- typedef std::vector< _S >::difference_type difference_type
- typedef std::vector< _S >::allocator_type allocator_type
- typedef std::vector< _S >::iterator iterator
- typedef std::vector< _S >::const_iterator const_iterator
- typedef std::vector< _S >::reverse_iterator reverse_iterator
- typedef std::vector< _S >::const_reverse_iterator const_reverse_iterator
- typedef mtl::dense_tag sparsity
- typedef mtl::oned_tag dimension
- typedef mtl::scaled1D< _Self > scaled_type
- typedef mtl::light1D< _S > subrange_type
- typedef _Self IndexArray
- typedef _Self IndexArrayRef
- enum { N = 0 }

### Public Methods

- __linalg_cvec (const std::vector< _S > &s)
- ~__linalg_cvec ()
- iterator begin ()
- iterator end ()
- const_iterator begin () const
- const_iterator end () const

**template<class _S> class __linalg_cvec< _S >**

#### 6.3.1 Member Typedef Documentation

##### 6.3.1.1 template<class _S> typedef std::vector<_S>::allocator_type __linalg_cvec< _S >::allocator_type

Definition at line 125 of file linalg.h.

##### 6.3.1.2 template<class _S> typedef std::vector<_S>::const_iterator __linalg_cvec< _S >::const_iterator

Definition at line 127 of file linalg.h.

##### 6.3.1.3 template<class _S> typedef std::vector<_S>::const_pointer __linalg_cvec< _S >::const_pointer

Definition at line 122 of file linalg.h.

#### 6.3.1.4 template<class _S> typedef std::vector<_S>::const_reference __linalg_cvec< _S >::const_reference

Definition at line 121 of file linalg.h.

#### 6.3.1.5 template<class _S> typedef std::vector<_S>::const_reverse_iterator __linalg_cvec< _S >::const_reverse_iterator

Definition at line 129 of file linalg.h.

#### 6.3.1.6 template<class _S> typedef std::vector<_S>::difference_type __linalg_cvec< _S >::difference_type

Definition at line 124 of file linalg.h.

#### 6.3.1.7 template<class _S> typedef mtl::oned_tag __linalg_cvec< _S >::dimension

Definition at line 132 of file linalg.h.

#### 6.3.1.8 template<class _S> typedef _Self __linalg_cvec< _S >::IndexArray

Definition at line 137 of file linalg.h.

#### 6.3.1.9 template<class _S> typedef _Self __linalg_cvec< _S >::IndexArrayRef

Definition at line 138 of file linalg.h.

#### 6.3.1.10 template<class _S> typedef std::vector<_S>::iterator __linalg_cvec< _S >::iterator

Definition at line 126 of file linalg.h.

#### 6.3.1.11 template<class _S> typedef std::vector<_S>::pointer __linalg_cvec< _S >::pointer

Definition at line 120 of file linalg.h.

#### 6.3.1.12 template<class _S> typedef std::vector<_S>::reference __linalg_cvec< _S >::reference

Definition at line 119 of file linalg.h.

#### 6.3.1.13 template<class _S> typedef std::vector<_S>::reverse_iterator __linalg_cvec< _S >::reverse_iterator

Definition at line 128 of file linalg.h.

#### 6.3.1.14 template<class _S> typedef mtl::scaled1D<_Self> __linalg_cvec< _S >::scaled_type

Definition at line 134 of file linalg.h.

#### 6.3.1.15 template<class _S> typedef std::vector<_S>::size_type __linalg_cvec< _S >::size_type

Definition at line 123 of file linalg.h.

#### 6.3.1.16 template<class _S> typedef mtl::dense_tag __linalg_cvec< _S >::sparsity

Definition at line 131 of file linalg.h.

#### 6.3.1.17 template<class _S> typedef mtl::light1D<_S> __linalg_cvec< _S >::subrange_type

Definition at line 136 of file linalg.h.

#### 6.3.1.18 template<class _S> typedef std::vector<_S>::value_type __linalg_cvec< _S >::value_type

Definition at line 118 of file linalg.h.

### 6.3.2 Member Enumeration Documentation

#### 6.3.2.1 template<class _S> anonymous enum

**Enumeration values:**

**N**

Definition at line 116 of file linalg.h.

### 6.3.3 Constructor & Destructor Documentation

#### 6.3.3.1 template<class _S> __linalg_cvec< _S >::__linalg_cvec (const std::vector< _S > & *s*) [inline]

Definition at line 140 of file linalg.h.

#### 6.3.3.2 template<class _S> __linalg_cvec< _S >::~__linalg_cvec () [inline]

Definition at line 141 of file linalg.h.

### 6.3.4 Member Function Documentation

#### 6.3.4.1 template<class _S> const_iterator __linalg_cvec< _S >::begin () const [inline]

Definition at line 146 of file linalg.h.

#### 6.3.4.2 template<class _S> iterator __linalg_cvec< _S >::begin () [inline]

Definition at line 143 of file linalg.h.

#### 6.3.4.3 template<class _S> const_iterator __linalg_cvec< _S >::end () const [inline]

Definition at line 147 of file linalg.h.

#### 6.3.4.4 template<class _S> iterator __linalg_cvec< _S >::end () [inline]

Definition at line 144 of file linalg.h.

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

- linalg.h

## 6.4 __linalg_vec< _S > Class Template Reference

`#include <linalg.h>`

### Public Types

- typedef std::vector< _S >::value_type value_type
- typedef std::vector< _S >::reference reference
- typedef std::vector< _S >::pointer pointer
- typedef std::vector< _S >::const_reference const_reference
- typedef std::vector< _S >::const_pointer const_pointer
- typedef std::vector< _S >::size_type size_type
- typedef std::vector< _S >::difference_type difference_type
- typedef std::vector< _S >::allocator_type allocator_type
- typedef std::vector< _S >::iterator iterator
- typedef std::vector< _S >::const_iterator const_iterator
- typedef std::vector< _S >::reverse_iterator reverse_iterator
- typedef std::vector< _S >::const_reverse_iterator const_reverse_iterator
- typedef mtl::dense_tag sparsity
- typedef mtl::oned_tag dimension
- typedef mtl::scaled1D< _Self > scaled_type
- typedef mtl::light1D< _S > subrange_type
- typedef _Self IndexArray
- typedef _Self IndexArrayRef
- enum { N = 0 }

### Public Methods

- __linalg_vec (std::vector< _S > &s)
- ~__linalg_vec ()
- iterator begin ()
- iterator end ()
- const_iterator begin () const
- const_iterator end () const

**template<class _S> class __linalg_vec< _S >**

### 6.4.1 Member Typedef Documentation

#### 6.4.1.1 template<class _S> typedef std::vector<_S>::allocator_type __linalg_vec< _S >::allocator_type

Definition at line 168 of file linalg.h.

#### 6.4.1.2 template<class _S> typedef std::vector<_S>::const_iterator __linalg_vec< _S >::const_iterator

Definition at line 170 of file linalg.h.

#### 6.4.1.3 template<class _S> typedef std::vector<_S>::const_pointer __linalg_vec< _S >::const_pointer

Definition at line 165 of file linalg.h.

#### 6.4.1.4 template<class _S> typedef std::vector<_S>::const_reference __linalg_vec< _S >::const_reference

Definition at line 164 of file linalg.h.

#### 6.4.1.5 template<class _S> typedef std::vector<_S>::const_reverse_iterator __linalg_vec< _S >::const_reverse_iterator

Definition at line 172 of file linalg.h.

#### 6.4.1.6 template<class _S> typedef std::vector<_S>::difference_type __linalg_vec< _S >::difference_type

Definition at line 167 of file linalg.h.

#### 6.4.1.7 template<class _S> typedef mtl::oned_tag __linalg_vec< _S >::dimension

Definition at line 175 of file linalg.h.

#### 6.4.1.8 template<class _S> typedef _Self __linalg_vec< _S >::IndexArray

Definition at line 180 of file linalg.h.

#### 6.4.1.9 template<class _S> typedef _Self __linalg_vec< _S >::IndexArrayRef

Definition at line 181 of file linalg.h.

#### 6.4.1.10 template<class _S> typedef std::vector<_S>::iterator __linalg_vec< _S >::iterator

Definition at line 169 of file linalg.h.

#### 6.4.1.11 template<class _S> typedef std::vector<_S>::pointer __linalg_vec< _S >::pointer

Definition at line 163 of file linalg.h.

#### 6.4.1.12 template<class _S> typedef std::vector<_S>::reference __linalg_vec< _S >::reference

Definition at line 162 of file linalg.h.

#### 6.4.1.13 template<class _S> typedef std::vector<_S>::reverse_iterator __linalg_vec< _S >::reverse_iterator

Definition at line 171 of file linalg.h.

#### 6.4.1.14 template<class _S> typedef mtl::scaled1D<_Self> __linalg_vec< _S >::scaled_type

Definition at line 177 of file linalg.h.

#### 6.4.1.15 template<class _S> typedef std::vector<_S>::size_type __linalg_vec< _S >::size_type

Definition at line 166 of file linalg.h.

#### 6.4.1.16 template<class _S> typedef mtl::dense_tag __linalg_vec< _S >::sparsity

Definition at line 174 of file linalg.h.

#### 6.4.1.17 template<class _S> typedef mtl::light1D<_S> __linalg_vec< _S >::subrange_type

Definition at line 179 of file linalg.h.

#### 6.4.1.18 template<class _S> typedef std::vector<_S>::value_type __linalg_vec< _S >::value_type

Definition at line 161 of file linalg.h.

### 6.4.2 Member Enumeration Documentation

#### 6.4.2.1 template<class _S> anonymous enum

**Enumeration values:**

**N**

Definition at line 159 of file linalg.h.

### 6.4.3 Constructor & Destructor Documentation

#### 6.4.3.1 template<class _S> __linalg_vec< _S >::__linalg_vec (std::vector< _S > & *s*) [inline]

Definition at line 184 of file linalg.h.

#### 6.4.3.2 template<class _S> __linalg_vec< _S >::~__linalg_vec () [inline]

Definition at line 185 of file linalg.h.

### 6.4.4 Member Function Documentation

#### 6.4.4.1 template<class _S> const_iterator __linalg_vec< _S >::begin () const [inline]

Definition at line 190 of file linalg.h.

#### 6.4.4.2 template<class _S> iterator __linalg_vec< _S >::begin () [inline]

Definition at line 187 of file linalg.h.

#### 6.4.4.3 template<class _S> const_iterator __linalg_vec< _S >::end () const [inline]

Definition at line 191 of file linalg.h.

#### 6.4.4.4 template<class _S> iterator _linalg_vec< _S >::end () [inline]

Definition at line 188 of file linalg.h.

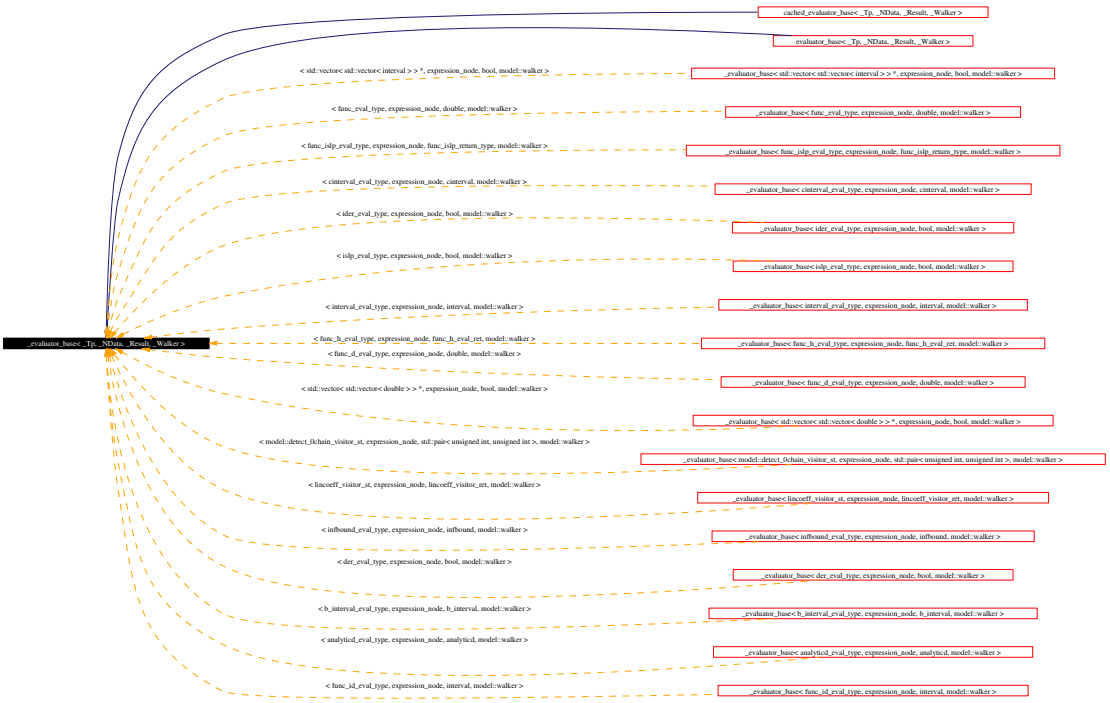
The documentation for this class was generated from the following file:

- linalg.h

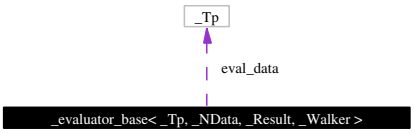
## 6.5 _evaluator_base< _Tp, _NData, _Result, _Walker > Class Template Reference

`#include <evaluator.h>`

Inheritance diagram for _evaluator_base< _Tp, _NData, _Result, _Walker >:




Collaboration diagram for _evaluator_base< _Tp, _NData, _Result, _Walker >:




### Public Types

- typedef _Tp data_type
- typedef _NData node_data_type
- typedef _Result return_value
- typedef _Walker walker

**Public Methods**

- _evaluator_base ()
- _evaluator_base (const _Tp &__x)
- _evaluator_base (const _Self &__x)
- virtual ~_evaluator_base ()
- virtual return_value vvalue ()
- virtual return_value value ()
- virtual int vcollect (const return_value &__cresult)
- virtual int collect (const node_data_type &__data, const return_value &__cresult)
- virtual void postorder (const node_data_type &__data)

**Protected Attributes**

- _Tp eval_data

**template<class _Tp, class _NData, class _Result, class _Walker> class _evaluator_base< _Tp, _NData, _Result, _Walker >**

### 6.5.1 Member Typedef Documentation

#### 6.5.1.1 template<class _Tp, class _NData, class _Result, class _Walker> typedef _Tp _evaluator_base< _Tp, _NData, _Result, _Walker >::data_type

Definition at line 245 of file evaluator.h.

#### 6.5.1.2 template<class _Tp, class _NData, class _Result, class _Walker> typedef _NData _evaluator_base< _Tp, _NData, _Result, _Walker >::node_data_type

Reimplemented in evaluator_base< _Tp, _NData, _Result, _Walker >, cached_evaluator_base< _Tp, _NData, _Result, _Walker >, forward_evaluator_base< _Tp, _NData, _Result, _Walker >, backward_evaluator_base< _Tp, _NData, _Result, _Walker >, cached_forward_evaluator_base< _Tp, _NData, _Result, _Walker >, cached_backward_evaluator_base< _Tp, _NData, _Result, _Walker >, cached_evaluator_base< std::vector< std::vector< interval > > *, expression_node, bool, model::walker >, cached_evaluator_base< func_eval_type, expression_node, double, model::walker >, cached_evaluator_base< func_islp_eval_type, expression_node, func_islp_return_type, model::walker >, cached_evaluator_base< cinterval_eval_type, expression_node, cinterval, model::walker >, cached_evaluator_base< ider_eval_type, expression_node, bool, model::walker >, cached_evaluator_base< islp_eval_type, expression_node, bool, model::walker >, cached_evaluator_base< interval_eval_type, expression_node, interval, model::walker >, cached_evaluator_base< func_h_eval_type, expression_node, func_h_eval_ret, model::walker >, cached_evaluator_base< func_d_eval_type, expression_node, double, model::walker >, cached_evaluator_base< std::vector< std::vector< double > > *, expression_node, bool, model::walker >, cached_evaluator_base< model::detect_0chain_visitor_st, expression_node, std::pair< unsigned int, unsigned int >, model::walker >, cached_evaluator_base< lincoeff_visitor_st, expression_node, lincoeff_visitor_ret, model::walker >, cached_evaluator_base< infbound_eval_type, expression_node, infbound, model::walker >, cached_evaluator_base< der_eval_type, expression_node, bool, model::walker >, cached_evaluator_base< b_interval_eval_type, expression_node, b_interval, model::walker >, cached_evaluator_base< analyticd_eval_type, expression_node, analyticd, model::walker >, cached_evaluator_base< func_id_eval_type, expression_node, interval, model::walker >, cached_forward_evaluator_base< std::vector< std::vector< interval > > *, expression_node, bool, model::walker >, cached_forward_evaluator_base< func_eval_type, expression_node, double, model::walker >, cached_forward_evaluator_base< func_islp_eval_type, expression_node, func_islp_return_type, model::walker >, cached_forward_evaluator_base< cinterval_eval_type,

expression_node, cinterval, model::walker >, cached_forward_evaluator_base< interval_eval_type, expression_node, interval, model::walker >, cached_forward_evaluator_base< func_h_eval_type, expression_node, func_h_eval_ret, model::walker >, cached_forward_evaluator_base< func_d_eval_type, expression_node, double, model::walker >, cached_forward_evaluator_base< std::vector< std::vector< double > > *, expression_node, bool, model::walker >, cached_forward_evaluator_base< model::detect_0chain_visitor_st, expression_node, std::pair< unsigned int, unsigned int >, model::walker >, cached_forward_evaluator_base< infbound_eval_type, expression_node, infbound, model::walker >, cached_forward_evaluator_base< b_interval_eval_type, expression_node, b_interval, model::walker >, cached_forward_evaluator_base< analyticd_eval_type, expression_node, analyticd, model::walker >, cached_forward_evaluator_base< func_id_eval_type, expression_node, interval, model::walker >, cached_backward_evaluator_base< ider_eval_type, expression_node, bool, model::walker >, cached_backward_evaluator_base< islp_eval_type, expression_node, bool, model::walker >, cached_backward_evaluator_base< lincoeff_visitor_st, expression_node, lincoeff_visitor_ret, model::walker >, and cached_backward_evaluator_base< der_eval_type, expression_node, bool, model::walker >.

Definition at line 246 of file evaluator.h.

#### 6.5.1.3 template<class _Tp, class _NData, class _Result, class _Walker> typedef _Result _evaluator_base< _Tp, _NData, _Result, _Walker >::return_value

Reimplemented in evaluator_base< _Tp, _NData, _Result, _Walker >, cached_evaluator_base< _Tp, _NData, _Result, _Walker >, forward_evaluator_base< _Tp, _NData, _Result, _Walker >, backward_evaluator_base< _Tp, _NData, _Result, _Walker >, cached_forward_evaluator_base< _Tp, _NData, _Result, _Walker >, cached_backward_evaluator_base< _Tp, _NData, _Result, _Walker >, cached_evaluator_base< std::vector< std::vector< interval > > *, expression_node, bool, model::walker >, cached_evaluator_base< func_eval_type, expression_node, double, model::walker >, cached_evaluator_base< func_islp_eval_type, expression_node, func_islp_return_type, model::walker >, cached_evaluator_base< cinterval_eval_type, expression_node, cinterval, model::walker >, cached_evaluator_base< ider_eval_type, expression_node, bool, model::walker >, cached_evaluator_base< islp_eval_type, expression_node, bool, model::walker >, cached_evaluator_base< interval_eval_type, expression_node, interval, model::walker >, cached_evaluator_base< func_h_eval_type, expression_node, func_h_eval_ret, model::walker >, cached_evaluator_base< func_d_eval_type, expression_node, double, model::walker >, cached_evaluator_base< std::vector< std::vector< double > > *, expression_node, bool, model::walker >, cached_evaluator_base< model::detect_0chain_visitor_st, expression_node, std::pair< unsigned int, unsigned int >, model::walker >, cached_evaluator_base< lincoeff_visitor_st, expression_node, lincoeff_visitor_ret, model::walker >, cached_evaluator_base< infbound_eval_type, expression_node, infbound, model::walker >, cached_evaluator_base< der_eval_type, expression_node, bool, model::walker >, cached_evaluator_base< b_interval_eval_type, expression_node, b_interval, model::walker >, cached_evaluator_base< analyticd_eval_type, expression_node, analyticd, model::walker >, cached_evaluator_base< func_id_eval_type, expression_node, interval, model::walker >, cached_forward_evaluator_base< std::vector< std::vector< interval > > *, expression_node, bool, model::walker >, cached_forward_evaluator_base< func_eval_type, expression_node, double, model::walker >, cached_forward_evaluator_base< func_islp_eval_type, expression_node, func_islp_return_type, model::walker >, cached_forward_evaluator_base< cinterval_eval_type, expression_node, cinterval, model::walker >, cached_forward_evaluator_base< interval_eval_type, expression_node, interval, model::walker >, cached_forward_evaluator_base< func_h_eval_type, expression_node, func_h_eval_ret, model::walker >, cached_forward_evaluator_base< func_d_eval_type, expression_node, double, model::walker >, cached_forward_evaluator_base< std::vector< std::vector< double > > *, expression_node, bool, model::walker >, cached_forward_evaluator_base< model::detect_0chain_visitor_st, expression_node, std::pair< unsigned int, unsigned int >, model::walker >, cached_forward_evaluator_base< infbound_eval_type, expression_node, infbound, model::walker >, cached_forward_evaluator_base< b_interval_eval_type, expression_node, b_interval, model::walker >, cached_forward_evaluator_base< analyticd_eval_type, expression_node, analyticd, model::walker >, cached_forward_evaluator_base< func_id_eval_type, expression_node, interval, model::walker >, cached_backward_evaluator_base< ider_eval_type, expression_node, bool, model::walker >, cached_backward_evaluator_base< islp_eval_type, expression_node,

node, bool, model::walker >, cached_backward_evaluator_base< lincoeff_visitor_st, expression_node, lincoeff_visitor_ret, model::walker >, and cached_backward_evaluator_base< der_eval_type, expression_node, bool, model::walker >.

Definition at line 248 of file evaluator.h.

#### 6.5.1.4 template<class _Tp, class _NData, class _Result, class _Walker> typedef _Walker _evaluator_base< _Tp, _NData, _Result, _Walker >::walker

Reimplemented in evaluator_base< _Tp, _NData, _Result, _Walker >, cached_evaluator_base< _Tp, _NData, _Result, _Walker >, forward_evaluator_base< _Tp, _NData, _Result, _Walker >, backward_evaluator_base< _Tp, _NData, _Result, _Walker >, cached_forward_evaluator_base< _Tp, _NData, _Result, _Walker >, cached_backward_evaluator_base< _Tp, _NData, _Result, _Walker >, cached_evaluator_base< std::vector< std::vector< interval > > *, expression_node, bool, model::walker >, cached_evaluator_base< func_eval_type, expression_node, double, model::walker >, cached_evaluator_base< func_islp_eval_type, expression_node, func_islp_return_type, model::walker >, cached_evaluator_base< cinterval_eval_type, expression_node, cinterval, model::walker >, cached_evaluator_base< ider_eval_type, expression_node, bool, model::walker >, cached_evaluator_base< islp_eval_type, expression_node, bool, model::walker >, cached_evaluator_base< interval_eval_type, expression_node, interval, model::walker >, cached_evaluator_base< func_h_eval_type, expression_node, func_h_eval_ret, model::walker >, cached_evaluator_base< func_d_eval_type, expression_node, double, model::walker >, cached_evaluator_base< std::vector< std::vector< double > > *, expression_node, bool, model::walker >, cached_evaluator_base< model::detect_0chain_visitor_st, expression_node, std::pair< unsigned int, unsigned int >, model::walker >, cached_evaluator_base< lincoeff_visitor_st, expression_node, lincoeff_visitor_ret, model::walker >, cached_evaluator_base< infbound_eval_type, expression_node, infbound, model::walker >, cached_evaluator_base< der_eval_type, expression_node, bool, model::walker >, cached_evaluator_base< b_interval_eval_type, expression_node, b_interval, model::walker >, cached_evaluator_base< analyticd_eval_type, expression_node, analyticd, model::walker >, cached_evaluator_base< func_id_eval_type, expression_node, interval, model::walker >, cached_forward_evaluator_base< std::vector< std::vector< interval > > *, expression_node, bool, model::walker >, cached_forward_evaluator_base< func_eval_type, expression_node, double, model::walker >, cached_forward_evaluator_base< func_islp_eval_type, expression_node, func_islp_return_type, model::walker >, cached_forward_evaluator_base< cinterval_eval_type, expression_node, cinterval, model::walker >, cached_forward_evaluator_base< interval_eval_type, expression_node, interval, model::walker >, cached_forward_evaluator_base< func_h_eval_type, expression_node, func_h_eval_ret, model::walker >, cached_forward_evaluator_base< func_d_eval_type, expression_node, double, model::walker >, cached_forward_evaluator_base< std::vector< std::vector< double > > *, expression_node, bool, model::walker >, cached_forward_evaluator_base< model::detect_0chain_visitor_st, expression_node, std::pair< unsigned int, unsigned int >, model::walker >, cached_forward_evaluator_base< infbound_eval_type, expression_node, infbound, model::walker >, cached_forward_evaluator_base< b_interval_eval_type, expression_node, b_interval, model::walker >, cached_forward_evaluator_base< analyticd_eval_type, expression_node, analyticd, model::walker >, cached_forward_evaluator_base< func_id_eval_type, expression_node, interval, model::walker >, cached_backward_evaluator_base< ider_eval_type, expression_node, bool, model::walker >, cached_backward_evaluator_base< islp_eval_type, expression_node, bool, model::walker >, cached_backward_evaluator_base< lincoeff_visitor_st, expression_node, lincoeff_visitor_ret, model::walker >, and cached_backward_evaluator_base< der_eval_type, expression_node, bool, model::walker >.

Definition at line 249 of file evaluator.h.

### 6.5.2 Constructor & Destructor Documentation

#### 6.5.2.1 template<class _Tp, class _NData, class _Result, class _Walker> _evaluator_base< _Tp, _NData, _Result, _Walker >::_evaluator_base () [inline]

Definition at line 255 of file evaluator.h.

#### 6.5.2.2 template<class _Tp, class _NData, class _Result, class _Walker> _evaluator_base< _Tp, _NData, _Result, _Walker >::_evaluator_base (const _Tp & __x) [inline]

Definition at line 256 of file evaluator.h.

#### 6.5.2.3 template<class _Tp, class _NData, class _Result, class _Walker> _evaluator_base< _Tp, _NData, _Result, _Walker >::_evaluator_base (const _Self & __x) [inline]

Definition at line 257 of file evaluator.h.

#### 6.5.2.4 template<class _Tp, class _NData, class _Result, class _Walker> virtual _evaluator_base< _Tp, _NData, _Result, _Walker >::~_evaluator_base () [inline, virtual]

Definition at line 259 of file evaluator.h.

### 6.5.3 Member Function Documentation

#### 6.5.3.1 template<class _Tp, class _NData, class _Result, class _Walker> virtual int _evaluator_base< _Tp, _NData, _Result, _Walker >::collect (const node_data_type & __*data*, const return_value & __*cresult*) [inline, virtual]

Reimplemented in forward_evaluator_base< _Tp, _NData, _Result, _Walker >, backward_evaluator_base< _Tp, _NData, _Result, _Walker >, cached_forward_evaluator_base< _Tp, _NData, _Result, _Walker >, cached_backward_evaluator_base< _Tp, _NData, _Result, _Walker >, cached_forward_evaluator_base< std::vector< std::vector< interval > > *, expression_node, bool, model::walker >, cached_forward_evaluator_base< func_eval_type, expression_node, double, model::walker >, cached_forward_evaluator_base< func_islp_eval_type, expression_node, func_islp_return_type, model::walker >, cached_forward_evaluator_base< cinterval_eval_type, expression_node, cinterval, model::walker >, cached_forward_evaluator_base< interval_eval_type, expression_node, interval, model::walker >, cached_forward_evaluator_base< func_h_eval_type, expression_node, func_h_eval_ret, model::walker >, cached_forward_evaluator_base< func_d_eval_type, expression_node, double, model::walker >, cached_forward_evaluator_base< std::vector< std::vector< double > > *, expression_node, bool, model::walker >, cached_forward_evaluator_base< model::detect_0chain_visitor_st, expression_node, std::pair< unsigned int, unsigned int >, model::walker >, cached_forward_evaluator_base< infbound_eval_type, expression_node, infbound, model::walker >, cached_forward_evaluator_base< b_interval_eval_type, expression_node, b_interval, model::walker >, cached_forward_evaluator_base< analyticd_eval_type, expression_node, analyticd, model::walker >, cached_forward_evaluator_base< func_id_eval_type, expression_node, interval, model::walker >, cached_backward_evaluator_base< ider_eval_type, expression_node, bool, model::walker >, cached_backward_evaluator_base< islp_eval_type, expression_node, bool, model::walker >, cached_backward_evaluator_base< lincoeff_visitor_st, expression_node, lincoeff_visitor_ret, model::walker >, and cached_backward_evaluator_base< der_eval_type, expression_node, bool, model::walker >.

Definition at line 265 of file evaluator.h.

#### 6.5.3.2 template<class _Tp, class _NData, class _Result, class _Walker> virtual void _evaluator_base< _Tp, _NData, _Result, _Walker >::postorder (const node_data_type & __*data*) [inline, virtual]

Reimplemented in forward_evaluator_base< _Tp, _NData, _Result, _Walker >, backward_evaluator_base< _Tp, _NData, _Result, _Walker >, cached_forward_evaluator_base< _Tp, _NData, _Result, _Walker >, cached_backward_evaluator_base< _Tp, _NData, _Result, _Walker >, cached_forward_evaluator_base<

std::vector< std::vector< interval > > *, expression_node, bool, model::walker >, cached_forward_evaluator_base< func_eval_type, expression_node, double, model::walker >, cached_forward_evaluator_base< func_islp_eval_type, expression_node, func_islp_return_type, model::walker >, cached_forward_evaluator_base< cinterval_eval_type, expression_node, cinterval, model::walker >, cached_forward_evaluator_base< interval_eval_type, expression_node, interval, model::walker >, cached_forward_evaluator_base< func_h_eval_type, expression_node, func_h_eval_ret, model::walker >, cached_forward_evaluator_base< func_d_eval_type, expression_node, double, model::walker >, cached_forward_evaluator_base< std::vector< std::vector< double > > *, expression_node, bool, model::walker >, cached_forward_evaluator_base< model::detect_0chain_visitor_st, expression_node, std::pair< unsigned int, unsigned int >, model::walker >, cached_forward_evaluator_base< infbound_eval_type, expression_node, infbound, model::walker >, cached_forward_evaluator_base< b_interval_eval_type, expression_node, b_interval, model::walker >, cached_forward_evaluator_base< analyticd_eval_type, expression_node, analyticd, model::walker >, cached_forward_evaluator_base< func_id_eval_type, expression_node, interval, model::walker >, cached_backward_evaluator_base< ider_eval_type, expression_node, bool, model::walker >, cached_backward_evaluator_base< islp_eval_type, expression_node, bool, model::walker >, cached_backward_evaluator_base< lincoeff_visitor_st, expression_node, lincoeff_visitor_ret, model::walker >, and cached_backward_evaluator_base< der_eval_type, expression_node, bool, model::walker >.

Definition at line 268 of file evaluator.h.

#### 6.5.3.3 template<class _Tp, class _NData, class _Result, class _Walker> virtual return_value _evaluator_base< _Tp, _NData, _Result, _Walker >::value () [inline, virtual]

Reimplemented in forward_evaluator_base< _Tp, _NData, _Result, _Walker >, backward_evaluator_base< _Tp, _NData, _Result, _Walker >, cached_forward_evaluator_base< _Tp, _NData, _Result, _Walker >, cached_backward_evaluator_base< _Tp, _NData, _Result, _Walker >, cached_forward_evaluator_base< std::vector< std::vector< interval > > *, expression_node, bool, model::walker >, cached_forward_evaluator_base< func_eval_type, expression_node, double, model::walker >, cached_forward_evaluator_base< func_islp_eval_type, expression_node, func_islp_return_type, model::walker >, cached_forward_evaluator_base< cinterval_eval_type, expression_node, cinterval, model::walker >, cached_forward_evaluator_base< interval_eval_type, expression_node, interval, model::walker >, cached_forward_evaluator_base< func_h_eval_type, expression_node, func_h_eval_ret, model::walker >, cached_forward_evaluator_base< func_d_eval_type, expression_node, double, model::walker >, cached_forward_evaluator_base< std::vector< std::vector< double > > *, expression_node, bool, model::walker >, cached_forward_evaluator_base< model::detect_0chain_visitor_st, expression_node, std::pair< unsigned int, unsigned int >, model::walker >, cached_forward_evaluator_base< infbound_eval_type, expression_node, infbound, model::walker >, cached_forward_evaluator_base< b_interval_eval_type, expression_node, b_interval, model::walker >, cached_forward_evaluator_base< analyticd_eval_type, expression_node, analyticd, model::walker >, cached_forward_evaluator_base< func_id_eval_type, expression_node, interval, model::walker >, cached_backward_evaluator_base< ider_eval_type, expression_node, bool, model::walker >, cached_backward_evaluator_base< islp_eval_type, expression_node, bool, model::walker >, cached_backward_evaluator_base< lincoeff_visitor_st, expression_node, lincoeff_visitor_ret, model::walker >, and cached_backward_evaluator_base< der_eval_type, expression_node, bool, model::walker >.

Definition at line 262 of file evaluator.h.

#### 6.5.3.4 template<class _Tp, class _NData, class _Result, class _Walker> virtual int _evaluator_base< _Tp, _NData, _Result, _Walker >::vcollect (const return_value & *__cresult*) [inline, virtual]

Reimplemented in forward_evaluator_base< _Tp, _NData, _Result, _Walker >, backward_evaluator_base< _Tp, _NData, _Result, _Walker >, cached_forward_evaluator_base< _Tp, _NData, _Result, _Walker >, cached_backward_evaluator_base< _Tp, _NData, _Result, _Walker >, cached_forward_evaluator_base< std::vector< std::vector< interval > > *, expression_node, bool, model::walker >, cached_forward_

evaluator_base< func_eval_type, expression_node, double, model::walker >, cached_forward_evaluator_base< func_islp_eval_type, expression_node, func_islp_return_type, model::walker >, cached_forward_evaluator_base< cinterval_eval_type, expression_node, cinterval, model::walker >, cached_forward_evaluator_base< interval_eval_type, expression_node, interval, model::walker >, cached_forward_evaluator_base< func_h_eval_type, expression_node, func_h_eval_ret, model::walker >, cached_forward_evaluator_base< func_d_eval_type, expression_node, double, model::walker >, cached_forward_evaluator_base< std::vector< std::vector< double > > *, expression_node, bool, model::walker >, cached_forward_evaluator_base< model::detect_0chain_visitor_st, expression_node, std::pair< unsigned int, unsigned int >, model::walker >, cached_forward_evaluator_base< infbound_eval_type, expression_node, infbound, model::walker >, cached_forward_evaluator_base< b_interval_eval_type, expression_node, b_interval, model::walker >, cached_forward_evaluator_base< analyticd_eval_type, expression_node, analyticd, model::walker >, cached_forward_evaluator_base< func_id_eval_type, expression_node, interval, model::walker >, cached_backward_evaluator_base< ider_eval_type, expression_node, bool, model::walker >, cached_backward_evaluator_base< islp_eval_type, expression_node, bool, model::walker >, cached_backward_evaluator_base< lincoeff_visitor_st, expression_node, lincoeff_visitor_ret, model::walker >, and cached_backward_evaluator_base< der_eval_type, expression_node, bool, model::walker >.

Definition at line 264 of file evaluator.h.

#### 6.5.3.5 template<class _Tp, class _NData, class _Result, class _Walker> virtual return_value _evaluator_base< _Tp, _NData, _Result, _Walker >::vvalue () [inline, virtual]

Reimplemented in forward_evaluator_base< _Tp, _NData, _Result, _Walker >, backward_evaluator_base< _Tp, _NData, _Result, _Walker >, cached_forward_evaluator_base< _Tp, _NData, _Result, _Walker >, cached_backward_evaluator_base< _Tp, _NData, _Result, _Walker >, cached_forward_evaluator_base< std::vector< std::vector< interval > > *, expression_node, bool, model::walker >, cached_forward_evaluator_base< func_eval_type, expression_node, double, model::walker >, cached_forward_evaluator_base< func_islp_eval_type, expression_node, func_islp_return_type, model::walker >, cached_forward_evaluator_base< cinterval_eval_type, expression_node, cinterval, model::walker >, cached_forward_evaluator_base< interval_eval_type, expression_node, interval, model::walker >, cached_forward_evaluator_base< func_h_eval_type, expression_node, func_h_eval_ret, model::walker >, cached_forward_evaluator_base< func_d_eval_type, expression_node, double, model::walker >, cached_forward_evaluator_base< std::vector< std::vector< double > > *, expression_node, bool, model::walker >, cached_forward_evaluator_base< model::detect_0chain_visitor_st, expression_node, std::pair< unsigned int, unsigned int >, model::walker >, cached_forward_evaluator_base< infbound_eval_type, expression_node, infbound, model::walker >, cached_forward_evaluator_base< b_interval_eval_type, expression_node, b_interval, model::walker >, cached_forward_evaluator_base< analyticd_eval_type, expression_node, analyticd, model::walker >, cached_forward_evaluator_base< func_id_eval_type, expression_node, interval, model::walker >, cached_backward_evaluator_base< ider_eval_type, expression_node, bool, model::walker >, cached_backward_evaluator_base< islp_eval_type, expression_node, bool, model::walker >, cached_backward_evaluator_base< lincoeff_visitor_st, expression_node, lincoeff_visitor_ret, model::walker >, and cached_backward_evaluator_base< der_eval_type, expression_node, bool, model::walker >.

Definition at line 261 of file evaluator.h.

### 6.5.4 Member Data Documentation

#### 6.5.4.1 template<class _Tp, class _NData, class _Result, class _Walker> _Tp _evaluator_base< _Tp, _NData, _Result, _Walker >::eval_data [protected]

Definition at line 252 of file evaluator.h.

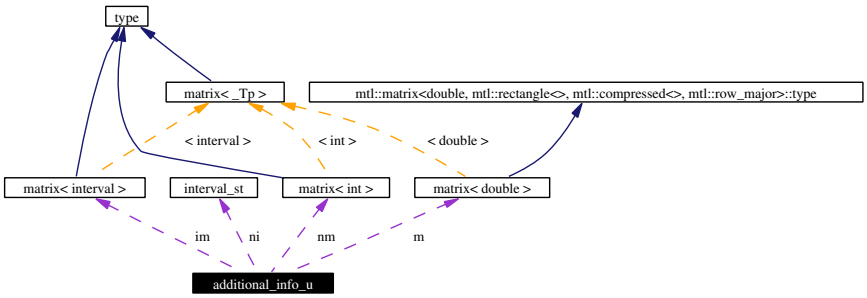
The documentation for this class was generated from the following file:

- evaluator.h

## 6.6 additional_info_u Class Reference

`#include <addinfo.h>`

Collaboration diagram for additional_info_u:




### Public Methods

- additional_info_u ()
- additional_info_u (bool __x)
- additional_info_u (int __x)
- additional_info_u (unsigned int __x)
- additional_info_u (double __x)
- additional_info_u (interval __x)
- additional_info_u (const char *__cp)
- additional_info_u (const std::string &__x)
- additional_info_u (const std::vector< bool > &__x)
- additional_info_u (const std::vector< int > &__x)
- additional_info_u (const std::vector< unsigned int > &__x)
- additional_info_u (const std::vector< double > &__x)
- additional_info_u (const std::vector< interval > &__x)
- additional_info_u (const matrix< double > &__x)
- additional_info_u (const matrix< int > &__x)
- additional_info_u (const matrix< interval > &__x)
- ~additional_info_u ()
- additional_info_u (const additional_info_u &__a)
- additional_info_u & operator= (bool __x)
- additional_info_u & operator= (int __x)
- additional_info_u & operator= (unsigned int __x)
- additional_info_u & operator= (double __x)
- additional_info_u & operator= (interval __x)
- additional_info_u & operator= (const std::string &__x)
- additional_info_u & operator= (const char *__x)
- additional_info_u & operator= (const std::vector< bool > &__x)
- additional_info_u & operator= (const std::vector< int > &__x)
- additional_info_u & operator= (const std::vector< unsigned int > &__x)
- additional_info_u & operator= (const std::vector< double > &__x)
- additional_info_u & operator= (const std::vector< interval > &__x)
- additional_info_u & operator= (const matrix< double > &__x)

- additional_info_u & operator= (const matrix< int > &__x)
- additional_info_u & operator= (const matrix< interval > &__x)
- additional_info_u & operator= (const additional_info_u &__a)
- additional_info_u & clear ()
- additional_info_u & set_m (matrix< double > *__m)
- additional_info_u & set_im (matrix< interval > *__m)
- additional_info_u & set_nm (matrix< int > *__m)
- bool nb () const
- int nn () const
- unsigned int nu () const
- double nd () const
- interval ni () const
- std::string & s () const
- std::vector< bool > & b () const
- std::vector< int > & n () const
- std::vector< unsigned int > & u () const
- std::vector< double > & d () const
- std::vector< interval > & i () const
- matrix< double > & m () const
- matrix< int > & nm () const
- matrix< interval > & im () const
- bool is_allocated () const
- bool empty () const
- int has_type ()

### 6.6.1 Constructor & Destructor Documentation

#### 6.6.1.1 additional_info_u::additional_info_u () [inline]

Definition at line 172 of file addinfo.h.

#### 6.6.1.2 additional_info_u::additional_info_u (bool __x) [inline]

Definition at line 173 of file addinfo.h.

#### 6.6.1.3 additional_info_u::additional_info_u (int __x) [inline]

Definition at line 175 of file addinfo.h.

#### 6.6.1.4 additional_info_u::additional_info_u (unsigned int __x) [inline]

Definition at line 177 of file addinfo.h.

#### 6.6.1.5 additional_info_u::additional_info_u (double __x) [inline]

Definition at line 179 of file addinfo.h.

#### 6.6.1.6 additional_info_u::additional_info_u (interval __x) [inline]

Definition at line 181 of file addinfo.h.

#### 6.6.1.7 additional_info_u::additional_info_u (const char * __cp) [inline]

Definition at line 183 of file addinfo.h.

#### 6.6.1.8 additional_info_u::additional_info_u (const std::string & __x) [inline]

Definition at line 185 of file addinfo.h.

#### 6.6.1.9 additional_info_u::additional_info_u (const std::vector< bool > & __x) [inline]

Definition at line 187 of file addinfo.h.

#### 6.6.1.10 additional_info_u::additional_info_u (const std::vector< int > & __x) [inline]

Definition at line 189 of file addinfo.h.

#### 6.6.1.11 additional_info_u::additional_info_u (const std::vector< unsigned int > & __x) [inline]

Definition at line 191 of file addinfo.h.

#### 6.6.1.12 additional_info_u::additional_info_u (const std::vector< double > & __x) [inline]

Definition at line 193 of file addinfo.h.

#### 6.6.1.13 additional_info_u::additional_info_u (const std::vector< interval > & __x) [inline]

Definition at line 196 of file addinfo.h.

#### 6.6.1.14 additional_info_u::additional_info_u (const matrix< double > & __x) [inline]

Definition at line 199 of file addinfo.h.

#### 6.6.1.15 additional_info_u::additional_info_u (const matrix< int > & __x) [inline]

Definition at line 202 of file addinfo.h.

#### 6.6.1.16 additional_info_u::additional_info_u (const matrix< interval > & __x) [inline]

Definition at line 205 of file addinfo.h.

#### 6.6.1.17 additional_info_u::~additional_info_u () [inline]

Definition at line 209 of file addinfo.h.

#### 6.6.1.18 additional_info_u::additional_info_u (const additional_info_u & __a) [inline]

Definition at line 212 of file addinfo.h.

### 6.6.2 Member Function Documentation

#### 6.6.2.1 std::vector<bool>& additional_info_u::b () const [inline]

Definition at line 352 of file addinfo.h.

#### 6.6.2.2 additional_info_u& additional_info_u::clear () [inline]

Definition at line 307 of file addinfo.h.

#### 6.6.2.3 std::vector<double>& additional_info_u::d () const [inline]

Definition at line 355 of file addinfo.h.

#### 6.6.2.4 bool additional_info_u::empty () const [inline]

Definition at line 362 of file addinfo.h.

#### 6.6.2.5 int additional_info_u::has_type () [inline]

Definition at line 363 of file addinfo.h.

#### 6.6.2.6 std::vector<interval>& additional_info_u::i () const [inline]

Definition at line 356 of file addinfo.h.

#### 6.6.2.7 matrix<interval>& additional_info_u::im () const [inline]

Definition at line 359 of file addinfo.h.

#### 6.6.2.8 bool additional_info_u::is_allocated () const [inline]

Definition at line 361 of file addinfo.h.

#### 6.6.2.9 matrix<double>& additional_info_u::m () const [inline]

Definition at line 357 of file addinfo.h.

#### 6.6.2.10 std::vector<int>& additional_info_u::n () const [inline]

Definition at line 353 of file addinfo.h.

#### 6.6.2.11 bool additional_info_u::nb () const [inline]

Definition at line 346 of file addinfo.h.

#### 6.6.2.12 double additional_info_u::nd () const [inline]

Definition at line 349 of file addinfo.h.

#### 6.6.2.13 interval additional_info_u::ni () const [inline]

Definition at line 350 of file addinfo.h.

#### 6.6.2.14 matrix<int>& additional_info_u::nm () const [inline]

Definition at line 358 of file addinfo.h.

#### 6.6.2.15 int additional_info_u::nn () const [inline]

Definition at line 347 of file addinfo.h.

#### 6.6.2.16 unsigned int additional_info_u::nu () const [inline]

Definition at line 348 of file addinfo.h.

#### 6.6.2.17 additional_info_u& additional_info_u::operator= (const additional_info_u & *__a*) [inline]

Definition at line 300 of file addinfo.h.

#### 6.6.2.18 additional_info_u& additional_info_u::operator= (const matrix< interval > & *__x*) [inline]

Definition at line 294 of file addinfo.h.

#### 6.6.2.19 additional_info_u& additional_info_u::operator= (const matrix< int > & *__x*) [inline]

Definition at line 288 of file addinfo.h.

#### 6.6.2.20 additional_info_u& additional_info_u::operator= (const matrix< double > & *__x*) [inline]

Definition at line 282 of file addinfo.h.

#### 6.6.2.21 additional_info_u& additional_info_u::operator= (const std::vector< interval > & *__x*) [inline]

Definition at line 276 of file addinfo.h.

#### 6.6.2.22 additional_info_u& additional_info_u::operator= (const std::vector< double > & *__x*) [inline]

Definition at line 270 of file addinfo.h.

#### 6.6.2.23 additional_info_u& additional_info_u::operator= (const std::vector< unsigned int > & *__x*) [inline]

Definition at line 264 of file addinfo.h.

#### 6.6.2.24 additional_info_u& additional_info_u::operator= (const std::vector< int > & *__x*) [inline]

Definition at line 258 of file addinfo.h.

#### 6.6.2.25 additional_info_u& additional_info_u::operator= (const std::vector< bool > & _x) [inline]

Definition at line 252 of file addinfo.h.

#### 6.6.2.26 additional_info_u& additional_info_u::operator= (const char * _x) [inline]

Definition at line 246 of file addinfo.h.

#### 6.6.2.27 additional_info_u& additional_info_u::operator= (const std::string & _x) [inline]

Definition at line 240 of file addinfo.h.

#### 6.6.2.28 additional_info_u& additional_info_u::operator= (interval _x) [inline]

Definition at line 235 of file addinfo.h.

#### 6.6.2.29 additional_info_u& additional_info_u::operator= (double _x) [inline]

Definition at line 230 of file addinfo.h.

#### 6.6.2.30 additional_info_u& additional_info_u::operator= (unsigned int _x) [inline]

Definition at line 225 of file addinfo.h.

#### 6.6.2.31 additional_info_u& additional_info_u::operator= (int _x) [inline]

Definition at line 220 of file addinfo.h.

#### 6.6.2.32 additional_info_u& additional_info_u::operator= (bool _x) [inline]

Definition at line 215 of file addinfo.h.

#### 6.6.2.33 std::string& additional_info_u::s () const [inline]

Definition at line 351 of file addinfo.h.

#### 6.6.2.34 additional_info_u& additional_info_u::set_im (matrix< interval > * _m) [inline]

Definition at line 327 of file addinfo.h.

#### 6.6.2.35 additional_info_u& additional_info_u::set_m (matrix< double > * _m) [inline]

Definition at line 317 of file addinfo.h.

#### 6.6.2.36 additional_info_u& additional_info_u::set_nm (matrix< int > * _m) [inline]

Definition at line 337 of file addinfo.h.

#### 6.6.2.37 std::vector<unsigned int>& additional_info_u::u () const [inline]

Definition at line 354 of file addinfo.h.

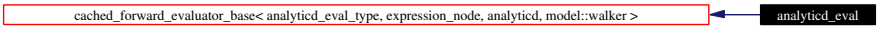
The documentation for this class was generated from the following file:

- addinfo.h

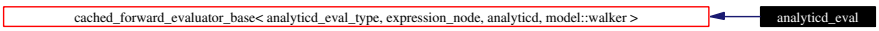
## 6.7 analyticd_eval Class Reference

`#include <ade_evaluator.h>`

Inheritance diagram for analyticd_eval:




Collaboration diagram for analyticd_eval:




### Public Types

- typedef cached_evaluator_base< analyticd_eval_type, expression_node, analyticd, model::walker >::node_data_type node_data_type
- typedef cached_evaluator_base< analyticd_eval_type, expression_node, analyticd, model::walker >::return_value return_value
- typedef cached_evaluator_base< analyticd_eval_type, expression_node, analyticd, model::walker >::walker walker
- typedef analyticd_eval_type data_type

### Public Methods

- analyticd_eval (const std::vector< analyticd > &__x, const variable_indicator &__v, const model &__m, std::vector< analyticd > *__c)
- analyticd_eval (const analyticd_eval &__v)
- ~analyticd_eval ()
- model::walker short_cut_to (const expression_node &__data)
- void initialize ()
- int initialize (const expression_node &__data)
- void calculate (const expression_node &__data)
- void retrieve_from_cache (const expression_node &__data)
- int update (const analyticd &__rval)
- int update (const expression_node &__data, const analyticd &__rval)
- analyticd calculate_value (bool eval_all)
- int preorder (const node_data_type &__data)
- void postorder (const node_data_type &__data)
- int collect (const node_data_type &__data, const return_value &__rval)
- int vcollect (const return_value &__rval)
- return_value value ()
- return_value vvalue ()

- void vinit ()
- virtual int initialize (const node_data_type &__data)
- virtual void calculate (const node_data_type &__data)
- virtual void retrieve_from_cache (const node_data_type &__data)
- virtual void cleanup (const node_data_type &__data)
- virtual int update (const node_data_type &__data, const return_value &__rval)
- virtual int update (const return_value &__rval)
- virtual walker short_cut_to (const node_data_type &__data) PURE_VIRTUAL public

**Protected Methods**

- bool is_cached (const node_data_type &__data)

**Protected Attributes**

- const variable_indicator * v_ind
- analyticd_eval_type eval_data

### 6.7.1 Member Typedef Documentation

#### 6.7.1.1 typedef analyticd_eval_type _evaluator_base< analyticd_eval_type, expression_node, analyticd, model::walker >::data_type [inherited]

Definition at line 245 of file evaluator.h.

#### 6.7.1.2 typedef cached_evaluator_base<analyticd_eval_type, expression_node, analyticd,model::walker>::node_data_type cached_forward_evaluator_base< analyticd_eval_type, expression_node, analyticd, model::walker >::node_data_type [inherited]

Reimplemented from cached_evaluator_base< analyticd_eval_type, expression_node, analyticd, model::walker >.

Definition at line 396 of file evaluator.h.

#### 6.7.1.3 typedef cached_evaluator_base<analyticd_eval_type, expression_node, analyticd,model::walker>::return_value cached_forward_evaluator_base< analyticd_eval_type, expression_node, analyticd, model::walker >::return_value [inherited]

Reimplemented from cached_evaluator_base< analyticd_eval_type, expression_node, analyticd, model::walker >.

Definition at line 398 of file evaluator.h.

#### 6.7.1.4 typedef cached_evaluator_base<analyticd_eval_type, expression_node, analyticd,model::walker>::walker cached_forward_evaluator_base< analyticd_eval_type, expression_node, analyticd, model::walker >::walker [inherited]

Reimplemented from cached_evaluator_base< analyticd_eval_type, expression_node, analyticd, model::walker >.

Definition at line 400 of file evaluator.h.

### 6.7.2 Constructor & Destructor Documentation

#### 6.7.2.1 analyticd_eval::analyticd_eval (const std::vector< analyticd > & *__x*, const variable_indicator & *__v*, const model & *__m*, std::vector< analyticd > * *__c*) [inline]

Definition at line 111 of file ade_evaluator.h.

#### 6.7.2.2 analyticd_eval::analyticd_eval (const analyticd_eval & *__v*) [inline]

Definition at line 123 of file ade_evaluator.h.

#### 6.7.2.3 analyticd_eval::~analyticd_eval () [inline]

Definition at line 125 of file ade_evaluator.h.

### 6.7.3 Member Function Documentation

#### 6.7.3.1 virtual void cached_forward_evaluator_base< analyticd_eval_type, expression_node, analyticd, model::walker >::calculate (const node_data_type & *__data*) [inline, virtual, inherited]

Definition at line 430 of file evaluator.h.

#### 6.7.3.2 void analyticd_eval::calculate (const expression_node & *__data*) [inline]

Definition at line 188 of file ade_evaluator.h.

#### 6.7.3.3 analyticd analyticd_eval::calculate_value (bool *eval_all*) [inline, virtual]

Reimplemented from cached_forward_evaluator_base< analyticd_eval_type, expression_node, analyticd, model::walker >.

Definition at line 732 of file ade_evaluator.h.

#### 6.7.3.4 virtual void cached_forward_evaluator_base< analyticd_eval_type, expression_node, analyticd, model::walker >::cleanup (const node_data_type & *__data*) [inline, virtual, inherited]

Definition at line 432 of file evaluator.h.

#### 6.7.3.5 int cached_forward_evaluator_base< analyticd_eval_type, expression_node, analyticd, model::walker >::collect (const node_data_type & *__data*, const return_value & *__rval*) [inline, virtual, inherited]

Reimplemented from _evaluator_base< analyticd_eval_type, expression_node, analyticd, model::walker >.

Definition at line 419 of file evaluator.h.

#### 6.7.3.6 virtual int cached_forward_evaluator_base< analyticd_eval_type, expression_node, analyticd, model::walker >::initialize (const node_data_type & *__data*) [inline, virtual, inherited]

Definition at line 429 of file evaluator.h.

#### 6.7.3.7 int analyticd_eval::initialize (const expression_node & *__data*) [inline]

Definition at line 132 of file ade_evaluator.h.

#### 6.7.3.8 void analyticd_eval::initialize () [inline, virtual]

Reimplemented from cached_forward_evaluator_base< analyticd_eval_type, expression_node, analyticd, model::walker >.

Definition at line 130 of file ade_evaluator.h.

#### 6.7.3.9 bool analyticd_eval::is_cached (const node_data_type & *__data*) [inline, protected, virtual]

Reimplemented from cached_forward_evaluator_base< analyticd_eval_type, expression_node, analyticd, model::walker >.

Definition at line 65 of file ade_evaluator.h.

#### 6.7.3.10 void cached_forward_evaluator_base< analyticd_eval_type, expression_node, analyticd, model::walker >::postorder (const node_data_type & *__data*) [inline, virtual, inherited]

Reimplemented from _evaluator_base< analyticd_eval_type, expression_node, analyticd, model::walker >.

Definition at line 417 of file evaluator.h.

#### 6.7.3.11 int cached_forward_evaluator_base< analyticd_eval_type, expression_node, analyticd, model::walker >::preorder (const node_data_type & *__data*) [inline, virtual, inherited]

Reimplemented from cached_evaluator_base< analyticd_eval_type, expression_node, analyticd, model::walker >.

Definition at line 408 of file evaluator.h.

#### 6.7.3.12 virtual void cached_forward_evaluator_base< analyticd_eval_type, expression_node, analyticd, model::walker >::retrieve_from_cache (const node_data_type & *__data*) [inline, virtual, inherited]

Definition at line 431 of file evaluator.h.

#### 6.7.3.13 void analyticd_eval::retrieve_from_cache (const expression_node & *__data*) [inline]

Definition at line 200 of file ade_evaluator.h.

#### 6.7.3.14 virtual walker cached_evaluator_base< analyticd_eval_type, expression_node, analyticd, model::walker >::short_cut_to (const node_data_type & *__data*) [inline, virtual, inherited]

Definition at line 303 of file evaluator.h.

#### 6.7.3.15 model::walker analyticd_eval::short_cut_to (const expression_node & *__data*) [inline]

Definition at line 127 of file ade_evaluator.h.

#### 6.7.3.16 virtual int cached_forward_evaluator_base< analyticd_eval_type, expression_node, analyticd, model::walker >::update (const return_value & __rval) [inline, virtual, inherited]

Definition at line 435 of file evaluator.h.

#### 6.7.3.17 virtual int cached_forward_evaluator_base< analyticd_eval_type, expression_node, analyticd, model::walker >::update (const node_data_type & __data, const return_value & __rval) [inline, virtual, inherited]

Definition at line 433 of file evaluator.h.

#### 6.7.3.18 int analyticd_eval::update (const expression_node & __data, const analyticd & __rval) [inline]

Definition at line 225 of file ade_evaluator.h.

#### 6.7.3.19 int analyticd_eval::update (const analyticd & __rval) [inline]

Definition at line 219 of file ade_evaluator.h.

#### 6.7.3.20 return_value cached_forward_evaluator_base< analyticd_eval_type, expression_node, analyticd, model::walker >::value () [inline, virtual, inherited]

Reimplemented from _evaluator_base< analyticd_eval_type, expression_node, analyticd, model::walker >.

Definition at line 423 of file evaluator.h.

#### 6.7.3.21 int cached_forward_evaluator_base< analyticd_eval_type, expression_node, analyticd, model::walker >::vcollect (const return_value & __rval) [inline, virtual, inherited]

Reimplemented from _evaluator_base< analyticd_eval_type, expression_node, analyticd, model::walker >.

Definition at line 421 of file evaluator.h.

#### 6.7.3.22 void cached_forward_evaluator_base< analyticd_eval_type, expression_node, analyticd, model::walker >::vinit () [inline, inherited]

Definition at line 425 of file evaluator.h.

#### 6.7.3.23 return_value cached_forward_evaluator_base< analyticd_eval_type, expression_node, analyticd, model::walker >::vvalue () [inline, virtual, inherited]

Reimplemented from _evaluator_base< analyticd_eval_type, expression_node, analyticd, model::walker >.

Definition at line 424 of file evaluator.h.

### 6.7.4 Member Data Documentation

#### 6.7.4.1 analyticd_eval_type _evaluator_base< analyticd_eval_type, expression_node, analyticd, model::walker >::eval_data [protected, inherited]

Definition at line 252 of file evaluator.h.

#### 6.7.4.2 const variable_indicator* cached_evaluator_base< analyticd_eval_type, expression_node, analyticd, model::walker >::v_ind [protected, inherited]

Definition at line 295 of file evaluator.h.

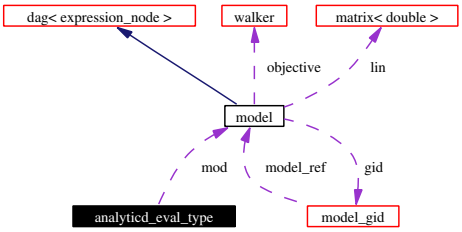
The documentation for this class was generated from the following file:

- ade_evaluator.h

## 6.8 analyticd_eval_type Struct Reference

`#include <ade_evaluator.h>`

Collaboration diagram for analyticd_eval_type:




### Public Attributes

- const std::vector< analyticd > * x
- std::vector< analyticd > * cache
- const model * mod
- void * p
- analyticd d
- int info
- analyticd r
- unsigned int n

### 6.8.1 Member Data Documentation

#### 6.8.1.1 std::vector<analyticd>* analyticd_eval_type::cache

Definition at line 47 of file ade_evaluator.h.

#### 6.8.1.2 analyticd analyticd_eval_type::d

Definition at line 50 of file ade_evaluator.h.

#### 6.8.1.3 int analyticd_eval_type::info

Definition at line 51 of file ade_evaluator.h.

#### 6.8.1.4 const model* analyticd_eval_type::mod

Definition at line 48 of file ade_evaluator.h.

#### 6.8.1.5 unsigned int analyticd_eval_type::n

Definition at line 53 of file ade_evaluator.h.

#### 6.8.1.6 void* analyticd_eval_type::p

Definition at line 49 of file ade_evaluator.h.

#### 6.8.1.7 analyticd analyticd_eval_type::r

Definition at line 52 of file ade_evaluator.h.

#### 6.8.1.8 const std::vector<analyticd>* analyticd_eval_type::x

Definition at line 46 of file ade_evaluator.h.

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

- ade_evaluator.h

## 6.9 annotation Class Reference

`#include <annotation.h>`

### Public Methods

- annotation (const **vdbl::tableid** &_ti, const **vdbl::rowid** &_ri)
- virtual ~annotation ()
- **vdbl::tableid** get_table () const
- **vdbl::rowid** get_entry () const

### 6.9.1 Constructor & Destructor Documentation

#### 6.9.1.1 annotation::annotation (const vdbl::tableid & *_ti*, const vdbl::rowid & *_ri*) [inline]

Definition at line 314 of file annotation.h.

#### 6.9.1.2 virtual annotation::~annotation () [inline, virtual]

Definition at line 316 of file annotation.h.

### 6.9.2 Member Function Documentation

#### 6.9.2.1 vdbl::rowid annotation::get_entry () const [inline]

Definition at line 319 of file annotation.h.

#### 6.9.2.2 vdbl::tableid annotation::get_table () const [inline]

Definition at line 318 of file annotation.h.

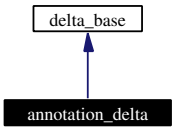
The documentation for this class was generated from the following file:

- annotation.h

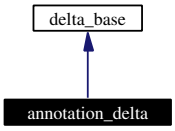
## 6.10 annotation_delta Class Reference

`#include <annotation_delta.h>`

Inheritance diagram for annotation_delta:




Collaboration diagram for annotation_delta:




### Public Methods

- annotation_delta (const std::string &_act)
- annotation_delta (const std::string &_act, const std::vector< annotation > &__a, const std::vector< annotation > &__r)
- annotation_delta (const std::string &_act, const annotation &_ad)
- annotation_delta (const std::string &_act, bool _dummy, const annotation &_rm)
- annotation_delta (const annotation_delta &__d)
- annotation_delta (const char *_act, const std::vector< annotation > &__a, const std::vector< annotation > &__r)
- annotation_delta (const char *_act, const annotation &_ad)
- annotation_delta (const char *_act, bool _dummy, const annotation &_rm)
- virtual annotation_delta * new_copy () const
- virtual void destroy_copy (annotation_delta *__d)
- virtual bool apply (work_node &_x, undelta_base *&_u) const
- virtual void destroy_copy (delta_base *__d)
- delta make_delta (const std::string &a)
- const std::string & get_action () const
- virtual void convert (work_node &_x, delta_base *&_d)
- virtual bool apply3 (work_node &_x, const work_node &_y, undelta_base *&_u) const

### Public Attributes

- std::vector< annotation > add
- std::vector< annotation > rm

### Protected Attributes

- std::string _action

## 6.10.1 Constructor & Destructor Documentation

### 6.10.1.1 annotation_delta::annotation_delta (const std::string & *_act*) [inline]

Definition at line 77 of file annotation_delta.h.

### 6.10.1.2 annotation_delta::annotation_delta (const std::string & *_act*, const std::vector< annotation > & *__a*, const std::vector< annotation > & *__r*) [inline]

Definition at line 79 of file annotation_delta.h.

### 6.10.1.3 annotation_delta::annotation_delta (const std::string & *_act*, const annotation & *_ad*) [inline]

Definition at line 84 of file annotation_delta.h.

### 6.10.1.4 annotation_delta::annotation_delta (const std::string & *_act*, bool *_dummy*, const annotation & *_rm*) [inline]

Definition at line 87 of file annotation_delta.h.

### 6.10.1.5 annotation_delta::annotation_delta (const annotation_delta & *__d*) [inline]

Definition at line 90 of file annotation_delta.h.

### 6.10.1.6 annotation_delta::annotation_delta (const char * *_act*, const std::vector< annotation > & *__a*, const std::vector< annotation > & *__r*) [inline]

Definition at line 98 of file annotation_delta.h.

### 6.10.1.7 annotation_delta::annotation_delta (const char * *_act*, const annotation & *_ad*) [inline]

Definition at line 103 of file annotation_delta.h.

### 6.10.1.8 annotation_delta::annotation_delta (const char * *_act*, bool *_dummy*, const annotation & *_rm*) [inline]

Definition at line 106 of file annotation_delta.h.

### 6.10.2 Member Function Documentation

#### 6.10.2.1 bool annotation_delta::apply (work_node & _x, undelta_base *& _u) const [virtual]

Reimplemented from delta_base.

Definition at line 29 of file annotation_delta.cc.

#### 6.10.2.2 bool delta_base::apply3 (work_node & _x, const work_node & _y, undelta_base *& _u) const [inline, virtual, inherited]

Definition at line 63 of file api_delta.h.

#### 6.10.2.3 virtual void delta_base::convert (work_node & _x, delta_base *& _d) [inline, virtual, inherited]

Reimplemented in table_delta.

Definition at line 107 of file api_deltabase.h.

#### 6.10.2.4 virtual void delta_base::destroy_copy (delta_base * __d) [inline, virtual, inherited]

Definition at line 95 of file api_deltabase.h.

#### 6.10.2.5 virtual void annotation_delta::destroy_copy (annotation_delta * __d) [inline, virtual]

Definition at line 111 of file annotation_delta.h.

#### 6.10.2.6 const std::string& delta_base::get_action () const [inline, inherited]

Definition at line 105 of file api_deltabase.h.

#### 6.10.2.7 delta delta_base::make_delta (const std::string & a) [inline, inherited]

Definition at line 99 of file api_deltabase.h.

#### 6.10.2.8 virtual annotation_delta* annotation_delta::new_copy () const [inline, virtual]

Reimplemented from delta_base.

Definition at line 109 of file annotation_delta.h.

### 6.10.3 Member Data Documentation

#### 6.10.3.1 std::string delta_base::_action [protected, inherited]

Definition at line 86 of file api_deltabase.h.

#### 6.10.3.2 std::vector<annotation> annotation_delta::add

Definition at line 73 of file annotation_delta.h.

#### 6.10.3.3 std::vector<annotation> annotation_delta::rm

Definition at line 74 of file annotation_delta.h.

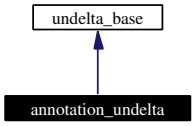
The documentation for this class was generated from the following files:

- annotation_delta.h
- annotation_delta.cc

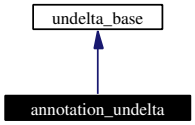
## 6.11 annotation_undelta Class Reference

`#include <annotation_delta.h>`

Inheritance diagram for annotation_undelta:




Collaboration diagram for annotation_undelta:




### Public Methods

- annotation_undelta (const std::vector< annotation > &__a, const std::vector< annotation > &__r)
- annotation_undelta (const std::vector< annotation > &__a)
- annotation_undelta (const annotation &__a)
- annotation_undelta (const annotation_undelta &__d)
- annotation_undelta * new_copy () const
- void destroy_copy (annotation_undelta *__d)
- bool unapply (work_node &_x) const
- virtual void destroy_copy (undelta_base *__d)
- undelta make_undelta ()
- virtual bool unapply3 (work_node &_x, const work_node &_y) const

### Public Attributes

- std::vector< annotation > added_ann
- std::vector< annotation > removed_ann

### Friends

- class annotation_delta

### 6.11.1 Constructor & Destructor Documentation

#### 6.11.1.1 annotation_undelta::annotation_undelta (const std::vector< annotation > & __a, const std::vector< annotation > & __r) [inline]

Definition at line 40 of file annotation_delta.h.

#### 6.11.1.2 annotation_undelta::annotation_undelta (const std::vector< annotation > & __a) [inline]

Definition at line 45 of file annotation_delta.h.

#### 6.11.1.3 annotation_undelta::annotation_undelta (const annotation & __a) [inline]

Definition at line 49 of file annotation_delta.h.

#### 6.11.1.4 annotation_undelta::annotation_undelta (const annotation_undelta & __d) [inline]

Definition at line 53 of file annotation_delta.h.

### 6.11.2 Member Function Documentation

#### 6.11.2.1 virtual void undelta_base::destroy_copy (undelta_base * __d) [inline, virtual, inherited]

Definition at line 146 of file api_deltabase.h.

#### 6.11.2.2 void annotation_undelta::destroy_copy (annotation_undelta * __d) [inline]

Definition at line 63 of file annotation_delta.h.

#### 6.11.2.3 undelta undelta_base::make_undelta () [inline, inherited]

Definition at line 150 of file api_deltabase.h.

#### 6.11.2.4 annotation_undelta* annotation_undelta::new_copy () const [inline, virtual]

Reimplemented from undelta_base.

Definition at line 62 of file annotation_delta.h.

#### 6.11.2.5 bool annotation_undelta::unapply (work_node & _x) const [virtual]

Reimplemented from undelta_base.

Definition at line 53 of file annotation_delta.cc.

#### 6.11.2.6 bool undelta_base::unapply3 (work_node & _x, const work_node & _y) const [inline, virtual, inherited]

Definition at line 69 of file api_delta.h.

### 6.11.3 Friends And Related Function Documentation

#### 6.11.3.1 friend class annotation_delta [friend]

Definition at line 67 of file annotation_delta.h.

### 6.11.4 Member Data Documentation

#### 6.11.4.1 std::vector<annotation> annotation_undelta::added_ann

Definition at line 35 of file annotation_delta.h.

#### 6.11.4.2 std::vector<annotation> annotation_undelta::removed_ann

Definition at line 37 of file annotation_delta.h.

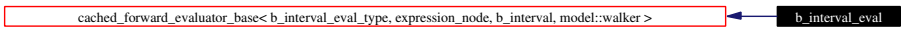
The documentation for this class was generated from the following files:

- annotation_delta.h
- annotation_delta.cc

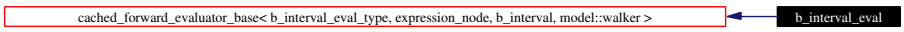
## 6.12 b_interval_eval Class Reference

`#include <bint_evaluator.h>`

Inheritance diagram for b_interval_eval:




Collaboration diagram for b_interval_eval:




### Public Types

- typedef cached_evaluator_base< b_interval_eval_type, expression_node, b_interval, model::walker >::node_data_type node_data_type
- typedef cached_evaluator_base< b_interval_eval_type, expression_node, b_interval, model::walker >::return_value return_value
- typedef cached_evaluator_base< b_interval_eval_type, expression_node, b_interval, model::walker >::walker walker
- typedef b_interval_eval_type data_type

### Public Methods

- b_interval_eval (const std::vector< b_interval > &__x, const variable_indicator &__v, const model &__m, std::vector< b_interval > *__c)
- b_interval_eval (const b_interval_eval &__v)
- ~b_interval_eval ()
- model::walker short_cut_to (const expression_node &__data)

- void initialize ()
- int initialize (const expression_node &__data)
- void calculate (const expression_node &__data)
- void retrieve_from_cache (const expression_node &__data)
- int update (const b_interval &__rval)
- int update (const expression_node &__data, const b_interval &__rval)
- b_interval calculate_value (bool eval_all)
- int preorder (const node_data_type &__data)
- void postorder (const node_data_type &__data)
- int collect (const node_data_type &__data, const return_value &__rval)
- int vcollect (const return_value &__rval)
- return_value value ()
- return_value vvalue ()
- void vinit ()
- virtual int initialize (const node_data_type &__data)
- virtual void calculate (const node_data_type &__data)
- virtual void retrieve_from_cache (const node_data_type &__data)
- virtual void cleanup (const node_data_type &__data)
- virtual int update (const node_data_type &__data, const return_value &__rval)
- virtual int update (const return_value &__rval)
- virtual walker short_cut_to (const node_data_type &__data) PURE_VIRTUAL public

**Protected Methods**

- bool is_cached (const node_data_type &__data)

**Protected Attributes**

- const variable_indicator * v_ind
- b_interval_eval_type eval_data

### 6.12.1 Member Typedef Documentation

#### 6.12.1.1 typedef b_interval_eval_type _evaluator_base< b_interval_eval_type, expression_node, b_interval, model::walker >::data_type [inherited]

Definition at line 245 of file evaluator.h.

#### 6.12.1.2 typedef cached_evaluator_base<b_interval_eval_type, expression_node, b_interval,model::walker>::node_data_type cached_forward_evaluator_base< b_interval_eval_type, expression_node, b_interval, model::walker >::node_data_type [inherited]

Reimplemented from cached_evaluator_base< b_interval_eval_type, expression_node, b_interval, model::walker >.

Definition at line 396 of file evaluator.h.

#### 6.12.1.3 typedef cached_evaluator_base<b_interval_eval_type, expression_node, b_interval,model::walker>::return_value cached_forward_evaluator_base< b_interval_eval_type, expression_node, b_interval, model::walker >::return_value [inherited]

Reimplemented from cached_evaluator_base< b_interval_eval_type, expression_node, b_interval, model::walker >.

Definition at line 398 of file evaluator.h.

#### 6.12.1.4 typedef cached_evaluator_base<b_interval_eval_type, expression_node, b_interval,model::walker>::walker cached_forward_evaluator_base< b_interval_eval_type, expression_node, b_interval, model::walker >::walker [inherited]

Reimplemented from cached_evaluator_base< b_interval_eval_type, expression_node, b_interval, model::walker >.

Definition at line 400 of file evaluator.h.

### 6.12.2 Constructor & Destructor Documentation

#### 6.12.2.1 b_interval_eval::b_interval_eval (const std::vector< b_interval > & *__x*, const variable_indicator & *__v*, const model & *__m*, std::vector< b_interval > * *__c*) [inline]

Definition at line 111 of file bint_evaluator.h.

#### 6.12.2.2 b_interval_eval::b_interval_eval (const b_interval_eval & *__v*) [inline]

Definition at line 123 of file bint_evaluator.h.

#### 6.12.2.3 b_interval_eval::~b_interval_eval () [inline]

Definition at line 125 of file bint_evaluator.h.

### 6.12.3 Member Function Documentation

#### 6.12.3.1 virtual void cached_forward_evaluator_base< b_interval_eval_type, expression_node, b_interval, model::walker >::calculate (const node_data_type & *__data*) [inline, virtual, inherited]

Definition at line 430 of file evaluator.h.

#### 6.12.3.2 void b_interval_eval::calculate (const expression_node & *__data*) [inline]

Definition at line 188 of file bint_evaluator.h.

#### 6.12.3.3 b_interval b_interval_eval::calculate_value (bool *eval_all*) [inline, virtual]

Reimplemented from cached_forward_evaluator_base< b_interval_eval_type, expression_node, b_interval, model::walker >.

Definition at line 732 of file bint_evaluator.h.

#### 6.12.3.4 virtual void cached_forward_evaluator_base< b_interval_eval_type, expression_node, b_interval, model::walker >::cleanup (const node_data_type & *__data*) [inline, virtual, inherited]

Definition at line 432 of file evaluator.h.

#### 6.12.3.5 int cached_forward_evaluator_base< b_interval_eval_type, expression_node, b_interval, model::walker >::collect (const node_data_type & *__data*, const return_value & *__rval*) [inline, virtual, inherited]

Reimplemented from _evaluator_base< b_interval_eval_type, expression_node, b_interval, model::walker >.

Definition at line 419 of file evaluator.h.

#### 6.12.3.6 virtual int cached_forward_evaluator_base< b_interval_eval_type, expression_node, b_interval, model::walker >::initialize (const node_data_type & *__data*) [inline, virtual, inherited]

Definition at line 429 of file evaluator.h.

#### 6.12.3.7 int b_interval_eval::initialize (const expression_node & *__data*) [inline]

Definition at line 132 of file bint_evaluator.h.

#### 6.12.3.8 void b_interval_eval::initialize () [inline, virtual]

Reimplemented from cached_forward_evaluator_base< b_interval_eval_type, expression_node, b_interval, model::walker >.

Definition at line 130 of file bint_evaluator.h.

#### 6.12.3.9 bool b_interval_eval::is_cached (const node_data_type & *__data*) [inline, protected, virtual]

Reimplemented from cached_forward_evaluator_base< b_interval_eval_type, expression_node, b_interval, model::walker >.

Definition at line 65 of file bint_evaluator.h.

#### 6.12.3.10 void cached_forward_evaluator_base< b_interval_eval_type, expression_node, b_interval, model::walker >::postorder (const node_data_type & *__data*) [inline, virtual, inherited]

Reimplemented from _evaluator_base< b_interval_eval_type, expression_node, b_interval, model::walker >.

Definition at line 417 of file evaluator.h.

#### 6.12.3.11 int cached_forward_evaluator_base< b_interval_eval_type, expression_node, b_interval, model::walker >::preorder (const node_data_type & *__data*) [inline, virtual, inherited]

Reimplemented from cached_evaluator_base< b_interval_eval_type, expression_node, b_interval, model::walker >.

Definition at line 408 of file evaluator.h.

**6.12.3.12 virtual void cached_forward_evaluator_base< b_interval_eval_type, expression_node, b_interval, model::walker >::retrieve_from_cache (const node_data_type & __data) [inline, virtual, inherited]**

Definition at line 431 of file evaluator.h.

**6.12.3.13 void b_interval_eval::retrieve_from_cache (const expression_node & __data) [inline]**

Definition at line 200 of file bint_evaluator.h.

**6.12.3.14 virtual walker cached_evaluator_base< b_interval_eval_type, expression_node, b_interval, model::walker >::short_cut_to (const node_data_type & __data) [inline, virtual, inherited]**

Definition at line 303 of file evaluator.h.

**6.12.3.15 model::walker b_interval_eval::short_cut_to (const expression_node & __data) [inline]**

Definition at line 127 of file bint_evaluator.h.

**6.12.3.16 virtual int cached_forward_evaluator_base< b_interval_eval_type, expression_node, b_interval, model::walker >::update (const return_value & __rval) [inline, virtual, inherited]**

Definition at line 435 of file evaluator.h.

**6.12.3.17 virtual int cached_forward_evaluator_base< b_interval_eval_type, expression_node, b_interval, model::walker >::update (const node_data_type & __data, const return_value & __rval) [inline, virtual, inherited]**

Definition at line 433 of file evaluator.h.

**6.12.3.18 int b_interval_eval::update (const expression_node & __data, const b_interval & __rval) [inline]**

Definition at line 225 of file bint_evaluator.h.

**6.12.3.19 int b_interval_eval::update (const b_interval & __rval) [inline]**

Definition at line 219 of file bint_evaluator.h.

**6.12.3.20 return_value cached_forward_evaluator_base< b_interval_eval_type, expression_node, b_interval, model::walker >::value () [inline, virtual, inherited]**

Reimplemented from _evaluator_base< b_interval_eval_type, expression_node, b_interval, model::walker >.

Definition at line 423 of file evaluator.h.

#### 6.12.3.21 int cached_forward_evaluator_base< b_interval_eval_type, expression_node, b_interval, model::walker >::vcollect (const return_value & *_rval*) [inline, virtual, inherited]

Reimplemented from _evaluator_base< b_interval_eval_type, expression_node, b_interval, model::walker >.

Definition at line 421 of file evaluator.h.

#### 6.12.3.22 void cached_forward_evaluator_base< b_interval_eval_type, expression_node, b_interval, model::walker >::vinit () [inline, inherited]

Definition at line 425 of file evaluator.h.

#### 6.12.3.23 return_value cached_forward_evaluator_base< b_interval_eval_type, expression_node, b_interval, model::walker >::vvalue () [inline, virtual, inherited]

Reimplemented from _evaluator_base< b_interval_eval_type, expression_node, b_interval, model::walker >.

Definition at line 424 of file evaluator.h.

### 6.12.4 Member Data Documentation

#### 6.12.4.1 b_interval_eval_type _evaluator_base< b_interval_eval_type, expression_node, b_interval, model::walker >::eval_data [protected, inherited]

Definition at line 252 of file evaluator.h.

#### 6.12.4.2 const variable_indicator* cached_evaluator_base< b_interval_eval_type, expression_node, b_interval, model::walker >::v_ind [protected, inherited]

Definition at line 295 of file evaluator.h.

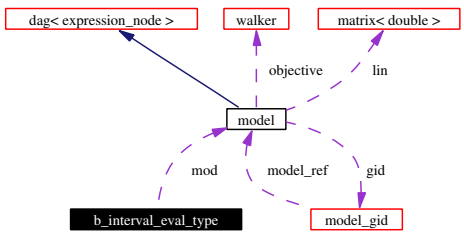
The documentation for this class was generated from the following file:

- bint_evaluator.h

## 6.13 b_interval_eval_type Struct Reference

`#include <bint_evaluator.h>`

Collaboration diagram for b_interval_eval_type:



### Public Attributes

- const std::vector< b_interval > * x
- std::vector< b_interval > * cache
- const model * mod
- void * p
- b_interval d
- int info
- b_interval r
- unsigned int n

### 6.13.1 Member Data Documentation

#### 6.13.1.1 std::vector<b_interval>* b_interval_eval_type::cache

Definition at line 47 of file bint_evaluator.h.

#### 6.13.1.2 b_interval b_interval_eval_type::d

Definition at line 50 of file bint_evaluator.h.

#### 6.13.1.3 int b_interval_eval_type::info

Definition at line 51 of file bint_evaluator.h.

#### 6.13.1.4 const model* b_interval_eval_type::mod

Definition at line 48 of file bint_evaluator.h.

#### 6.13.1.5 unsigned int b_interval_eval_type::n

Definition at line 53 of file bint_evaluator.h.

#### 6.13.1.6 void* b_interval_eval_type::p

Definition at line 49 of file bint_evaluator.h.

#### 6.13.1.7 b_interval b_interval_eval_type::r

Definition at line 52 of file bint_evaluator.h.

#### 6.13.1.8 const std::vector<b_interval>* b_interval_eval_type::x

Definition at line 46 of file bint_evaluator.h.

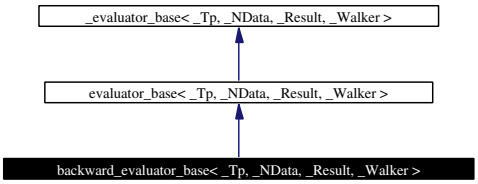
The documentation for this struct was generated from the following file:

- bint_evaluator.h

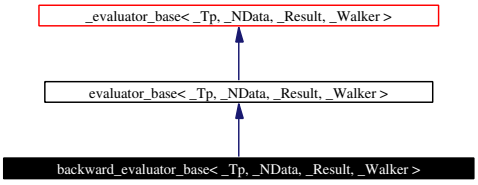
## 6.14 backward_evaluator_base< _Tp, _NData, _Result, _Walker > Class Template Reference

`#include <evaluator.h>`

Inheritance diagram for backward_evaluator_base< _Tp, _NData, _Result, _Walker >:




Collaboration diagram for backward_evaluator_base< _Tp, _NData, _Result, _Walker >:




### Public Types

- typedef _Base::node_data_type node_data_type
- typedef _Base::return_value return_value
- typedef _Base::walker walker
- typedef _Tp data_type

### Public Methods

- backward_evaluator_base ()
- backward_evaluator_base (const _Tp &__x)
- backward_evaluator_base (const _Self &__x)
- virtual ~backward_evaluator_base ()
- int preorder (const node_data_type &__data)
- void postorder (const node_data_type &__data)
- int collect (const node_data_type &__data, const return_value &__rval)
- int vcollect (const return_value &__rval)
- return_value value ()
- return_value vvalue ()
- void vinit ()
- virtual void initialize ()
- virtual void initialize (const node_data_type &__data)
- virtual int calculate (const node_data_type &__data)
- virtual void cleanup (const node_data_type &__data)
- virtual int update (const node_data_type &__data, const return_value &__rval)
- virtual int update (const return_value &__rval)
- virtual return_value calculate_value (bool eval_all)

**Protected Attributes**

- _Tp eval_data

**template<class _Tp, class _NData, class _Result, class _Walker> class backward_evaluator_base< _Tp, _NData, _Result, _Walker >**

### 6.14.1 Member Typedef Documentation

#### 6.14.1.1 template<class _Tp, class _NData, class _Result, class _Walker> typedef _Tp _evaluator_base< _Tp, _NData, _Result, _Walker >::data_type [inherited]

Definition at line 245 of file evaluator.h.

#### 6.14.1.2 template<class _Tp, class _NData, class _Result, class _Walker> typedef _Base::node_data_type backward_evaluator_base< _Tp, _NData, _Result, _Walker >::node_data_type

Reimplemented from evaluator_base< _Tp, _NData, _Result, _Walker >.

Definition at line 358 of file evaluator.h.

#### 6.14.1.3 template<class _Tp, class _NData, class _Result, class _Walker> typedef _Base::return_value backward_evaluator_base< _Tp, _NData, _Result, _Walker >::return_value

Reimplemented from evaluator_base< _Tp, _NData, _Result, _Walker >.

Definition at line 359 of file evaluator.h.

#### 6.14.1.4 template<class _Tp, class _NData, class _Result, class _Walker> typedef _Base::walker backward_evaluator_base< _Tp, _NData, _Result, _Walker >::walker

Reimplemented from evaluator_base< _Tp, _NData, _Result, _Walker >.

Definition at line 360 of file evaluator.h.

### 6.14.2 Constructor & Destructor Documentation

#### 6.14.2.1 template<class _Tp, class _NData, class _Result, class _Walker> backward_evaluator_base< _Tp, _NData, _Result, _Walker >::backward_evaluator_base () [inline]

Definition at line 363 of file evaluator.h.

#### 6.14.2.2 template<class _Tp, class _NData, class _Result, class _Walker> backward_evaluator_base< _Tp, _NData, _Result, _Walker >::backward_evaluator_base (const _Tp & __x) [inline]

Definition at line 364 of file evaluator.h.

#### 6.14.2.3 template<class _Tp, class _NData, class _Result, class _Walker> backward_evaluator_base< _Tp, _NData, _Result, _Walker >::backward_evaluator_base (const _Self & __x) [inline]

Definition at line 365 of file evaluator.h.

#### 6.14.2.4 template<class _Tp, class _NData, class _Result, class _Walker> virtual backward_evaluator_base< _Tp, _NData, _Result, _Walker >::~backward_evaluator_base () [inline, virtual]

Definition at line 366 of file evaluator.h.

### 6.14.3 Member Function Documentation

#### 6.14.3.1 template<class _Tp, class _NData, class _Result, class _Walker> virtual int backward_evaluator_base< _Tp, _NData, _Result, _Walker >::calculate (const node_data_type & *__data*) [inline, virtual]

Definition at line 383 of file evaluator.h.

#### 6.14.3.2 template<class _Tp, class _NData, class _Result, class _Walker> virtual return_value backward_evaluator_base< _Tp, _NData, _Result, _Walker >::calculate_value (bool *eval_all*) [inline, virtual]

Definition at line 388 of file evaluator.h.

#### 6.14.3.3 template<class _Tp, class _NData, class _Result, class _Walker> virtual void backward_evaluator_base< _Tp, _NData, _Result, _Walker >::cleanup (const node_data_type & *__data*) [inline, virtual]

Definition at line 384 of file evaluator.h.

#### 6.14.3.4 template<class _Tp, class _NData, class _Result, class _Walker> int backward_evaluator_base< _Tp, _NData, _Result, _Walker >::collect (const node_data_type & *__data*, const return_value & *__rval*) [inline, virtual]

Reimplemented from _evaluator_base< _Tp, _NData, _Result, _Walker >.

Definition at line 372 of file evaluator.h.

#### 6.14.3.5 template<class _Tp, class _NData, class _Result, class _Walker> virtual void backward_evaluator_base< _Tp, _NData, _Result, _Walker >::initialize (const node_data_type & *__data*) [inline, virtual]

Definition at line 382 of file evaluator.h.

#### 6.14.3.6 template<class _Tp, class _NData, class _Result, class _Walker> virtual void backward_evaluator_base< _Tp, _NData, _Result, _Walker >::initialize () [inline, virtual]

Definition at line 381 of file evaluator.h.

#### 6.14.3.7 template<class _Tp, class _NData, class _Result, class _Walker> void backward_evaluator_base< _Tp, _NData, _Result, _Walker >::postorder (const node_data_type & *__data*) [inline, virtual]

Reimplemented from _evaluator_base< _Tp, _NData, _Result, _Walker >.

Definition at line 371 of file evaluator.h.

#### 6.14.3.8 template<class _Tp, class _NData, class _Result, class _Walker> int backward_evaluator_base< _Tp, _NData, _Result, _Walker >::preorder (const node_data_type & __data) [inline, virtual]

Reimplemented from evaluator_base< _Tp, _NData, _Result, _Walker >.

Definition at line 369 of file evaluator.h.

#### 6.14.3.9 template<class _Tp, class _NData, class _Result, class _Walker> virtual int backward_evaluator_base< _Tp, _NData, _Result, _Walker >::update (const return_value & __rval) [inline, virtual]

Definition at line 387 of file evaluator.h.

#### 6.14.3.10 template<class _Tp, class _NData, class _Result, class _Walker> virtual int backward_evaluator_base< _Tp, _NData, _Result, _Walker >::update (const node_data_type & __data, const return_value & __rval) [inline, virtual]

Definition at line 385 of file evaluator.h.

#### 6.14.3.11 template<class _Tp, class _NData, class _Result, class _Walker> return_value backward_evaluator_base< _Tp, _NData, _Result, _Walker >::value () [inline, virtual]

Reimplemented from _evaluator_base< _Tp, _NData, _Result, _Walker >.

Definition at line 376 of file evaluator.h.

#### 6.14.3.12 template<class _Tp, class _NData, class _Result, class _Walker> int backward_evaluator_base< _Tp, _NData, _Result, _Walker >::vcollect (const return_value & __rval) [inline, virtual]

Reimplemented from _evaluator_base< _Tp, _NData, _Result, _Walker >.

Definition at line 374 of file evaluator.h.

#### 6.14.3.13 template<class _Tp, class _NData, class _Result, class _Walker> void backward_evaluator_base< _Tp, _NData, _Result, _Walker >::vinit () [inline]

Definition at line 378 of file evaluator.h.

#### 6.14.3.14 template<class _Tp, class _NData, class _Result, class _Walker> return_value backward_evaluator_base< _Tp, _NData, _Result, _Walker >::vvalue () [inline, virtual]

Reimplemented from _evaluator_base< _Tp, _NData, _Result, _Walker >.

Definition at line 377 of file evaluator.h.

### 6.14.4 Member Data Documentation

#### 6.14.4.1 template<class _Tp, class _NData, class _Result, class _Walker> _Tp _evaluator_base< _Tp, _NData, _Result, _Walker >::eval_data [protected, inherited]

Definition at line 252 of file evaluator.h.

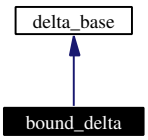
The documentation for this class was generated from the following file:

- evaluator.h

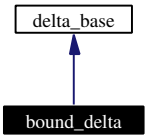
## 6.15 bound_delta Class Reference

`#include <bound_delta.h>`

Inheritance diagram for bound_delta:




Collaboration diagram for bound_delta:




### Public Methods

- bound_delta (const std::vector< unsigned int > &__i, const std::vector< interval > &__b)
- bound_delta (unsigned int __i, interval __b)
- bound_delta (const bound_delta &__d)
- bound_delta * new_copy () const
- void destroy_copy (bound_delta *__d)
- bool apply (work_node &_x, undelta_base *&_u) const
- virtual void destroy_copy (delta_base *__d)
- delta make_delta (const std::string &a)
- const std::string & get_action () const
- virtual void convert (work_node &_x, delta_base *&_d)
- virtual bool apply3 (work_node &_x, const work_node &_y, undelta_base *&_u) const

### Public Attributes

- std::vector< unsigned int > indices
- std::vector< interval > new_f_bounds

### Protected Attributes

- std::string _action

### 6.15.1 Constructor & Destructor Documentation

#### 6.15.1.1 bound_delta::bound_delta (const std::vector< unsigned int > & *_i*, const std::vector< interval > & *_b*) [inline]

Definition at line 79 of file bound_delta.h.

#### 6.15.1.2 bound_delta::bound_delta (unsigned int *_i*, interval *_b*) [inline]

Definition at line 84 of file bound_delta.h.

#### 6.15.1.3 bound_delta::bound_delta (const bound_delta & *_d*) [inline]

Definition at line 88 of file bound_delta.h.

### 6.15.2 Member Function Documentation

#### 6.15.2.1 bool bound_delta::apply (work_node & *_x*, undelta_base *& *_u*) const [virtual]

Reimplemented from delta_base.

Definition at line 29 of file bound_delta.cc.

#### 6.15.2.2 bool delta_base::apply3 (work_node & *_x*, const work_node & *_y*, undelta_base *& *_u*) const [inline, virtual, inherited]

Definition at line 63 of file api_delta.h.

#### 6.15.2.3 virtual void delta_base::convert (work_node & *_x*, delta_base *& *_d*) [inline, virtual, inherited]

Reimplemented in table_delta.

Definition at line 107 of file api_deltabase.h.

#### 6.15.2.4 virtual void delta_base::destroy_copy (delta_base * *_d*) [inline, virtual, inherited]

Definition at line 95 of file api_deltabase.h.

#### 6.15.2.5 void bound_delta::destroy_copy (bound_delta * *_d*) [inline]

Definition at line 97 of file bound_delta.h.

#### 6.15.2.6 const std::string& delta_base::get_action () const [inline, inherited]

Definition at line 105 of file api_deltabase.h.

#### 6.15.2.7 delta delta_base::make_delta (const std::string & *a*) [inline, inherited]

Definition at line 99 of file api_deltabase.h.

#### 6.15.2.8 bound_delta* bound_delta::new_copy () const [inline, virtual]

Reimplemented from delta_base.

Definition at line 96 of file bound_delta.h.

### 6.15.3 Member Data Documentation

#### 6.15.3.1 std::string delta_base::_action [protected, inherited]

Definition at line 86 of file api_deltabase.h.

#### 6.15.3.2 std::vector<unsigned int> bound_delta::indices

Definition at line 75 of file bound_delta.h.

#### 6.15.3.3 std::vector<interval> bound_delta::new_f_bounds

Definition at line 77 of file bound_delta.h.

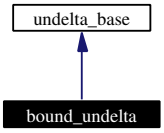
The documentation for this class was generated from the following files:

- bound_delta.h
- bound_delta.cc

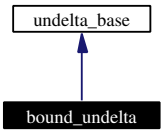
## 6.16 bound_undelta Class Reference

`#include <bound_delta.h>`

Inheritance diagram for bound_undelta:




Collaboration diagram for bound_undelta:




### Public Methods

- bound_undelta (const std::vector< unsigned int > &__i, const std::vector< interval > &__b, double _og, double _olv)
- bound_undelta (const std::vector< unsigned int > &__i, double _og, double _olv)
- bound_undelta (const bound_undelta &__d)

- bound_undelta * new_copy () const
- void destroy_copy (bound_undelta *__d)
- bool unapply (work_node &_x) const
- virtual void destroy_copy (undelta_base *__d)
- undelta make_undelta ()
- virtual bool unapply3 (work_node &_x, const work_node &_y) const

**Public Attributes**

- std::vector< unsigned int > indices
- std::vector< interval > old_f_bounds
- double old_gain
- double old_log_vol

**Friends**

- class bound_delta

### 6.16.1 Constructor & Destructor Documentation

#### 6.16.1.1 bound_undelta::bound_undelta (const std::vector< unsigned int > & *__i*, const std::vector< interval > & *__b*, double *_og*, double *_olv*) [inline]

Definition at line 42 of file bound_delta.h.

#### 6.16.1.2 bound_undelta::bound_undelta (const std::vector< unsigned int > & *__i*, double *_og*, double *_olv*) [inline]

Definition at line 48 of file bound_delta.h.

#### 6.16.1.3 bound_undelta::bound_undelta (const bound_undelta & *__d*) [inline]

Definition at line 53 of file bound_delta.h.

### 6.16.2 Member Function Documentation

#### 6.16.2.1 virtual void undelta_base::destroy_copy (undelta_base * *__d*) [inline, virtual, inherited]

Definition at line 146 of file api_deltabase.h.

#### 6.16.2.2 void bound_undelta::destroy_copy (bound_undelta * *__d*) [inline]

Definition at line 65 of file bound_delta.h.

#### 6.16.2.3 undelta undelta_base::make_undelta () [inline, inherited]

Definition at line 150 of file api_deltabase.h.

#### 6.16.2.4 bound_undelta* bound_undelta::new_copy () const [inline, virtual]

Reimplemented from undelta_base.

Definition at line 64 of file bound_delta.h.

#### 6.16.2.5 bool bound_undelta::unapply (work_node & _x) const [virtual]

Reimplemented from undelta_base.

Definition at line 66 of file bound_delta.cc.

#### 6.16.2.6 bool undelta_base::unapply3 (work_node & _x, const work_node & _y) const [inline, virtual, inherited]

Definition at line 69 of file api_delta.h.

### 6.16.3 Friends And Related Function Documentation

#### 6.16.3.1 friend class bound_delta [friend]

Definition at line 69 of file bound_delta.h.

### 6.16.4 Member Data Documentation

#### 6.16.4.1 std::vector<unsigned int> bound_undelta::indices

Definition at line 35 of file bound_delta.h.

#### 6.16.4.2 std::vector<interval> bound_undelta::old_f_bounds

Definition at line 37 of file bound_delta.h.

#### 6.16.4.3 double bound_undelta::old_gain

Definition at line 40 of file bound_delta.h.

#### 6.16.4.4 double bound_undelta::old_log_vol

Definition at line 40 of file bound_delta.h.

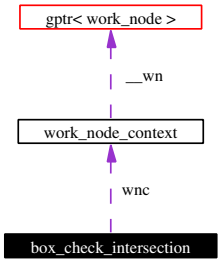
The documentation for this class was generated from the following files:

- bound_delta.h
- bound_delta.cc

## 6.17 box_check_intersection Class Reference

`#include <dbtools.h>`

Collaboration diagram for box_check_intersection:




### Public Types

- typedef work_node_context context
- typedef bool return_type

### Public Methods

- box_check_intersection (**vdbl::colid** _x)
- box_check_intersection (const box_check_intersection &_i)
- virtual ~box_check_intersection ()
- bool operator() () const
- bool def () const
- void setcontext (const context *c, const vdbl::row *r)

#### 6.17.1 Member Typedef Documentation

##### 6.17.1.1 typedef work_node_context box_check_intersection::context

Definition at line 80 of file dbtools.h.

##### 6.17.1.2 typedef bool box_check_intersection::return_type

Definition at line 89 of file dbtools.h.

#### 6.17.2 Constructor & Destructor Documentation

##### 6.17.2.1 box_check_intersection::box_check_intersection (vdbl::colid _x) [inline]

Definition at line 91 of file dbtools.h.

##### 6.17.2.2 box_check_intersection::box_check_intersection (const box_check_intersection & _i) [inline]

Definition at line 93 of file dbtools.h.

##### 6.17.2.3 virtual box_check_intersection::~box_check_intersection () [inline, virtual]

Definition at line 95 of file dbtools.h.

### 6.17.3 Member Function Documentation

#### 6.17.3.1 bool box_check_intersection::def () const [inline]

Definition at line 98 of file dbtools.h.

#### 6.17.3.2 bool box_check_intersection::operator() ()

Definition at line 66 of file dbtools.cc.

#### 6.17.3.3 void box_check_intersection::setcontext (const context * *c*, const vdbl::row * *r*) [inline]

Definition at line 99 of file dbtools.h.

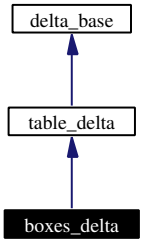
The documentation for this class was generated from the following files:

- dbtools.h
- dbtools.cc

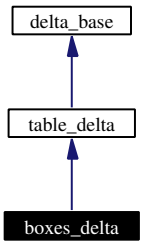
## 6.18 boxes_delta Class Reference

`#include <boxes_delta.h>`

Inheritance diagram for boxes_delta:




Collaboration diagram for boxes_delta:




### Public Types

- typedef std::pair< std::string, dbt_row > t_line
- typedef std::vector< t_line > t_ctr

### Public Methods

- boxes_delta (bool _add=true)
- boxes_delta (const dbt_row &_b, bool _add=true)
- boxes_delta (const std::vector< dbt_row > &_b, bool _add=true)
- boxes_delta (const boxes_delta &__d)
- boxes_delta * new_copy () const
- void destroy_copy (boxes_delta *__d)
- void create_table (work_node &_x, vdbl::standard_table *&ptb, const std::string &__t) const
- bool apply (work_node &_x, undelta_base *&_u) const
- void add (const t_line &_tl)
- void add (const std::string &_tn, const dbt_row &_r)
- void add (const std::vector< t_line > &_tlv)
- void rm (const annotation &_tr)
- void rm (const std::vector< annotation > &_trv)
- virtual void destroy_copy (table_delta *__d)
- virtual void destroy_copy (delta_base *__d)
- void convert (work_node &_x, delta_base *&_u)
- delta make_delta (const std::string &a)
- const std::string & get_action () const
- virtual bool apply3 (work_node &_x, const work_node &_y, undelta_base *&_u) const

### Protected Attributes

- std::string _action

### 6.18.1 Member Typedef Documentation

#### 6.18.1.1 typedef std::vector<t_line> table_delta::t_ctr [inherited]

Definition at line 36 of file table_delta.h.

#### 6.18.1.2 typedef std::pair<std::string,dbt_row> table_delta::t_line [inherited]

Definition at line 35 of file table_delta.h.

### 6.18.2 Constructor & Destructor Documentation

#### 6.18.2.1 boxes_delta::boxes_delta (bool *_add* = true) [inline]

Definition at line 39 of file boxes_delta.h.

#### 6.18.2.2 boxes_delta::boxes_delta (const dbt_row & *_b*, bool *_add* = true) [inline]

Definition at line 41 of file boxes_delta.h.

#### 6.18.2.3 boxes_delta::boxes_delta (const std::vector< dbt_row > & *_b*, bool *_add* = true) [inline]

Definition at line 45 of file boxes_delta.h.

#### 6.18.2.4 boxes_delta::boxes_delta (const boxes_delta & *__d*) [inline]

Definition at line 49 of file boxes_delta.h.

### 6.18.3 Member Function Documentation

#### 6.18.3.1 void table_delta::add (const std::vector< t_line > & *_tlv*) [inline, inherited]

Definition at line 73 of file table_delta.h.

#### 6.18.3.2 void table_delta::add (const std::string & *_tn*, const dbt_row & *_r*) [inline, inherited]

Definition at line 71 of file table_delta.h.

#### 6.18.3.3 void table_delta::add (const t_line & *_tl*) [inline, inherited]

Definition at line 70 of file table_delta.h.

#### 6.18.3.4 bool boxes_delta::apply (work_node & *_x*, undelta_base *& *_u*) const [virtual]

Reimplemented from table_delta.

Definition at line 30 of file boxes_delta.cc.

#### 6.18.3.5 bool delta_base::apply3 (work_node & *_x*, const work_node & *_y*, undelta_base *& *_u*) const [inline, virtual, inherited]

Definition at line 63 of file api_delta.h.

#### 6.18.3.6 void table_delta::convert (work_node & *_x*, delta_base *& *_u*) [virtual, inherited]

Reimplemented from delta_base.

Definition at line 37 of file table_delta.cc.

#### 6.18.3.7 void boxes_delta::create_table (work_node & *_x*, vdbl::standard_table *& *ptb*, const std::string & *__t*) const [virtual]

Reimplemented from table_delta.

Definition at line 38 of file boxes_delta.cc.

#### 6.18.3.8 virtual void delta_base::destroy_copy (delta_base * *__d*) [inline, virtual, inherited]

Definition at line 95 of file api_deltabase.h.

#### 6.18.3.9 virtual void table_delta::destroy_copy (table_delta * *__d*) [inline, virtual, inherited]

Definition at line 81 of file table_delta.h.

#### 6.18.3.10 void boxes_delta::destroy_copy (boxes_delta * *_d*) [inline]

Definition at line 53 of file boxes_delta.h.

#### 6.18.3.11 const std::string& delta_base::get_action () const [inline, inherited]

Definition at line 105 of file api_deltabase.h.

#### 6.18.3.12 delta delta_base::make_delta (const std::string & *a*) [inline, inherited]

Definition at line 99 of file api_deltabase.h.

#### 6.18.3.13 boxes_delta* boxes_delta::new_copy () const [inline, virtual]

Reimplemented from table_delta.

Definition at line 51 of file boxes_delta.h.

#### 6.18.3.14 void table_delta::rm (const std::vector< annotation > & *_trv*) [inline, inherited]

Definition at line 76 of file table_delta.h.

#### 6.18.3.15 void table_delta::rm (const annotation & *_tr*) [inline, inherited]

Definition at line 75 of file table_delta.h.

### 6.18.4 Member Data Documentation

#### 6.18.4.1 std::string delta_base::_action [protected, inherited]

Definition at line 86 of file api_deltabase.h.

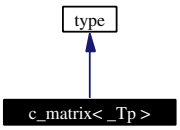
The documentation for this class was generated from the following files:

- boxes_delta.h
- boxes_delta.cc

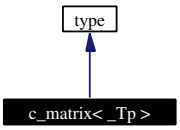
## 6.19 c_matrix< _Tp > Class Template Reference

`#include <linalg.h>`

Inheritance diagram for c_matrix< _Tp >:




Collaboration diagram for c_matrix< _Tp >:




**Public Types**

- typedef _Base::OneD Col

**Public Methods**

- c_matrix ()
- c_matrix (const _Self &__m)
- c_matrix (const size_t &n, const size_t &m)

**template<class _Tp> class c_matrix< _Tp >**

### 6.19.1 Member Typedef Documentation

#### 6.19.1.1 template<class _Tp> typedef _Base::OneD c_matrix< _Tp >::Col

Definition at line 31 of file linalg.h.

### 6.19.2 Constructor & Destructor Documentation

#### 6.19.2.1 template<class _Tp> c_matrix< _Tp >::c_matrix () [inline]

Definition at line 33 of file linalg.h.

#### 6.19.2.2 template<class _Tp> c_matrix< _Tp >::c_matrix (const _Self & *__m*) [inline]

Definition at line 34 of file linalg.h.

#### 6.19.2.3 template<class _Tp> c_matrix< _Tp >::c_matrix (const size_t & *n*, const size_t & *m*) [inline]

Definition at line 35 of file linalg.h.

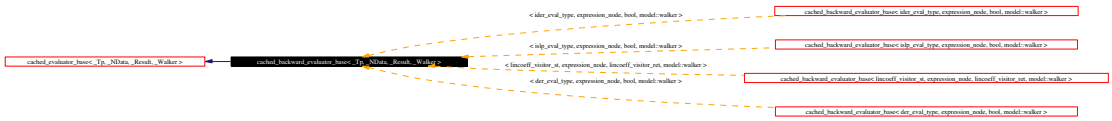
The documentation for this class was generated from the following file:

- linalg.h

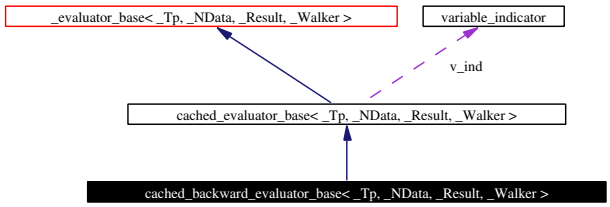
## 6.20 cached_backward_evaluator_base< _Tp, _NData, _Result, _Walker > Class Template Reference

`#include <evaluator.h>`

Inheritance diagram for cached_backward_evaluator_base< _Tp, _NData, _Result, _Walker >:



Collaboration diagram for cached_backward_evaluator_base< _Tp, _NData, _Result, _Walker >:




### Public Types

- typedef cached_evaluator_base< _Tp, _NData, _Result, _Walker >::node_data_type node_data_type
- typedef cached_evaluator_base< _Tp, _NData, _Result, _Walker >::return_value return_value
- typedef cached_evaluator_base< _Tp, _NData, _Result, _Walker >::walker walker
- typedef _Tp data_type

### Public Methods

- virtual bool is_cached (const node_data_type &__data)
- int preorder (const node_data_type &__data)
- void postorder (const node_data_type &__data)
- int collect (const node_data_type &__data, const return_value &__rval)
- int vcollect (const return_value &__rval)
- void vinit ()
- return_value value ()
- return_value vvalue ()
- virtual void initialize ()
- virtual int calculate (const node_data_type &__data)
- virtual void cleanup (const node_data_type &__data)
- virtual void retrieve_from_cache (const node_data_type &__data)
- virtual int update (const node_data_type &__data, const return_value &__rval)
- virtual int update (const return_value &__rval)
- virtual return_value calculate_value (bool eval_all)
- virtual walker short_cut_to (const node_data_type &__data) PURE_VIRTUAL public

### Protected Attributes

- const variable_indicator * v_ind
- _Tp eval_data

**template<class _Tp, class _NData, class _Result, class _Walker> class cached_backward_evaluator_base< _Tp, _NData, _Result, _Walker >**

### 6.20.1 Member Typedef Documentation

#### 6.20.1.1 template<class _Tp, class _NData, class _Result, class _Walker> typedef _Tp _evaluator_base< _Tp, _NData, _Result, _Walker >::data_type [inherited]

Definition at line 245 of file evaluator.h.

#### 6.20.1.2 template<class _Tp, class _NData, class _Result, class _Walker> typedef cached_evaluator_base<_Tp, _NData, _Result,_Walker>::node_data_type cached_backward_evaluator_base< _Tp, _NData, _Result, _Walker >::node_data_type

Reimplemented from cached_evaluator_base< _Tp, _NData, _Result, _Walker >.

Definition at line 444 of file evaluator.h.

#### 6.20.1.3 template<class _Tp, class _NData, class _Result, class _Walker> typedef cached_evaluator_base<_Tp, _NData, _Result,_Walker>::return_value cached_backward_evaluator_base< _Tp, _NData, _Result, _Walker >::return_value

Reimplemented from cached_evaluator_base< _Tp, _NData, _Result, _Walker >.

Definition at line 446 of file evaluator.h.

#### 6.20.1.4 template<class _Tp, class _NData, class _Result, class _Walker> typedef cached_evaluator_base<_Tp, _NData, _Result,_Walker>::walker cached_backward_evaluator_base< _Tp, _NData, _Result, _Walker >::walker

Reimplemented from cached_evaluator_base< _Tp, _NData, _Result, _Walker >.

Definition at line 448 of file evaluator.h.

### 6.20.2 Member Function Documentation

#### 6.20.2.1 template<class _Tp, class _NData, class _Result, class _Walker> virtual int cached_backward_evaluator_base< _Tp, _NData, _Result, _Walker >::calculate (const node_data_type & *__data*) [inline, virtual]

Definition at line 476 of file evaluator.h.

#### 6.20.2.2 template<class _Tp, class _NData, class _Result, class _Walker> virtual return_value cached_backward_evaluator_base< _Tp, _NData, _Result, _Walker >::calculate_value (bool *eval_all*) [inline, virtual]

Reimplemented in der_eval, der_eval, ider_eval, islp_eval, and lincoeff_visitor.

Definition at line 482 of file evaluator.h.

#### 6.20.2.3 template<class _Tp, class _NData, class _Result, class _Walker> virtual void cached_backward_evaluator_base< _Tp, _NData, _Result, _Walker >::cleanup (const node_data_type & *__data*) [inline, virtual]

Definition at line 477 of file evaluator.h.

#### 6.20.2.4 template<class _Tp, class _NData, class _Result, class _Walker> int cached_backward_evaluator_base< _Tp, _NData, _Result, _Walker >::collect (const node_data_type & *__data*, const return_value & *__rval*) [inline, virtual]

Reimplemented from _evaluator_base< _Tp, _NData, _Result, _Walker >.

Definition at line 466 of file evaluator.h.

#### 6.20.2.5 template<class _Tp, class _NData, class _Result, class _Walker> virtual void cached_backward_evaluator_base< _Tp, _NData, _Result, _Walker >::initialize () [inline, virtual]

Reimplemented in der_eval, der_eval, ider_eval, islp_eval, and lincoeff_visitor.

Definition at line 475 of file evaluator.h.

#### 6.20.2.6 template<class _Tp, class _NData, class _Result, class _Walker> virtual bool cached_backward_evaluator_base< _Tp, _NData, _Result, _Walker >::is_cached (const node_data_type & *__data*) [inline, virtual]

Reimplemented in der_eval, der_eval, ider_eval, and islp_eval.

Definition at line 453 of file evaluator.h.

#### 6.20.2.7 template<class _Tp, class _NData, class _Result, class _Walker> void cached_backward_evaluator_base< _Tp, _NData, _Result, _Walker >::postorder (const node_data_type & *__data*) [inline, virtual]

Reimplemented from _evaluator_base< _Tp, _NData, _Result, _Walker >.

Definition at line 465 of file evaluator.h.

#### 6.20.2.8 template<class _Tp, class _NData, class _Result, class _Walker> int cached_backward_evaluator_base< _Tp, _NData, _Result, _Walker >::preorder (const node_data_type & *__data*) [inline, virtual]

Reimplemented from cached_evaluator_base< _Tp, _NData, _Result, _Walker >.

Definition at line 456 of file evaluator.h.

#### 6.20.2.9 template<class _Tp, class _NData, class _Result, class _Walker> virtual void cached_backward_evaluator_base< _Tp, _NData, _Result, _Walker >::retrieve_from_cache (const node_data_type & *__data*) [inline, virtual]

Definition at line 478 of file evaluator.h.

#### 6.20.2.10 template<class _Tp, class _NData, class _Result, class _Walker> virtual walker cached_evaluator_base< _Tp, _NData, _Result, _Walker >::short_cut_to (const node_data_type & *__data*) [inline, virtual, inherited]

Definition at line 303 of file evaluator.h.

#### 6.20.2.11 template<class _Tp, class _NData, class _Result, class _Walker> virtual int cached_backward_evaluator_base< _Tp, _NData, _Result, _Walker >::update (const return_value & *__rval*) [inline, virtual]

Definition at line 481 of file evaluator.h.

#### 6.20.2.12 template<class _Tp, class _NData, class _Result, class _Walker> virtual int cached_backward_evaluator_base< _Tp, _NData, _Result, _Walker >::update (const node_data_type & __data, const return_value & __rval) [inline, virtual]

Definition at line 479 of file evaluator.h.

#### 6.20.2.13 template<class _Tp, class _NData, class _Result, class _Walker> return_value cached_backward_evaluator_base< _Tp, _NData, _Result, _Walker >::value () [inline, virtual]

Reimplemented from _evaluator_base< _Tp, _NData, _Result, _Walker >.

Definition at line 471 of file evaluator.h.

#### 6.20.2.14 template<class _Tp, class _NData, class _Result, class _Walker> int cached_backward_evaluator_base< _Tp, _NData, _Result, _Walker >::vcollect (const return_value & __rval) [inline, virtual]

Reimplemented from _evaluator_base< _Tp, _NData, _Result, _Walker >.

Definition at line 468 of file evaluator.h.

#### 6.20.2.15 template<class _Tp, class _NData, class _Result, class _Walker> void cached_backward_evaluator_base< _Tp, _NData, _Result, _Walker >::vinit () [inline]

Definition at line 470 of file evaluator.h.

#### 6.20.2.16 template<class _Tp, class _NData, class _Result, class _Walker> return_value cached_backward_evaluator_base< _Tp, _NData, _Result, _Walker >::vvalue () [inline, virtual]

Reimplemented from _evaluator_base< _Tp, _NData, _Result, _Walker >.

Definition at line 472 of file evaluator.h.

### 6.20.3 Member Data Documentation

#### 6.20.3.1 template<class _Tp, class _NData, class _Result, class _Walker> _Tp _evaluator_base< _Tp, _NData, _Result, _Walker >::eval_data [protected, inherited]

Definition at line 252 of file evaluator.h.

#### 6.20.3.2 template<class _Tp, class _NData, class _Result, class _Walker> const variable_indicator* cached_evaluator_base< _Tp, _NData, _Result, _Walker >::v_ind [protected, inherited]

Definition at line 295 of file evaluator.h.

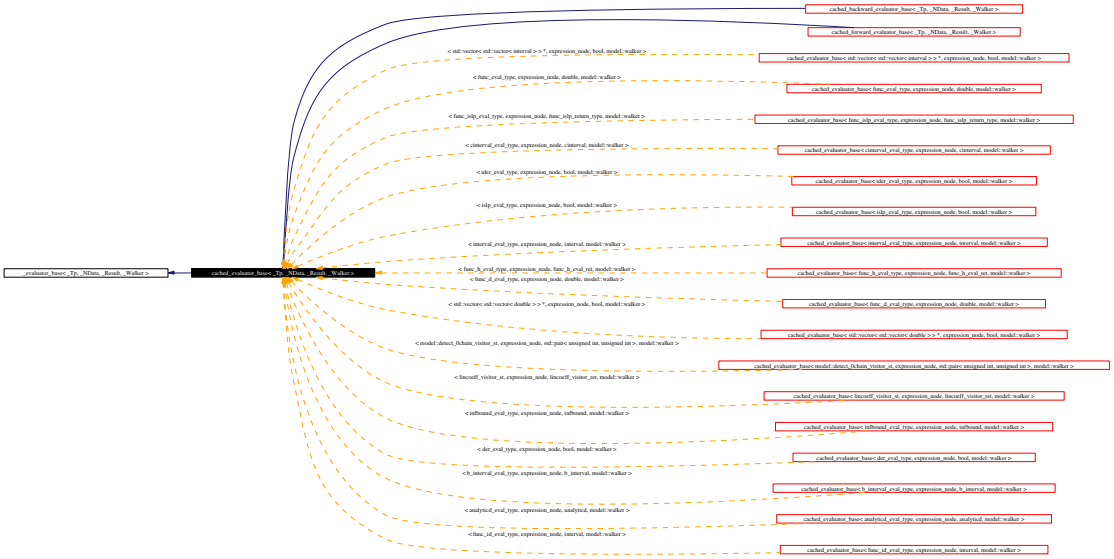
The documentation for this class was generated from the following file:

- evaluator.h

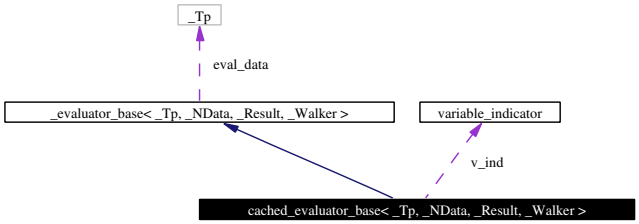
## 6.21 cached_evaluator_base< _Tp, _NData, _Result, _Walker > Class Template Reference

`#include <evaluator.h>`

Inheritance diagram for cached_evaluator_base< _Tp, _NData, _Result, _Walker >:



Collaboration diagram for cached_evaluator_base< _Tp, _NData, _Result, _Walker >:




### Public Types

- typedef _Base::node_data_type node_data_type
- typedef _Base::return_value return_value
- typedef _Base::walker walker
- typedef _Tp data_type

### Public Methods

- virtual int preorder (const node_data_type &__data)
- virtual walker short_cut_to (const node_data_type &__data) PURE_VIRTUAL public
- cached_evaluator_base (const _Tp &__x, const variable_indicator &__v)
- cached_evaluator_base (const _Self &__x)
- virtual ~cached_evaluator_base ()

- virtual return_value vvalue ()
- virtual return_value value ()
- virtual int vcollect (const return_value &__cresult)
- virtual int collect (const node_data_type &__data, const return_value &__cresult)
- virtual void postorder (const node_data_type &__data)

**Protected Attributes**

- const variable_indicator * v_ind
- _Tp eval_data

**template<class _Tp, class _NData, class _Result, class _Walker> class cached_evaluator_base< _Tp, _NData, _Result, _Walker >**

### 6.21.1 Member Typedef Documentation

#### 6.21.1.1 template<class _Tp, class _NData, class _Result, class _Walker> typedef _Tp _evaluator_base< _Tp, _NData, _Result, _Walker >::data_type [inherited]

Definition at line 245 of file evaluator.h.

#### 6.21.1.2 template<class _Tp, class _NData, class _Result, class _Walker> typedef _Base::node_data_type cached_evaluator_base< _Tp, _NData, _Result, _Walker >::node_data_type

Reimplemented from _evaluator_base< _Tp, _NData, _Result, _Walker >.

Reimplemented in cached_forward_evaluator_base< _Tp, _NData, _Result, _Walker >, cached_backward_evaluator_base< _Tp, _NData, _Result, _Walker >, cached_forward_evaluator_base< std::vector< std::vector< interval > > *, expression_node, bool, model::walker >, cached_forward_evaluator_base< func_eval_type, expression_node, double, model::walker >, cached_forward_evaluator_base< func_islp_eval_type, expression_node, func_islp_return_type, model::walker >, cached_forward_evaluator_base< cinterval_eval_type, expression_node, cinterval, model::walker >, cached_forward_evaluator_base< interval_eval_type, expression_node, interval, model::walker >, cached_forward_evaluator_base< func_h_eval_type, expression_node, func_h_eval_ret, model::walker >, cached_forward_evaluator_base< func_d_eval_type, expression_node, double, model::walker >, cached_forward_evaluator_base< std::vector< std::vector< double > > *, expression_node, bool, model::walker >, cached_forward_evaluator_base< model::detect_0chain_visitor_st, expression_node, std::pair< unsigned int, unsigned int >, model::walker >, cached_forward_evaluator_base< infbound_eval_type, expression_node, infbound, model::walker >, cached_forward_evaluator_base< b_interval_eval_type, expression_node, b_interval, model::walker >, cached_forward_evaluator_base< analyticd_eval_type, expression_node, analyticd, model::walker >, cached_forward_evaluator_base< func_id_eval_type, expression_node, interval, model::walker >, cached_backward_evaluator_base< ider_eval_type, expression_node, bool, model::walker >, cached_backward_evaluator_base< islp_eval_type, expression_node, bool, model::walker >, cached_backward_evaluator_base< lincoeff_visitor_st, expression_node, lincoeff_visitor_ret, model::walker >, and cached_backward_evaluator_base< der_eval_type, expression_node, bool, model::walker >.

Definition at line 298 of file evaluator.h.

#### 6.21.1.3 template<class _Tp, class _NData, class _Result, class _Walker> typedef _Base::return_value cached_evaluator_base< _Tp, _NData, _Result, _Walker >::return_value

Reimplemented from _evaluator_base< _Tp, _NData, _Result, _Walker >.

Reimplemented in cached_forward_evaluator_base< _Tp, _NData, _Result, _Walker >, cached_backward_evaluator_base< _Tp, _NData, _Result, _Walker >, cached_forward_evaluator_base< std::vector< std::vector< interval > > *, expression_node, bool, model::walker >, cached_forward_evaluator_base< func_eval_type, expression_node, double, model::walker >, cached_forward_evaluator_base< func_islp_eval_type, expression_node, func_islp_return_type, model::walker >, cached_forward_evaluator_base< cinterval_eval_type, expression_node, cinterval, model::walker >, cached_forward_evaluator_base< interval_eval_type, expression_node, interval, model::walker >, cached_forward_evaluator_base< func_h_eval_type, expression_node, func_h_eval_ret, model::walker >, cached_forward_evaluator_base< func_d_eval_type, expression_node, double, model::walker >, cached_forward_evaluator_base< std::vector< std::vector< double > > *, expression_node, bool, model::walker >, cached_forward_evaluator_base< model::detect_0chain_visitor_st, expression_node, std::pair< unsigned int, unsigned int >, model::walker >, cached_forward_evaluator_base< infbound_eval_type, expression_node, infbound, model::walker >, cached_forward_evaluator_base< b_interval_eval_type, expression_node, b_interval, model::walker >, cached_forward_evaluator_base< analyticd_eval_type, expression_node, analyticd, model::walker >, cached_forward_evaluator_base< func_id_eval_type, expression_node, interval, model::walker >, cached_backward_evaluator_base< ider_eval_type, expression_node, bool, model::walker >, cached_backward_evaluator_base< islp_eval_type, expression_node, bool, model::walker >, cached_backward_evaluator_base< lincoeff_visitor_st, expression_node, lincoeff_visitor_ret, model::walker >, and cached_backward_evaluator_base< der_eval_type, expression_node, bool, model::walker >.

Definition at line 299 of file evaluator.h.

#### 6.21.1.4 template<class _Tp, class _NData, class _Result, class _Walker> typedef _Base::walker cached_evaluator_base< _Tp, _NData, _Result, _Walker >::walker

Reimplemented from _evaluator_base< _Tp, _NData, _Result, _Walker >.

Reimplemented in cached_forward_evaluator_base< _Tp, _NData, _Result, _Walker >, cached_backward_evaluator_base< _Tp, _NData, _Result, _Walker >, cached_forward_evaluator_base< std::vector< std::vector< interval > > *, expression_node, bool, model::walker >, cached_forward_evaluator_base< func_eval_type, expression_node, double, model::walker >, cached_forward_evaluator_base< func_islp_eval_type, expression_node, func_islp_return_type, model::walker >, cached_forward_evaluator_base< cinterval_eval_type, expression_node, cinterval, model::walker >, cached_forward_evaluator_base< interval_eval_type, expression_node, interval, model::walker >, cached_forward_evaluator_base< func_h_eval_type, expression_node, func_h_eval_ret, model::walker >, cached_forward_evaluator_base< func_d_eval_type, expression_node, double, model::walker >, cached_forward_evaluator_base< std::vector< std::vector< double > > *, expression_node, bool, model::walker >, cached_forward_evaluator_base< model::detect_0chain_visitor_st, expression_node, std::pair< unsigned int, unsigned int >, model::walker >, cached_forward_evaluator_base< infbound_eval_type, expression_node, infbound, model::walker >, cached_forward_evaluator_base< b_interval_eval_type, expression_node, b_interval, model::walker >, cached_forward_evaluator_base< analyticd_eval_type, expression_node, analyticd, model::walker >, cached_forward_evaluator_base< func_id_eval_type, expression_node, interval, model::walker >, cached_backward_evaluator_base< ider_eval_type, expression_node, bool, model::walker >, cached_backward_evaluator_base< islp_eval_type, expression_node, bool, model::walker >, cached_backward_evaluator_base< lincoeff_visitor_st, expression_node, lincoeff_visitor_ret, model::walker >, and cached_backward_evaluator_base< der_eval_type, expression_node, bool, model::walker >.

Definition at line 300 of file evaluator.h.

### 6.21.2 Constructor & Destructor Documentation

#### 6.21.2.1 template<class _Tp, class _NData, class _Result, class _Walker> cached_evaluator_base< _Tp, _NData, _Result, _Walker >::cached_evaluator_base (const _Tp & *__x*, const variable_indicator & *__v*) [inline]

Definition at line 307 of file evaluator.h.

#### 6.21.2.2 template<class _Tp, class _NData, class _Result, class _Walker> cached_evaluator_base< _Tp, _NData, _Result, _Walker >::cached_evaluator_base (const _Self & __x) [inline]

Definition at line 309 of file evaluator.h.

#### 6.21.2.3 template<class _Tp, class _NData, class _Result, class _Walker> virtual cached_evaluator_base< _Tp, _NData, _Result, _Walker >::~cached_evaluator_base () [inline, virtual]

Definition at line 311 of file evaluator.h.

### 6.21.3 Member Function Documentation

#### 6.21.3.1 template<class _Tp, class _NData, class _Result, class _Walker> virtual int _evaluator_base< _Tp, _NData, _Result, _Walker >::collect (const node_data_type & __*data*, const return_value & __*cresult*) [inline, virtual, inherited]

Reimplemented in forward_evaluator_base< _Tp, _NData, _Result, _Walker >, backward_evaluator_base< _Tp, _NData, _Result, _Walker >, cached_forward_evaluator_base< _Tp, _NData, _Result, _Walker >, cached_backward_evaluator_base< _Tp, _NData, _Result, _Walker >, cached_forward_evaluator_base< std::vector< std::vector< interval > > *, expression_node, bool, model::walker >, cached_forward_evaluator_base< func_eval_type, expression_node, double, model::walker >, cached_forward_evaluator_base< func_islp_eval_type, expression_node, func_islp_return_type, model::walker >, cached_forward_evaluator_base< cinterval_eval_type, expression_node, cinterval, model::walker >, cached_forward_evaluator_base< interval_eval_type, expression_node, interval, model::walker >, cached_forward_evaluator_base< func_h_eval_type, expression_node, func_h_eval_ret, model::walker >, cached_forward_evaluator_base< func_d_eval_type, expression_node, double, model::walker >, cached_forward_evaluator_base< std::vector< std::vector< double > > *, expression_node, bool, model::walker >, cached_forward_evaluator_base< model::detect_0chain_visitor_st, expression_node, std::pair< unsigned int, unsigned int >, model::walker >, cached_forward_evaluator_base< infbound_eval_type, expression_node, infbound, model::walker >, cached_forward_evaluator_base< b_interval_eval_type, expression_node, b_interval, model::walker >, cached_forward_evaluator_base< analyticd_eval_type, expression_node, analyticd, model::walker >, cached_forward_evaluator_base< func_id_eval_type, expression_node, interval, model::walker >, cached_backward_evaluator_base< ider_eval_type, expression_node, bool, model::walker >, cached_backward_evaluator_base< islp_eval_type, expression_node, bool, model::walker >, cached_backward_evaluator_base< lincoeff_visitor_st, expression_node, lincoeff_visitor_ret, model::walker >, and cached_backward_evaluator_base< der_eval_type, expression_node, bool, model::walker >.

Definition at line 265 of file evaluator.h.

#### 6.21.3.2 template<class _Tp, class _NData, class _Result, class _Walker> virtual void _evaluator_base< _Tp, _NData, _Result, _Walker >::postorder (const node_data_type & __*data*) [inline, virtual, inherited]

Reimplemented in forward_evaluator_base< _Tp, _NData, _Result, _Walker >, backward_evaluator_base< _Tp, _NData, _Result, _Walker >, cached_forward_evaluator_base< _Tp, _NData, _Result, _Walker >, cached_backward_evaluator_base< _Tp, _NData, _Result, _Walker >, cached_forward_evaluator_base< std::vector< std::vector< interval > > *, expression_node, bool, model::walker >, cached_forward_evaluator_base< func_eval_type, expression_node, double, model::walker >, cached_forward_evaluator_base< func_islp_eval_type, expression_node, func_islp_return_type, model::walker >, cached_forward_evaluator_base< cinterval_eval_type, expression_node, cinterval, model::walker >, cached_forward_evaluator_base< interval_eval_type, expression_node, interval, model::walker >, cached_forward_

evaluator_base< func_h_eval_type, expression_node, func_h_eval_ret, model::walker >, cached_forward_evaluator_base< func_d_eval_type, expression_node, double, model::walker >, cached_forward_evaluator_base< std::vector< std::vector< double > > *, expression_node, bool, model::walker >, cached_forward_evaluator_base< model::detect_0chain_visitor_st, expression_node, std::pair< unsigned int, unsigned int >, model::walker >, cached_forward_evaluator_base< infbound_eval_type, expression_node, infbound, model::walker >, cached_forward_evaluator_base< b_interval_eval_type, expression_node, b_interval, model::walker >, cached_forward_evaluator_base< analyticd_eval_type, expression_node, analyticd, model::walker >, cached_forward_evaluator_base< func_id_eval_type, expression_node, interval, model::walker >, cached_backward_evaluator_base< ider_eval_type, expression_node, bool, model::walker >, cached_backward_evaluator_base< islp_eval_type, expression_node, bool, model::walker >, cached_backward_evaluator_base< lincoeff_visitor_st, expression_node, lincoeff_visitor_ret, model::walker >, and cached_backward_evaluator_base< der_eval_type, expression_node, bool, model::walker >.

Definition at line 268 of file evaluator.h.

#### 6.21.3.3 template<class _Tp, class _NData, class _Result, class _Walker> virtual int cached_evaluator_base< _Tp, _NData, _Result, _Walker >::preorder (const node_data_type & *_data*) [inline, virtual]

Reimplemented in cached_forward_evaluator_base< _Tp, _NData, _Result, _Walker >, cached_backward_evaluator_base< _Tp, _NData, _Result, _Walker >, cached_forward_evaluator_base< std::vector< std::vector< interval > > *, expression_node, bool, model::walker >, cached_forward_evaluator_base< func_eval_type, expression_node, double, model::walker >, cached_forward_evaluator_base< func_islp_eval_type, expression_node, func_islp_return_type, model::walker >, cached_forward_evaluator_base< cinterval_eval_type, expression_node, cinterval, model::walker >, cached_forward_evaluator_base< interval_eval_type, expression_node, interval, model::walker >, cached_forward_evaluator_base< func_h_eval_type, expression_node, func_h_eval_ret, model::walker >, cached_forward_evaluator_base< func_d_eval_type, expression_node, double, model::walker >, cached_forward_evaluator_base< std::vector< std::vector< double > > *, expression_node, bool, model::walker >, cached_forward_evaluator_base< model::detect_0chain_visitor_st, expression_node, std::pair< unsigned int, unsigned int >, model::walker >, cached_forward_evaluator_base< infbound_eval_type, expression_node, infbound, model::walker >, cached_forward_evaluator_base< b_interval_eval_type, expression_node, b_interval, model::walker >, cached_forward_evaluator_base< analyticd_eval_type, expression_node, analyticd, model::walker >, cached_forward_evaluator_base< func_id_eval_type, expression_node, interval, model::walker >, cached_backward_evaluator_base< ider_eval_type, expression_node, bool, model::walker >, cached_backward_evaluator_base< islp_eval_type, expression_node, bool, model::walker >, cached_backward_evaluator_base< lincoeff_visitor_st, expression_node, lincoeff_visitor_ret, model::walker >, and cached_backward_evaluator_base< der_eval_type, expression_node, bool, model::walker >.

Definition at line 302 of file evaluator.h.

#### 6.21.3.4 template<class _Tp, class _NData, class _Result, class _Walker> virtual walker cached_evaluator_base< _Tp, _NData, _Result, _Walker >::short_cut_to (const node_data_type & *_data*) [inline, virtual]

Definition at line 303 of file evaluator.h.

#### 6.21.3.5 template<class _Tp, class _NData, class _Result, class _Walker> virtual return_value _evaluator_base< _Tp, _NData, _Result, _Walker >::value () [inline, virtual, inherited]

Reimplemented in forward_evaluator_base< _Tp, _NData, _Result, _Walker >, backward_evaluator_base< _Tp, _NData, _Result, _Walker >, cached_forward_evaluator_base< _Tp, _NData, _Result, _Walker >, cached_backward_evaluator_base< _Tp, _NData, _Result, _Walker >, cached_forward_evaluator_base<

std::vector< std::vector< interval > > *, expression_node, bool, model::walker >, cached_forward_evaluator_base< func_eval_type, expression_node, double, model::walker >, cached_forward_evaluator_base< func_islp_eval_type, expression_node, func_islp_return_type, model::walker >, cached_forward_evaluator_base< cinterval_eval_type, expression_node, cinterval, model::walker >, cached_forward_evaluator_base< interval_eval_type, expression_node, interval, model::walker >, cached_forward_evaluator_base< func_h_eval_type, expression_node, func_h_eval_ret, model::walker >, cached_forward_evaluator_base< func_d_eval_type, expression_node, double, model::walker >, cached_forward_evaluator_base< std::vector< std::vector< double > > *, expression_node, bool, model::walker >, cached_forward_evaluator_base< model::detect_0chain_visitor_st, expression_node, std::pair< unsigned int, unsigned int >, model::walker >, cached_forward_evaluator_base< infbound_eval_type, expression_node, infbound, model::walker >, cached_forward_evaluator_base< b_interval_eval_type, expression_node, b_interval, model::walker >, cached_forward_evaluator_base< analyticd_eval_type, expression_node, analyticd, model::walker >, cached_forward_evaluator_base< func_id_eval_type, expression_node, interval, model::walker >, cached_backward_evaluator_base< ider_eval_type, expression_node, bool, model::walker >, cached_backward_evaluator_base< islp_eval_type, expression_node, bool, model::walker >, cached_backward_evaluator_base< lincoeff_visitor_st, expression_node, lincoeff_visitor_ret, model::walker >, and cached_backward_evaluator_base< der_eval_type, expression_node, bool, model::walker >.

Definition at line 262 of file evaluator.h.

#### 6.21.3.6 template<class _Tp, class _NData, class _Result, class _Walker> virtual int _evaluator_base< _Tp, _NData, _Result, _Walker >::vcollect (const return_value & *__cresult*) [inline, virtual, inherited]

Reimplemented in forward_evaluator_base< _Tp, _NData, _Result, _Walker >, backward_evaluator_base< _Tp, _NData, _Result, _Walker >, cached_forward_evaluator_base< _Tp, _NData, _Result, _Walker >, cached_backward_evaluator_base< _Tp, _NData, _Result, _Walker >, cached_forward_evaluator_base< std::vector< std::vector< interval > > *, expression_node, bool, model::walker >, cached_forward_evaluator_base< func_eval_type, expression_node, double, model::walker >, cached_forward_evaluator_base< func_islp_eval_type, expression_node, func_islp_return_type, model::walker >, cached_forward_evaluator_base< cinterval_eval_type, expression_node, cinterval, model::walker >, cached_forward_evaluator_base< interval_eval_type, expression_node, interval, model::walker >, cached_forward_evaluator_base< func_h_eval_type, expression_node, func_h_eval_ret, model::walker >, cached_forward_evaluator_base< func_d_eval_type, expression_node, double, model::walker >, cached_forward_evaluator_base< std::vector< std::vector< double > > *, expression_node, bool, model::walker >, cached_forward_evaluator_base< model::detect_0chain_visitor_st, expression_node, std::pair< unsigned int, unsigned int >, model::walker >, cached_forward_evaluator_base< infbound_eval_type, expression_node, infbound, model::walker >, cached_forward_evaluator_base< b_interval_eval_type, expression_node, b_interval, model::walker >, cached_forward_evaluator_base< analyticd_eval_type, expression_node, analyticd, model::walker >, cached_forward_evaluator_base< func_id_eval_type, expression_node, interval, model::walker >, cached_backward_evaluator_base< ider_eval_type, expression_node, bool, model::walker >, cached_backward_evaluator_base< islp_eval_type, expression_node, bool, model::walker >, cached_backward_evaluator_base< lincoeff_visitor_st, expression_node, lincoeff_visitor_ret, model::walker >, and cached_backward_evaluator_base< der_eval_type, expression_node, bool, model::walker >.

Definition at line 264 of file evaluator.h.

#### 6.21.3.7 template<class _Tp, class _NData, class _Result, class _Walker> virtual return_value _evaluator_base< _Tp, _NData, _Result, _Walker >::vvalue () [inline, virtual, inherited]

Reimplemented in forward_evaluator_base< _Tp, _NData, _Result, _Walker >, backward_evaluator_base< _Tp, _NData, _Result, _Walker >, cached_forward_evaluator_base< _Tp, _NData, _Result, _Walker >, cached_backward_evaluator_base< _Tp, _NData, _Result, _Walker >, cached_forward_evaluator_base<

std::vector< std::vector< interval > > *, expression_node, bool, model::walker >, cached_forward_evaluator_base< func_eval_type, expression_node, double, model::walker >, cached_forward_evaluator_base< func_islp_eval_type, expression_node, func_islp_return_type, model::walker >, cached_forward_evaluator_base< cinterval_eval_type, expression_node, cinterval, model::walker >, cached_forward_evaluator_base< interval_eval_type, expression_node, interval, model::walker >, cached_forward_evaluator_base< func_h_eval_type, expression_node, func_h_eval_ret, model::walker >, cached_forward_evaluator_base< func_d_eval_type, expression_node, double, model::walker >, cached_forward_evaluator_base< std::vector< std::vector< double > > *, expression_node, bool, model::walker >, cached_forward_evaluator_base< model::detect_0chain_visitor_st, expression_node, std::pair< unsigned int, unsigned int >, model::walker >, cached_forward_evaluator_base< infbound_eval_type, expression_node, infbound, model::walker >, cached_forward_evaluator_base< b_interval_eval_type, expression_node, b_interval, model::walker >, cached_forward_evaluator_base< analyticd_eval_type, expression_node, analyticd, model::walker >, cached_forward_evaluator_base< func_id_eval_type, expression_node, interval, model::walker >, cached_backward_evaluator_base< ider_eval_type, expression_node, bool, model::walker >, cached_backward_evaluator_base< islp_eval_type, expression_node, bool, model::walker >, cached_backward_evaluator_base< lincoeff_visitor_st, expression_node, lincoeff_visitor_ret, model::walker >, and cached_backward_evaluator_base< der_eval_type, expression_node, bool, model::walker >.

Definition at line 261 of file evaluator.h.

### 6.21.4 Member Data Documentation

#### 6.21.4.1 template<class _Tp, class _NData, class _Result, class _Walker> _Tp _evaluator_base< _Tp, _NData, _Result, _Walker >::eval_data [protected, inherited]

Definition at line 252 of file evaluator.h.

#### 6.21.4.2 template<class _Tp, class _NData, class _Result, class _Walker> const variable_indicator* cached_evaluator_base< _Tp, _NData, _Result, _Walker >::v_ind [protected]

Definition at line 295 of file evaluator.h.

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

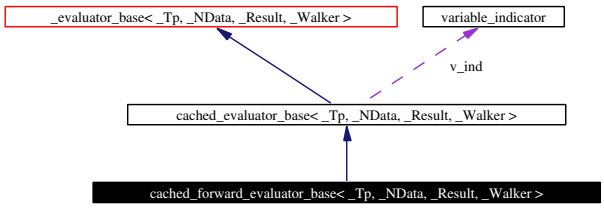
- evaluator.h

## 6.22 cached_forward_evaluator_base< _Tp, _NData, _Result, _Walker > Class Template Reference

`#include <evaluator.h>`

Inheritance diagram for cached_forward_evaluator_base< _Tp, _NData, _Result, _Walker >:

Collaboration diagram for cached_forward_evaluator_base< _Tp, _NData, _Result, _Walker >:




## Public Types

- typedef cached_evaluator_base< _Tp, _NData, _Result, _Walker >::node_data_type node_data_type
- typedef cached_evaluator_base< _Tp, _NData, _Result, _Walker >::return_value return_value
- typedef cached_evaluator_base< _Tp, _NData, _Result, _Walker >::walker walker
- typedef _Tp data_type

## Public Methods

- virtual bool is_cached (const node_data_type &__data)
- int preorder (const node_data_type &__data)
- void postorder (const node_data_type &__data)
- int collect (const node_data_type &__data, const return_value &__rval)
- int vcollect (const return_value &__rval)
- return_value value ()
- return_value vvalue ()
- void vinit ()
- virtual void initialize ()
- virtual int initialize (const node_data_type &__data)
- virtual void calculate (const node_data_type &__data)
- virtual void retrieve_from_cache (const node_data_type &__data)
- virtual void cleanup (const node_data_type &__data)
- virtual int update (const node_data_type &__data, const return_value &__rval)
- virtual int update (const return_value &__rval)
- virtual return_value calculate_value (bool eval_all)
- virtual walker short_cut_to (const node_data_type &__data) PURE_VIRTUAL public

**Protected Attributes**

- const variable_indicator * v_ind
- _Tp eval_data

**template<class _Tp, class _NData, class _Result, class _Walker> class cached_forward_evaluator_base< _Tp, _NData, _Result, _Walker >**

### 6.22.1 Member Typedef Documentation

#### 6.22.1.1 template<class _Tp, class _NData, class _Result, class _Walker> typedef _Tp _evaluator_base< _Tp, _NData, _Result, _Walker >::data_type [inherited]

Definition at line 245 of file evaluator.h.

#### 6.22.1.2 template<class _Tp, class _NData, class _Result, class _Walker> typedef cached_evaluator_base<_Tp, _NData, _Result,_Walker>::node_data_type cached_forward_evaluator_base< _Tp, _NData, _Result, _Walker >::node_data_type

Reimplemented from cached_evaluator_base< _Tp, _NData, _Result, _Walker >.

Definition at line 396 of file evaluator.h.

#### 6.22.1.3 template<class _Tp, class _NData, class _Result, class _Walker> typedef cached_evaluator_base<_Tp, _NData, _Result,_Walker>::return_value cached_forward_evaluator_base< _Tp, _NData, _Result, _Walker >::return_value

Reimplemented from cached_evaluator_base< _Tp, _NData, _Result, _Walker >.

Definition at line 398 of file evaluator.h.

#### 6.22.1.4 template<class _Tp, class _NData, class _Result, class _Walker> typedef cached_evaluator_base<_Tp, _NData, _Result,_Walker>::walker cached_forward_evaluator_base< _Tp, _NData, _Result, _Walker >::walker

Reimplemented from cached_evaluator_base< _Tp, _NData, _Result, _Walker >.

Definition at line 400 of file evaluator.h.

### 6.22.2 Member Function Documentation

#### 6.22.2.1 template<class _Tp, class _NData, class _Result, class _Walker> virtual void cached_forward_evaluator_base< _Tp, _NData, _Result, _Walker >::calculate (const node_data_type & __*data*) [inline, virtual]

Definition at line 430 of file evaluator.h.

#### 6.22.2.2 template<class _Tp, class _NData, class _Result, class _Walker> virtual return_value cached_forward_evaluator_base< _Tp, _NData, _Result, _Walker >::calculate_value (bool *eval_all*) [inline, virtual]

Reimplemented in analyticd_eval, b_interval_eval, cinterval_eval, prep_d_eval, func_d_eval, func_eval, func_h_eval, prep_id_eval, func_id_eval, infbound_eval, interval_eval, prep_islp_eval, func_islp_eval, and detect_0chain_visitor.

Definition at line 436 of file evaluator.h.

#### 6.22.2.3 template<class _Tp, class _NData, class _Result, class _Walker> virtual void cached_forward_evaluator_base< _Tp, _NData, _Result, _Walker >::cleanup (const node_data_type & __data) [inline, virtual]

Definition at line 432 of file evaluator.h.

#### 6.22.2.4 template<class _Tp, class _NData, class _Result, class _Walker> int cached_forward_evaluator_base< _Tp, _NData, _Result, _Walker >::collect (const node_data_type & __data, const return_value & __rval) [inline, virtual]

Reimplemented from _evaluator_base< _Tp, _NData, _Result, _Walker >.

Definition at line 419 of file evaluator.h.

#### 6.22.2.5 template<class _Tp, class _NData, class _Result, class _Walker> virtual int cached_forward_evaluator_base< _Tp, _NData, _Result, _Walker >::initialize (const node_data_type & __data) [inline, virtual]

Definition at line 429 of file evaluator.h.

#### 6.22.2.6 template<class _Tp, class _NData, class _Result, class _Walker> virtual void cached_forward_evaluator_base< _Tp, _NData, _Result, _Walker >::initialize () [inline, virtual]

Reimplemented in analyticd_eval, b_interval_eval, cinterval_eval, prep_d_eval, func_d_eval, func_eval, func_h_eval, prep_id_eval, func_id_eval, infbound_eval, interval_eval, prep_islp_eval, func_islp_eval, and detect_0chain_visitor.

Definition at line 428 of file evaluator.h.

#### 6.22.2.7 template<class _Tp, class _NData, class _Result, class _Walker> virtual bool cached_forward_evaluator_base< _Tp, _NData, _Result, _Walker >::is_cached (const node_data_type & __data) [inline, virtual]

Reimplemented in analyticd_eval, b_interval_eval, cinterval_eval, func_d_eval, func_eval, func_h_eval, func_id_eval, infbound_eval, interval_eval, and func_islp_eval.

Definition at line 405 of file evaluator.h.

#### 6.22.2.8 template<class _Tp, class _NData, class _Result, class _Walker> void cached_forward_evaluator_base< _Tp, _NData, _Result, _Walker >::postorder (const node_data_type & __data) [inline, virtual]

Reimplemented from _evaluator_base< _Tp, _NData, _Result, _Walker >.

Definition at line 417 of file evaluator.h.

#### 6.22.2.9 template<class _Tp, class _NData, class _Result, class _Walker> int cached_forward_evaluator_base< _Tp, _NData, _Result, _Walker >::preorder (const node_data_type & __data) [inline, virtual]

Reimplemented from cached_evaluator_base< _Tp, _NData, _Result, _Walker >.

Definition at line 408 of file evaluator.h.

**6.22.2.10 template<class _Tp, class _NData, class _Result, class _Walker> virtual void cached_forward_evaluator_base< _Tp, _NData, _Result, _Walker >::retrieve_from_cache (const node_data_type & __data)** `[inline, virtual]`

Definition at line 431 of file evaluator.h.

**6.22.2.11 template<class _Tp, class _NData, class _Result, class _Walker> virtual walker cached_evaluator_base< _Tp, _NData, _Result, _Walker >::short_cut_to (const node_data_type & __data)** `[inline, virtual, inherited]`

Definition at line 303 of file evaluator.h.

**6.22.2.12 template<class _Tp, class _NData, class _Result, class _Walker> virtual int cached_forward_evaluator_base< _Tp, _NData, _Result, _Walker >::update (const return_value & __rval)** `[inline, virtual]`

Definition at line 435 of file evaluator.h.

**6.22.2.13 template<class _Tp, class _NData, class _Result, class _Walker> virtual int cached_forward_evaluator_base< _Tp, _NData, _Result, _Walker >::update (const node_data_type & __data, const return_value & __rval)** `[inline, virtual]`

Definition at line 433 of file evaluator.h.

**6.22.2.14 template<class _Tp, class _NData, class _Result, class _Walker> return_value cached_forward_evaluator_base< _Tp, _NData, _Result, _Walker >::value ()** `[inline, virtual]`

Reimplemented from _evaluator_base< _Tp, _NData, _Result, _Walker >.

Definition at line 423 of file evaluator.h.

**6.22.2.15 template<class _Tp, class _NData, class _Result, class _Walker> int cached_forward_evaluator_base< _Tp, _NData, _Result, _Walker >::vcollect (const return_value & __rval)** `[inline, virtual]`

Reimplemented from _evaluator_base< _Tp, _NData, _Result, _Walker >.

Definition at line 421 of file evaluator.h.

**6.22.2.16 template<class _Tp, class _NData, class _Result, class _Walker> void cached_forward_evaluator_base< _Tp, _NData, _Result, _Walker >::vinit ()** `[inline]`

Definition at line 425 of file evaluator.h.

**6.22.2.17 template<class _Tp, class _NData, class _Result, class _Walker> return_value cached_forward_evaluator_base< _Tp, _NData, _Result, _Walker >::vvalue ()** `[inline, virtual]`

Reimplemented from _evaluator_base< _Tp, _NData, _Result, _Walker >.

Definition at line 424 of file evaluator.h.

### 6.22.3 Member Data Documentation

#### 6.22.3.1 template<class _Tp, class _NData, class _Result, class _Walker> _Tp _evaluator_base< _Tp, _NData, _Result, _Walker >::eval_data [protected, inherited]

Definition at line 252 of file evaluator.h.

#### 6.22.3.2 template<class _Tp, class _NData, class _Result, class _Walker> const variable_indicator* cached_evaluator_base< _Tp, _NData, _Result, _Walker >::v_ind [protected, inherited]

Definition at line 295 of file evaluator.h.

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

- evaluator.h

## 6.23 certificate Class Reference

`#include <certificate.h>`

### Public Attributes

- void * cert

### 6.23.1 Member Data Documentation

#### 6.23.1.1 void* certificate::cert

Definition at line 38 of file certificate.h.

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

- certificate.h

## 6.24 children_compare Class Reference

`#include <expr-inline.h>`

### Public Methods

- bool operator() (const expression_node &__x, const expression_node &__y) const

### 6.24.1 Member Function Documentation

#### 6.24.1.1 bool children_compare::operator() (const expression_node & __x, const expression_node & __y) const [inline]

Definition at line 76 of file expr-inline.h.

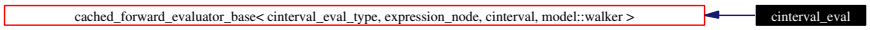
The documentation for this class was generated from the following file:

- expr-inline.h

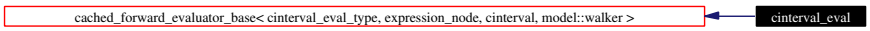
## 6.25 cinterval_eval Class Reference

`#include <cint_evaluator.h>`

Inheritance diagram for cinterval_eval:




Collaboration diagram for cinterval_eval:




### Public Types

- typedef cached_evaluator_base< cinterval_eval_type, expression_node, cinterval, model::walker >::node_data_type node_data_type
- typedef cached_evaluator_base< cinterval_eval_type, expression_node, cinterval, model::walker >::return_value return_value
- typedef cached_evaluator_base< cinterval_eval_type, expression_node, cinterval, model::walker >::walker walker
- typedef cinterval_eval_type data_type

### Public Methods

- cinterval_eval (const std::vector< cinterval > &__x, const variable_indicator &__v, const model &__m, std::vector< cinterval > *__c)
- cinterval_eval (const cinterval_eval &__v)
- ~cinterval_eval ()
- model::walker short_cut_to (const expression_node &__data)
- void initialize ()
- int initialize (const expression_node &__data)
- void calculate (const expression_node &__data)
- void retrieve_from_cache (const expression_node &__data)
- int update (const cinterval &__rval)
- int update (const expression_node &__data, const cinterval &__rval)
- cinterval calculate_value (bool eval_all)
- int preorder (const node_data_type &__data)
- void postorder (const node_data_type &__data)
- int collect (const node_data_type &__data, const return_value &__rval)
- int vcollect (const return_value &__rval)
- return_value value ()
- return_value vvalue ()
- void vinit ()
- virtual int initialize (const node_data_type &__data)
- virtual void calculate (const node_data_type &__data)
- virtual void retrieve_from_cache (const node_data_type &__data)
- virtual void cleanup (const node_data_type &__data)
- virtual int update (const node_data_type &__data, const return_value &__rval)
- virtual int update (const return_value &__rval)
- virtual walker short_cut_to (const node_data_type &__data) PURE_VIRTUAL public

**Protected Methods**

- bool is_cached (const node_data_type &__data)

**Protected Attributes**

- const variable_indicator * v_ind
- cinterval_eval_type eval_data

### 6.25.1 Member Typedef Documentation

#### 6.25.1.1 typedef cinterval_eval_type _evaluator_base< cinterval_eval_type, expression_node, cinterval, model::walker >::data_type [inherited]

Definition at line 245 of file evaluator.h.

#### 6.25.1.2 typedef cached_evaluator_base<cinterval_eval_type, expression_node, cinterval,model::walker>::node_data_type cached_forward_evaluator_base< cinterval_eval_type, expression_node, cinterval, model::walker >::node_data_type [inherited]

Reimplemented from cached_evaluator_base< cinterval_eval_type, expression_node, cinterval, model::walker >.

Definition at line 396 of file evaluator.h.

#### 6.25.1.3 typedef cached_evaluator_base<cinterval_eval_type, expression_node, cinterval,model::walker>::return_value cached_forward_evaluator_base< cinterval_eval_type, expression_node, cinterval, model::walker >::return_value [inherited]

Reimplemented from cached_evaluator_base< cinterval_eval_type, expression_node, cinterval, model::walker >.

Definition at line 398 of file evaluator.h.

#### 6.25.1.4 typedef cached_evaluator_base<cinterval_eval_type, expression_node, cinterval,model::walker>::walker cached_forward_evaluator_base< cinterval_eval_type, expression_node, cinterval, model::walker >::walker [inherited]

Reimplemented from cached_evaluator_base< cinterval_eval_type, expression_node, cinterval, model::walker >.

Definition at line 400 of file evaluator.h.

### 6.25.2 Constructor & Destructor Documentation

#### 6.25.2.1 cinterval_eval::cinterval_eval (const std::vector< cinterval > & __x, const variable_indicator & __v, const model & __m, std::vector< cinterval > * __c) [inline]

Definition at line 111 of file cint_evaluator.h.

#### 6.25.2.2 cinterval_eval::cinterval_eval (const cinterval_eval & __v) [inline]

Definition at line 123 of file cint_evaluator.h.

#### 6.25.2.3 cinterval_eval::~cinterval_eval () [inline]

Definition at line 125 of file cint_evaluator.h.

### 6.25.3 Member Function Documentation

#### 6.25.3.1 virtual void cached_forward_evaluator_base< cinterval_eval_type, expression_node, cinterval, model::walker >::calculate (const node_data_type & *__data*) [inline, virtual, inherited]

Definition at line 430 of file evaluator.h.

#### 6.25.3.2 void cinterval_eval::calculate (const expression_node & *__data*) [inline]

Definition at line 188 of file cint_evaluator.h.

#### 6.25.3.3 cinterval cinterval_eval::calculate_value (bool *eval_all*) [inline, virtual]

Reimplemented from cached_forward_evaluator_base< cinterval_eval_type, expression_node, cinterval, model::walker >.

Definition at line 674 of file cint_evaluator.h.

#### 6.25.3.4 virtual void cached_forward_evaluator_base< cinterval_eval_type, expression_node, cinterval, model::walker >::cleanup (const node_data_type & *__data*) [inline, virtual, inherited]

Definition at line 432 of file evaluator.h.

#### 6.25.3.5 int cached_forward_evaluator_base< cinterval_eval_type, expression_node, cinterval, model::walker >::collect (const node_data_type & *__data*, const return_value & *__rval*) [inline, virtual, inherited]

Reimplemented from _evaluator_base< cinterval_eval_type, expression_node, cinterval, model::walker >.

Definition at line 419 of file evaluator.h.

#### 6.25.3.6 virtual int cached_forward_evaluator_base< cinterval_eval_type, expression_node, cinterval, model::walker >::initialize (const node_data_type & *__data*) [inline, virtual, inherited]

Definition at line 429 of file evaluator.h.

#### 6.25.3.7 int cinterval_eval::initialize (const expression_node & *__data*) [inline]

Definition at line 132 of file cint_evaluator.h.

#### 6.25.3.8 void cinterval_eval::initialize () [inline, virtual]

Reimplemented from cached_forward_evaluator_base< cinterval_eval_type, expression_node, cinterval, model::walker >.

Definition at line 130 of file cint_evaluator.h.

#### 6.25.3.9 bool cinterval_eval::is_cached (const node_data_type & *__data*) [inline, protected, virtual]

Reimplemented from cached_forward_evaluator_base< cinterval_eval_type, expression_node, cinterval, model::walker >.

Definition at line 65 of file cint_evaluator.h.

#### 6.25.3.10 void cached_forward_evaluator_base< cinterval_eval_type, expression_node, cinterval, model::walker >::postorder (const node_data_type & *__data*) [inline, virtual, inherited]

Reimplemented from _evaluator_base< cinterval_eval_type, expression_node, cinterval, model::walker >.

Definition at line 417 of file evaluator.h.

#### 6.25.3.11 int cached_forward_evaluator_base< cinterval_eval_type, expression_node, cinterval, model::walker >::preorder (const node_data_type & *__data*) [inline, virtual, inherited]

Reimplemented from cached_evaluator_base< cinterval_eval_type, expression_node, cinterval, model::walker >.

Definition at line 408 of file evaluator.h.

#### 6.25.3.12 virtual void cached_forward_evaluator_base< cinterval_eval_type, expression_node, cinterval, model::walker >::retrieve_from_cache (const node_data_type & *__data*) [inline, virtual, inherited]

Definition at line 431 of file evaluator.h.

#### 6.25.3.13 void cinterval_eval::retrieve_from_cache (const expression_node & *__data*) [inline]

Definition at line 198 of file cint_evaluator.h.

#### 6.25.3.14 virtual walker cached_evaluator_base< cinterval_eval_type, expression_node, cinterval, model::walker >::short_cut_to (const node_data_type & *__data*) [inline, virtual, inherited]

Definition at line 303 of file evaluator.h.

#### 6.25.3.15 model::walker cinterval_eval::short_cut_to (const expression_node & *__data*) [inline]

Definition at line 127 of file cint_evaluator.h.

#### 6.25.3.16 virtual int cached_forward_evaluator_base< cinterval_eval_type, expression_node, cinterval, model::walker >::update (const return_value & *__rval*) [inline, virtual, inherited]

Definition at line 435 of file evaluator.h.

#### 6.25.3.17 virtual int cached_forward_evaluator_base< cinterval_eval_type, expression_node, cinterval, model::walker >::update (const node_data_type & __data, const return_value & __rval) [inline, virtual, inherited]

Definition at line 433 of file evaluator.h.

#### 6.25.3.18 int cinterval_eval::update (const expression_node & __data, const cinterval & __rval) [inline]

Definition at line 223 of file cint_evaluator.h.

#### 6.25.3.19 int cinterval_eval::update (const cinterval & __rval) [inline]

Definition at line 217 of file cint_evaluator.h.

#### 6.25.3.20 return_value cached_forward_evaluator_base< cinterval_eval_type, expression_node, cinterval, model::walker >::value () [inline, virtual, inherited]

Reimplemented from _evaluator_base< cinterval_eval_type, expression_node, cinterval, model::walker >.

Definition at line 423 of file evaluator.h.

#### 6.25.3.21 int cached_forward_evaluator_base< cinterval_eval_type, expression_node, cinterval, model::walker >::vcollect (const return_value & __rval) [inline, virtual, inherited]

Reimplemented from _evaluator_base< cinterval_eval_type, expression_node, cinterval, model::walker >.

Definition at line 421 of file evaluator.h.

#### 6.25.3.22 void cached_forward_evaluator_base< cinterval_eval_type, expression_node, cinterval, model::walker >::vinit () [inline, inherited]

Definition at line 425 of file evaluator.h.

#### 6.25.3.23 return_value cached_forward_evaluator_base< cinterval_eval_type, expression_node, cinterval, model::walker >::vvalue () [inline, virtual, inherited]

Reimplemented from _evaluator_base< cinterval_eval_type, expression_node, cinterval, model::walker >.

Definition at line 424 of file evaluator.h.

### 6.25.4 Member Data Documentation

#### 6.25.4.1 cinterval_eval_type _evaluator_base< cinterval_eval_type, expression_node, cinterval, model::walker >::eval_data [protected, inherited]

Definition at line 252 of file evaluator.h.

#### 6.25.4.2 const variable_indicator* cached_evaluator_base< cinterval_eval_type, expression_node, cinterval, model::walker >::v_ind [protected, inherited]

Definition at line 295 of file evaluator.h.

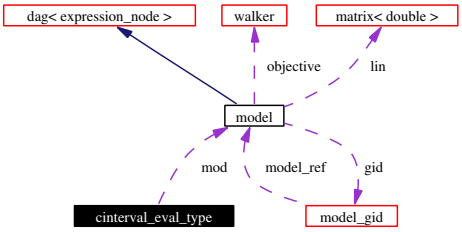
The documentation for this class was generated from the following file:

- cint_evaluator.h

## 6.26 cinterval_eval_type Struct Reference

`#include <cint_evaluator.h>`

Collaboration diagram for cinterval_eval_type:




### Public Attributes

- const std::vector< cinterval > * x
- std::vector< cinterval > * cache
- const model * mod
- void * p
- cinterval d
- int info
- cinterval r
- unsigned int n

### 6.26.1 Member Data Documentation

#### 6.26.1.1 std::vector<cinterval>* cinterval_eval_type::cache

Definition at line 47 of file cint_evaluator.h.

#### 6.26.1.2 cinterval cinterval_eval_type::d

Definition at line 50 of file cint_evaluator.h.

#### 6.26.1.3 int cinterval_eval_type::info

Definition at line 51 of file cint_evaluator.h.

#### 6.26.1.4 const model* cinterval_eval_type::mod

Definition at line 48 of file cint_evaluator.h.

#### 6.26.1.5 unsigned int cinterval_eval_type::n

Definition at line 53 of file cint_evaluator.h.

#### 6.26.1.6 void* cinterval_eval_type::p

Definition at line 49 of file cint_evaluator.h.

#### 6.26.1.7 cinterval cinterval_eval_type::r

Definition at line 52 of file cint_evaluator.h.

#### 6.26.1.8 const std::vector<cinterval>* cinterval_eval_type::x

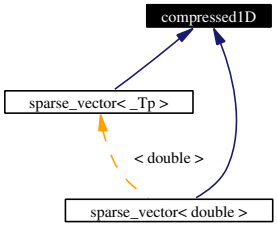
Definition at line 46 of file cint_evaluator.h.

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

- cint_evaluator.h

## 6.27 compressed1D Class Reference

Inheritance diagram for compressed1D:



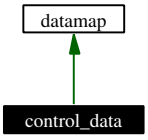

The documentation for this class was generated from the following file:

- linalg.h

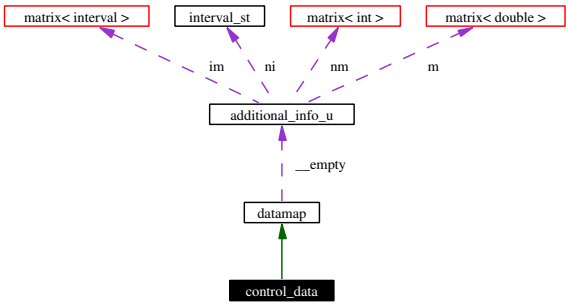
## 6.28 control_data Class Reference

`#include <control_data.h>`

Inheritance diagram for control_data:




Collaboration diagram for control_data:




## Public Methods

- control_data ()
- control_data (const std::string &__n, const additional_info_u &__v)
- control_data (const char *__n, const additional_info_u &__v)
- control_data (const control_data &__c)
- control_data (const std::string &serv, const info_contents &inf)
- virtual ~control_data ()
- control_data & operator= (const control_data &__c)
- control_data & operator= (const info_contents &__i)
- void service (const std::string &__s)
- void service (const char *__s)
- const std::string & service () const
- bool check_service (const std::string &__n) const
- bool check_service (const char *__n) const
- void set (const info_contents &__i)
- void set (const std::string &__s, const additional_info_u &__h)
- void set (const char *__cp, const additional_info_u &__h)
- void set (const std::string &__s, int i, const additional_info_u &__h)
- void set (const char *__cp, int i, const additional_info_u &__h)
- const additional_info_u & get (const std::string &__s) const
- const additional_info_u & get (const char *__cp) const
- const additional_info_u & get (const std::string &__s, int i) const
- const additional_info_u & get (const char *__cp, int i) const
- void unset (const std::string &__s)
- void unset (const char *__cp)
- void unset (const std::string &__s, int i)
- void unset (const char *__cp, int i)
- bool is_set (const std::string &__s) const
- bool is_set (const char *__cp) const
- bool is_set (const std::string &__s, int i) const
- bool is_set (const char *__cp, int i) const
- bool which_set (const std::string &__s, std::vector< int > &__idx) const
- bool which_set (const char *__cp, std::vector< int > &__idx) const
- template<class _S> void assign (const std::string &__s, const std::vector< _S > *&__b) const
- template<class _S> void assign (const std::string &__s, const matrix< _S > *&__b) const
- template<class _S> void assign (const std::string &__s, _S &__b) const
- void assign (const std::string &__s, std::string &__b, const char *__def) const

- template<class _S> void assign (const std::string &__s, const std::vector< _S > *&__b, const std::vector< _S > *__def) const
- template<class _S> void assign (const std::string &__s, const matrix< _S > *&__b, const matrix< _S > *__def) const
- template<class _S> void assign (const std::string &__s, _S &__b, _S __def) const
- void assign (const char *__s, std::string &__b, const char *__def) const
- template<class _S> void assign (const char *__s, const std::vector< _S > *&__b) const
- template<class _S> void assign (const char *__s, const matrix< _S > *&__b) const
- template<class _S> void assign (const char *__s, _S &__b) const
- template<class _S> void assign (const char *__s, const std::vector< _S > *&__b, const std::vector< _S > *__def) const
- template<class _S> void assign (const char *__s, const matrix< _S > *&__b, const matrix< _S > *__def) const
- template<class _S> void assign (const char *__s, _S &__b, _S __def) const
- template<class _S> void assign_i (const std::string &__s, int i, const std::vector< _S > *&__b) const
- template<class _S> void assign_i (const std::string &__s, int i, const matrix< _S > *&__b) const
- template<class _S> void assign_i (const std::string &__s, int i, _S &__b) const
- void assign_i (const std::string &__s, int i, std::string &__b, const char *__def) const
- template<class _S> void assign_i (const std::string &__s, int i, const std::vector< _S > *&__b, const std::vector< _S > *__def) const
- template<class _S> void assign_i (const std::string &__s, int i, const matrix< _S > *&__b, const matrix< _S > *__def) const
- template<class _S> void assign_i (const std::string &__s, int i, _S &__b, _S __def) const
- template<class _S> void assign_i (const char *__s, int i, const std::vector< _S > *&__b) const
- template<class _S> void assign_i (const char *__s, int i, const matrix< _S > *&__b) const
- template<class _S> void assign_i (const char *__s, int i, _S &__b) const
- void assign_i (const char *__s, int i, std::string &__b, const char *__def) const
- template<class _S> void assign_i (const char *__s, int i, const std::vector< _S > *&__b, const std::vector< _S > *__def) const
- template<class _S> void assign_i (const char *__s, int i, const matrix< _S > *&__b, const matrix< _S > *__def) const
- template<class _S> void assign_i (const char *__s, int i, _S &__b, _S __def) const

### Protected Methods

- bool sinsert (const std::string &__s, const additional_info_u &__h, bool replace)
- bool sinsert (const char *__cp, const additional_info_u &__h, bool replace)
- bool sinsert (const std::string &__s, int i, const additional_info_u &__h, bool replace)
- bool sinsert (const char *__cp, int i, const additional_info_u &__h, bool replace)
- const additional_info_u & sfind (const std::string &__s) const
- const additional_info_u & sfind (const char *__cp) const
- const additional_info_u & sfind (const std::string &__s, int i) const
- const additional_info_u & sfind (const char *__cp, int i) const
- void remove (const std::string &__s)
- void remove (const char *__cp)
- void remove (const std::string &__s, int i)
- void remove (const char *__cp, int i)
- bool defd (const std::string &__s) const
- bool defd (const char *__cp) const
- bool defd (const std::string &__s, int i) const

- bool defd (const char *__cp, int i) const
- bool which (const std::string &__s, std::vector< int > &__idx) const
- bool which (const char *__cp, std::vector< int > &__idx) const
- bool retrieve (const std::string &__s, bool &__b) const
- bool retrieve (const std::string &__s, bool &__b, bool __def) const
- bool retrieve (const std::string &__s, int &__d) const
- bool retrieve (const std::string &__s, int &__d, int __def) const
- bool retrieve (const std::string &__s, unsigned int &__d) const
- bool retrieve (const std::string &__s, unsigned int &__d, unsigned int __def) const
- bool retrieve (const std::string &__s, double &__d) const
- bool retrieve (const std::string &__s, double &__d, double __def) const
- bool retrieve (const std::string &__s, interval &__b) const
- bool retrieve (const std::string &__s, interval &__b, const interval &__def) const
- bool retrieve (const std::string &__s, std::string &__is) const
- bool retrieve (const std::string &__s, std::string &__is, const std::string &__def) const
- bool retrieve (const std::string &__s, const std::vector< bool > *&__b) const
- bool retrieve (const std::string &__s, const std::vector< bool > *&__b, const std::vector< bool > *__def) const
- bool retrieve (const std::string &__s, const std::vector< int > *&__b) const
- bool retrieve (const std::string &__s, const std::vector< int > *&__b, const std::vector< int > *__def) const
- bool retrieve (const std::string &__s, const std::vector< unsigned int > *&__b) const
- bool retrieve (const std::string &__s, const std::vector< unsigned int > *&__b, const std::vector< unsigned int > *__def) const
- bool retrieve (const std::string &__s, const std::vector< double > *&__b) const
- bool retrieve (const std::string &__s, const std::vector< double > *&__b, const std::vector< double > *__def) const
- bool retrieve (const std::string &__s, const std::vector< interval > *&__b) const
- bool retrieve (const std::string &__s, const std::vector< interval > *&__b, const std::vector< interval > *__def) const
- bool retrieve (const std::string &__s, const matrix< double > *&__b) const
- bool retrieve (const std::string &__s, const matrix< double > *&__b, const matrix< double > *__def) const
- bool retrieve (const std::string &__s, const matrix< int > *&__b) const
- bool retrieve (const std::string &__s, const matrix< int > *&__b, const matrix< int > *__def) const
- bool retrieve (const std::string &__s, const matrix< interval > *&__b) const
- bool retrieve (const std::string &__s, const matrix< interval > *&__b, const matrix< interval > *__def) const
- bool retrieve (const char *__s, bool &__b) const
- bool retrieve (const char *__s, bool &__b, bool __def) const
- bool retrieve (const char *__s, int &__d) const
- bool retrieve (const char *__s, int &__d, int __def) const
- bool retrieve (const char *__s, unsigned int &__d) const
- bool retrieve (const char *__s, unsigned int &__d, unsigned int __def) const
- bool retrieve (const char *__s, double &__d) const
- bool retrieve (const char *__s, double &__d, double __def) const
- bool retrieve (const char *__s, interval &__b) const
- bool retrieve (const char *__s, interval &__b, const interval &__def) const
- bool retrieve (const char *__s, std::string &__is) const
- bool retrieve (const char *__s, std::string &__b, const std::string &__def) const
- bool retrieve (const char *__s, const std::vector< bool > *&__b) const

- bool retrieve (const char *__s, const std::vector< bool > *&__b, const std::vector< bool > *__def) const
- bool retrieve (const char *__s, const std::vector< unsigned int > *&__b) const
- bool retrieve (const char *__s, const std::vector< unsigned int > *&__b, const std::vector< unsigned int > *__def) const
- bool retrieve (const char *__s, const std::vector< int > *&__b) const
- bool retrieve (const char *__s, const std::vector< int > *&__b, const std::vector< int > *__def) const
- bool retrieve (const char *__s, const std::vector< double > *&__b) const
- bool retrieve (const char *__s, const std::vector< double > *&__b, const std::vector< double > *__-def) const
- bool retrieve (const char *__s, const std::vector< interval > *&__b) const
- bool retrieve (const char *__s, const std::vector< interval > *&__b, const std::vector< interval > *__def) const
- bool retrieve (const char *__s, const matrix< double > *&__b) const
- bool retrieve (const char *__s, const matrix< double > *&__b, const matrix< double > *__def) const
- bool retrieve (const char *__s, const matrix< int > *&__b) const
- bool retrieve (const char *__s, const matrix< int > *&__b, const matrix< int > *__def) const
- bool retrieve (const char *__s, const matrix< interval > *&__b) const
- bool retrieve (const char *__s, const matrix< interval > *&__b, const matrix< interval > *__def) const
- bool retrieve_i (const std::string &__s, int i, bool &__b) const
- bool retrieve_i (const std::string &__s, int i, bool &__b, bool __def) const
- bool retrieve_i (const std::string &__s, int i, int &__d) const
- bool retrieve_i (const std::string &__s, int i, int &__d, int __def) const
- bool retrieve_i (const std::string &__s, int i, unsigned int &__d) const
- bool retrieve_i (const std::string &__s, int i, unsigned int &__d, unsigned int __def) const
- bool retrieve_i (const std::string &__s, int i, double &__d) const
- bool retrieve_i (const std::string &__s, int i, double &__d, double __def) const
- bool retrieve_i (const std::string &__s, int i, interval &__b) const
- bool retrieve_i (const std::string &__s, int i, interval &__b, const interval &__def) const
- bool retrieve_i (const std::string &__s, int i, std::string &__is) const
- bool retrieve_i (const std::string &__s, int i, std::string &__is, const std::string &__def) const
- bool retrieve_i (const std::string &__s, int i, const std::vector< bool > *&__b) const
- bool retrieve_i (const std::string &__s, int i, const std::vector< bool > *&__b, const std::vector< bool > *__def) const
- bool retrieve_i (const std::string &__s, int i, const std::vector< int > *&__b) const
- bool retrieve_i (const std::string &__s, int i, const std::vector< int > *&__b, const std::vector< int > *__def) const
- bool retrieve_i (const std::string &__s, int i, const std::vector< unsigned int > *&__b) const
- bool retrieve_i (const std::string &__s, int i, const std::vector< unsigned int > *&__b, const std::vector< unsigned int > *__def) const
- bool retrieve_i (const std::string &__s, int i, const std::vector< double > *&__b) const
- bool retrieve_i (const std::string &__s, int i, const std::vector< double > *&__b, const std::vector< double > *__def) const
- bool retrieve_i (const std::string &__s, int i, const std::vector< interval > *&__b) const
- bool retrieve_i (const std::string &__s, int i, const std::vector< interval > *&__b, const std::vector< interval > *__def) const
- bool retrieve_i (const std::string &__s, int i, const matrix< double > *&__b) const
- bool retrieve_i (const std::string &__s, int i, const matrix< double > *&__b, const matrix< double > *__def) const
- bool retrieve_i (const std::string &__s, int i, const matrix< int > *&__b) const

- bool retrieve_i (const std::string &__s, int i, const matrix< int > *&__b, const matrix< int > *__def) const
- bool retrieve_i (const std::string &__s, int i, const matrix< interval > *&__b) const
- bool retrieve_i (const std::string &__s, int i, const matrix< interval > *&__b, const matrix< interval > *__def) const
- bool retrieve_i (const char *__s, int i, bool &__b) const
- bool retrieve_i (const char *__s, int i, bool &__b, bool __def) const
- bool retrieve_i (const char *__s, int i, int &__d) const
- bool retrieve_i (const char *__s, int i, int &__d, int __def) const
- bool retrieve_i (const char *__s, int i, unsigned int &__d) const
- bool retrieve_i (const char *__s, int i, unsigned int &__d, unsigned int __def) const
- bool retrieve_i (const char *__s, int i, double &__d) const
- bool retrieve_i (const char *__s, int i, double &__d, double __def) const
- bool retrieve_i (const char *__s, int i, interval &__b) const
- bool retrieve_i (const char *__s, int i, interval &__b, const interval &__def) const
- bool retrieve_i (const char *__s, int i, std::string &__is) const
- bool retrieve_i (const char *__s, int i, std::string &__b, const std::string &__def) const
- bool retrieve_i (const char *__s, int i, const std::vector< bool > *&__b) const
- bool retrieve_i (const char *__s, int i, const std::vector< bool > *&__b, const std::vector< bool > *__def) const
- bool retrieve_i (const char *__s, int i, const std::vector< int > *&__b) const
- bool retrieve_i (const char *__s, int i, const std::vector< int > *&__b, const std::vector< int > *__def) const
- bool retrieve_i (const char *__s, int i, const std::vector< unsigned int > *&__b) const
- bool retrieve_i (const char *__s, int i, const std::vector< unsigned int > *&__b, const std::vector< unsigned int > *__def) const
- bool retrieve_i (const char *__s, int i, const std::vector< double > *&__b) const
- bool retrieve_i (const char *__s, int i, const std::vector< double > *&__b, const std::vector< double > *__def) const
- bool retrieve_i (const char *__s, int i, const std::vector< interval > *&__b) const
- bool retrieve_i (const char *__s, int i, const std::vector< interval > *&__b, const std::vector< interval > *__def) const
- bool retrieve_i (const char *__s, int i, const matrix< double > *&__b) const
- bool retrieve_i (const char *__s, int i, const matrix< double > *&__b, const matrix< double > *__def) const
- bool retrieve_i (const char *__s, int i, const matrix< int > *&__b) const
- bool retrieve_i (const char *__s, int i, const matrix< int > *&__b, const matrix< int > *__def) const
- bool retrieve_i (const char *__s, int i, const matrix< interval > *&__b) const
- bool retrieve_i (const char *__s, int i, const matrix< interval > *&__b, const matrix< interval > *__def) const

### 6.28.1 Constructor & Destructor Documentation

#### 6.28.1.1 control_data::control_data () [inline]

Definition at line 41 of file control_data.h.

#### 6.28.1.2 control_data::control_data (const std::string & *__n*, const additional_info_u & *__v*) [inline]

Definition at line 42 of file control_data.h.

#### 6.28.1.3 control_data::control_data (const char * __n, const additional_info_u & __v) [inline]

Definition at line 44 of file control_data.h.

#### 6.28.1.4 control_data::control_data (const control_data & __c) [inline]

Definition at line 46 of file control_data.h.

#### 6.28.1.5 control_data::control_data (const std::string & *serv*, const info_contents & *inf*) [inline]

Definition at line 47 of file control_data.h.

#### 6.28.1.6 virtual control_data::~control_data () [inline, virtual]

Definition at line 49 of file control_data.h.

### 6.28.2 Member Function Documentation

#### 6.28.2.1 template<class _S> void control_data::assign (const char * __s, _S & __b, _S __def) const [inline]

Definition at line 160 of file cdat-inline.h.

#### 6.28.2.2 template<class _S> void control_data::assign (const char * __s, const matrix< _S > *& __b, const matrix< _S > * __def) const [inline]

Definition at line 153 of file cdat-inline.h.

#### 6.28.2.3 template<class _S> void control_data::assign (const char * __s, const std::vector< _S > *& __b, const std::vector< _S > * __def) const [inline]

Definition at line 146 of file cdat-inline.h.

#### 6.28.2.4 template<class _S> void control_data::assign (const char * __s, _S & __b) const [inline]

Definition at line 134 of file cdat-inline.h.

#### 6.28.2.5 template<class _S> void control_data::assign (const char * __s, const matrix< _S > *& __b) const [inline]

Definition at line 128 of file cdat-inline.h.

#### 6.28.2.6 template<class _S> void control_data::assign (const char * __s, const std::vector< _S > *& __b) const [inline]

Definition at line 121 of file cdat-inline.h.

#### 6.28.2.7 void control_data::assign (const char * __s, std::string & __b, const char * __def) const [inline]

Definition at line 139 of file cdat-inline.h.

#### 6.28.2.8 template<class _S> void control_data::assign (const std::string & __s, _S & __b, _S __def) const [inline]

Definition at line 114 of file cdat-inline.h.

#### 6.28.2.9 template<class _S> void control_data::assign (const std::string & __s, const matrix< _S > *& __b, const matrix< _S > * __def) const [inline]

Definition at line 107 of file cdat-inline.h.

#### 6.28.2.10 template<class _S> void control_data::assign (const std::string & __s, const std::vector< _S > *& __b, const std::vector< _S > * __def) const [inline]

Definition at line 99 of file cdat-inline.h.

#### 6.28.2.11 void control_data::assign (const std::string & __s, std::string & __b, const char * __def) const [inline]

Definition at line 92 of file cdat-inline.h.

#### 6.28.2.12 template<class _S> void control_data::assign (const std::string & __s, _S & __b) const [inline]

Definition at line 82 of file cdat-inline.h.

#### 6.28.2.13 template<class _S> void control_data::assign (const std::string & __s, const matrix< _S > *& __b) const [inline]

Definition at line 70 of file cdat-inline.h.

#### 6.28.2.14 template<class _S> void control_data::assign (const std::string & __s, const std::vector< _S > *& __b) const [inline]

Definition at line 58 of file cdat-inline.h.

#### 6.28.2.15 template<class _S> void control_data::assign_i (const char * __s, int i, _S & __b, _S __def) const [inline]

Definition at line 273 of file cdat-inline.h.

#### 6.28.2.16 template<class _S> void control_data::assign_i (const char * __s, int i, const matrix< _S > *& __b, const matrix< _S > * __def) const [inline]

Definition at line 265 of file cdat-inline.h.

#### 6.28.2.17 template<class _S> void control_data::assign_i (const char * __s, int i, const std::vector< _S > *& __b, const std::vector< _S > * __def) const [inline]

Definition at line 257 of file cdat-inline.h.

**6.28.2.18 void control_data::assign_i (const char * __s, int i, std::string & __b, const char * __def) const** `[inline]`

Definition at line 250 of file cdat-inline.h.

**6.28.2.19 template<class _S> void control_data::assign_i (const char * __s, int i, _S & __b) const** `[inline]`

Definition at line 245 of file cdat-inline.h.

**6.28.2.20 template<class _S> void control_data::assign_i (const char * __s, int i, const matrix< _S >*& __b) const** `[inline]`

Definition at line 238 of file cdat-inline.h.

**6.28.2.21 template<class _S> void control_data::assign_i (const char * __s, int i, const std::vector< _S >*& __b) const** `[inline]`

Definition at line 231 of file cdat-inline.h.

**6.28.2.22 template<class _S> void control_data::assign_i (const std::string & __s, int i, _S & __b, _S __def) const** `[inline]`

Definition at line 224 of file cdat-inline.h.

**6.28.2.23 template<class _S> void control_data::assign_i (const std::string & __s, int i, const matrix< _S >*& __b, const matrix< _S >* __def) const** `[inline]`

Definition at line 216 of file cdat-inline.h.

**6.28.2.24 template<class _S> void control_data::assign_i (const std::string & __s, int i, const std::vector< _S >*& __b, const std::vector< _S >* __def) const** `[inline]`

Definition at line 208 of file cdat-inline.h.

**6.28.2.25 void control_data::assign_i (const std::string & __s, int i, std::string & __b, const char * __def) const** `[inline]`

Definition at line 201 of file cdat-inline.h.

**6.28.2.26 template<class _S> void control_data::assign_i (const std::string & __s, int i, _S & __b) const** `[inline]`

Definition at line 191 of file cdat-inline.h.

**6.28.2.27 template<class _S> void control_data::assign_i (const std::string & __s, int i, const matrix< _S >*& __b) const** `[inline]`

Definition at line 179 of file cdat-inline.h.

#### 6.28.2.28 template<class _S> void control_data::assign_i (const std::string & __s, int i, const std::vector< _S > *& __b) const [inline]

Definition at line 167 of file cdat-inline.h.

#### 6.28.2.29 bool control_data::check_service (const char * __n) const [inline]

Definition at line 52 of file cdat-inline.h.

#### 6.28.2.30 bool control_data::check_service (const std::string & __n) const [inline]

Definition at line 47 of file cdat-inline.h.

#### 6.28.2.31 bool datamap::defd (const char * __cp, int i) const [inline, inherited]

Definition at line 121 of file datamap-inline.h.

#### 6.28.2.32 bool datamap::defd (const std::string & __s, int i) const [inline, inherited]

Definition at line 116 of file datamap-inline.h.

#### 6.28.2.33 bool datamap::defd (const char * __cp) const [inline, inherited]

Definition at line 111 of file datamap-inline.h.

#### 6.28.2.34 bool datamap::defd (const std::string & __s) const [inline, inherited]

Definition at line 106 of file datamap-inline.h.

#### 6.28.2.35 const additional_info_u& control_data::get (const char * __cp, int i) const [inline]

Definition at line 87 of file control_data.h.

#### 6.28.2.36 const additional_info_u& control_data::get (const std::string & __s, int i) const [inline]

Definition at line 85 of file control_data.h.

#### 6.28.2.37 const additional_info_u& control_data::get (const char * __cp) const [inline]

Definition at line 83 of file control_data.h.

#### 6.28.2.38 const additional_info_u& control_data::get (const std::string & __s) const [inline]

Definition at line 81 of file control_data.h.

#### 6.28.2.39 bool control_data::is_set (const char * __cp, int i) const [inline]

Definition at line 98 of file control_data.h.

#### 6.28.2.40 bool control_data::is_set (const std::string & __s, int i) const [inline]

Definition at line 97 of file control_data.h.

#### 6.28.2.41 bool control_data::is_set (const char * __cp) const [inline]

Definition at line 96 of file control_data.h.

#### 6.28.2.42 bool control_data::is_set (const std::string & __s) const [inline]

Definition at line 95 of file control_data.h.

#### 6.28.2.43 control_data& control_data::operator= (const info_contents & __i) [inline]

Definition at line 58 of file control_data.h.

#### 6.28.2.44 control_data& control_data::operator= (const control_data & __c) [inline]

Definition at line 51 of file control_data.h.

#### 6.28.2.45 void datamap::remove (const char * __cp, int i) [inline, inherited]

Definition at line 73 of file datamap-inline.h.

#### 6.28.2.46 void datamap::remove (const std::string & __s, int i) [inline, inherited]

Definition at line 67 of file datamap-inline.h.

#### 6.28.2.47 void datamap::remove (const char * __cp) [inline, inherited]

Definition at line 62 of file datamap-inline.h.

#### 6.28.2.48 void datamap::remove (const std::string & __s) [inline, inherited]

Definition at line 54 of file datamap-inline.h.

#### 6.28.2.49 bool datamap::retrieve (const char * __s, const matrix< interval > *& __b, const matrix< interval > * __def) const [inline, inherited]

Definition at line 597 of file datamap-inline.h.

#### 6.28.2.50 bool datamap::retrieve (const char * __s, const matrix< interval > *& __b) const [inline, inherited]

Definition at line 592 of file datamap-inline.h.

#### 6.28.2.51 bool datamap::retrieve (const char * __s, const matrix< int > *& __b, const matrix< int > * __def) const [inline, inherited]

Definition at line 586 of file datamap-inline.h.

#### 6.28.2.52 bool datamap::retrieve (const char * __s, const matrix< int > *& __b) const [inline, inherited]

Definition at line 581 of file datamap-inline.h.

**6.28.2.53 bool datamap::retrieve (const char * __s, const matrix< double > *& __b, const matrix< double > * __def) const** `[inline, inherited]`

Definition at line 575 of file datamap-inline.h.

**6.28.2.54 bool datamap::retrieve (const char * __s, const matrix< double > *& __b) const** `[inline, inherited]`

Definition at line 570 of file datamap-inline.h.

**6.28.2.55 bool datamap::retrieve (const char * __s, const std::vector< interval > *& __b, const std::vector< interval > * __def) const** `[inline, inherited]`

Definition at line 565 of file datamap-inline.h.

**6.28.2.56 bool datamap::retrieve (const char * __s, const std::vector< interval > *& __b) const** `[inline, inherited]`

Definition at line 560 of file datamap-inline.h.

**6.28.2.57 bool datamap::retrieve (const char * __s, const std::vector< double > *& __b, const std::vector< double > * __def) const** `[inline, inherited]`

Definition at line 555 of file datamap-inline.h.

**6.28.2.58 bool datamap::retrieve (const char * __s, const std::vector< double > *& __b) const** `[inline, inherited]`

Definition at line 550 of file datamap-inline.h.

**6.28.2.59 bool datamap::retrieve (const char * __s, const std::vector< int > *& __b, const std::vector< int > * __def) const** `[inline, inherited]`

Definition at line 535 of file datamap-inline.h.

**6.28.2.60 bool datamap::retrieve (const char * __s, const std::vector< int > *& __b) const** `[inline, inherited]`

Definition at line 530 of file datamap-inline.h.

**6.28.2.61 bool datamap::retrieve (const char * __s, const std::vector< unsigned int > *& __b, const std::vector< unsigned int > * __def) const** `[inline, inherited]`

Definition at line 545 of file datamap-inline.h.

**6.28.2.62 bool datamap::retrieve (const char * __s, const std::vector< unsigned int > *& __b) const** `[inline, inherited]`

Definition at line 540 of file datamap-inline.h.

**6.28.2.63 bool datamap::retrieve (const char * __s, const std::vector< bool > *& __b, const std::vector< bool > * __def) const** `[inline, inherited]`

Definition at line 525 of file datamap-inline.h.

**6.28.2.64 bool datamap::retrieve (const char * __s, const std::vector< bool > *& __b) const** `[inline, inherited]`

Definition at line 520 of file datamap-inline.h.

**6.28.2.65 bool datamap::retrieve (const char * __s, std::string & __b, const std::string & __def) const** `[inline, inherited]`

Definition at line 515 of file datamap-inline.h.

**6.28.2.66 bool datamap::retrieve (const char * __s, std::string & __is) const** `[inline, inherited]`

Definition at line 510 of file datamap-inline.h.

**6.28.2.67 bool datamap::retrieve (const char * __s, interval & __b, const interval & __def) const** `[inline, inherited]`

Definition at line 505 of file datamap-inline.h.

**6.28.2.68 bool datamap::retrieve (const char * __s, interval & __b) const** `[inline, inherited]`

Definition at line 500 of file datamap-inline.h.

**6.28.2.69 bool datamap::retrieve (const char * __s, double & __d, double __def) const** `[inline, inherited]`

Definition at line 495 of file datamap-inline.h.

**6.28.2.70 bool datamap::retrieve (const char * __s, double & __d) const** `[inline, inherited]`

Definition at line 490 of file datamap-inline.h.

**6.28.2.71 bool datamap::retrieve (const char * __s, unsigned int & __d, unsigned int __def) const** `[inline, inherited]`

Definition at line 485 of file datamap-inline.h.

**6.28.2.72 bool datamap::retrieve (const char * __s, unsigned int & __d) const** `[inline, inherited]`

Definition at line 480 of file datamap-inline.h.

**6.28.2.73 bool datamap::retrieve (const char * __s, int & __d, int __def) const** `[inline, inherited]`

Definition at line 475 of file datamap-inline.h.

**6.28.2.74 bool datamap::retrieve (const char * __s, int & __d) const** `[inline, inherited]`

Definition at line 470 of file datamap-inline.h.

**6.28.2.75 bool datamap::retrieve (const char * __s, bool & __b, bool __def) const** `[inline, inherited]`

Definition at line 465 of file datamap-inline.h.

**6.28.2.76 bool datamap::retrieve (const char * __s, bool & __b) const** `[inline, inherited]`

Definition at line 460 of file datamap-inline.h.

**6.28.2.77 bool datamap::retrieve (const std::string & __s, const matrix< interval > *& __b, const matrix< interval > * __def) const** `[inline, inherited]`

Definition at line 449 of file datamap-inline.h.

**6.28.2.78 bool datamap::retrieve (const std::string & __s, const matrix< interval > *& __b) const** `[inline, inherited]`

Definition at line 437 of file datamap-inline.h.

**6.28.2.79 bool datamap::retrieve (const std::string & __s, const matrix< int > *& __b, const matrix< int > * __def) const** `[inline, inherited]`

Definition at line 427 of file datamap-inline.h.

**6.28.2.80 bool datamap::retrieve (const std::string & __s, const matrix< int > *& __b) const** `[inline, inherited]`

Definition at line 415 of file datamap-inline.h.

**6.28.2.81 bool datamap::retrieve (const std::string & __s, const matrix< double > *& __b, const matrix< double > * __def) const** `[inline, inherited]`

Definition at line 405 of file datamap-inline.h.

**6.28.2.82 bool datamap::retrieve (const std::string & __s, const matrix< double > *& __b) const** `[inline, inherited]`

Definition at line 393 of file datamap-inline.h.

**6.28.2.83 bool datamap::retrieve (const std::string & __s, const std::vector< interval > *& __b, const std::vector< interval > * __def) const** `[inline, inherited]`

Definition at line 383 of file datamap-inline.h.

**6.28.2.84 bool datamap::retrieve (const std::string & __s, const std::vector< interval > *& __b) const** `[inline, inherited]`

Definition at line 371 of file datamap-inline.h.

**6.28.2.85 bool datamap::retrieve (const std::string & __s, const std::vector< double > *& __b, const std::vector< double > * __def) const** `[inline, inherited]`

Definition at line 361 of file datamap-inline.h.

**6.28.2.86 bool datamap::retrieve (const std::string & __s, const std::vector< double > *& __b) const** `[inline, inherited]`

Definition at line 349 of file datamap-inline.h.

**6.28.2.87 bool datamap::retrieve (const std::string & __s, const std::vector< unsigned int > *& __b, const std::vector< unsigned int > * __def) const** `[inline, inherited]`

Definition at line 339 of file datamap-inline.h.

**6.28.2.88 bool datamap::retrieve (const std::string & __s, const std::vector< unsigned int > *& __b) const** `[inline, inherited]`

Definition at line 327 of file datamap-inline.h.

**6.28.2.89 bool datamap::retrieve (const std::string & __s, const std::vector< int > *& __b, const std::vector< int > * __def) const** `[inline, inherited]`

Definition at line 317 of file datamap-inline.h.

**6.28.2.90 bool datamap::retrieve (const std::string & __s, const std::vector< int > *& __b) const** `[inline, inherited]`

Definition at line 305 of file datamap-inline.h.

**6.28.2.91 bool datamap::retrieve (const std::string & __s, const std::vector< bool > *& __b, const std::vector< bool > * __def) const** `[inline, inherited]`

Definition at line 295 of file datamap-inline.h.

**6.28.2.92 bool datamap::retrieve (const std::string & __s, const std::vector< bool > *& __b) const** `[inline, inherited]`

Definition at line 283 of file datamap-inline.h.

**6.28.2.93 bool datamap::retrieve (const std::string & __s, std::string & __is, const std::string & __def) const** `[inline, inherited]`

Definition at line 273 of file datamap-inline.h.

**6.28.2.94 bool datamap::retrieve (const std::string & _s, std::string & _is) const** `[inline, inherited]`

Definition at line 261 of file datamap-inline.h.

**6.28.2.95 bool datamap::retrieve (const std::string & _s, interval & _b, const interval & _def) const** `[inline, inherited]`

Definition at line 251 of file datamap-inline.h.

**6.28.2.96 bool datamap::retrieve (const std::string & _s, interval & _b) const** `[inline, inherited]`

Definition at line 239 of file datamap-inline.h.

**6.28.2.97 bool datamap::retrieve (const std::string & _s, double & _d, double _def) const** `[inline, inherited]`

Definition at line 229 of file datamap-inline.h.

**6.28.2.98 bool datamap::retrieve (const std::string & _s, double & _d) const** `[inline, inherited]`

Definition at line 217 of file datamap-inline.h.

**6.28.2.99 bool datamap::retrieve (const std::string & _s, unsigned int & _d, unsigned int _def) const** `[inline, inherited]`

Definition at line 207 of file datamap-inline.h.

**6.28.2.100 bool datamap::retrieve (const std::string & _s, unsigned int & _d) const** `[inline, inherited]`

Definition at line 195 of file datamap-inline.h.

**6.28.2.101 bool datamap::retrieve (const std::string & _s, int & _d, int _def) const** `[inline, inherited]`

Definition at line 185 of file datamap-inline.h.

**6.28.2.102 bool datamap::retrieve (const std::string & _s, int & _d) const** `[inline, inherited]`

Definition at line 173 of file datamap-inline.h.

**6.28.2.103 bool datamap::retrieve (const std::string & _s, bool & _b, bool _def) const** `[inline, inherited]`

Definition at line 163 of file datamap-inline.h.

**6.28.2.104 bool datamap::retrieve (const std::string & __s, bool & __b) const** `[inline, inherited]`

Definition at line 146 of file datamap-inline.h.

**6.28.2.105 bool datamap::retrieve_i (const char * __s, int *i*, const matrix< interval > *& __b, const matrix< interval > * __def) const** `[inline, inherited]`

Definition at line 1021 of file datamap-inline.h.

**6.28.2.106 bool datamap::retrieve_i (const char * __s, int *i*, const matrix< interval > *& __b) const** `[inline, inherited]`

Definition at line 1016 of file datamap-inline.h.

**6.28.2.107 bool datamap::retrieve_i (const char * __s, int *i*, const matrix< int > *& __b, const matrix< int > * __def) const** `[inline, inherited]`

Definition at line 1010 of file datamap-inline.h.

**6.28.2.108 bool datamap::retrieve_i (const char * __s, int *i*, const matrix< int > *& __b) const** `[inline, inherited]`

Definition at line 1005 of file datamap-inline.h.

**6.28.2.109 bool datamap::retrieve_i (const char * __s, int *i*, const matrix< double > *& __b, const matrix< double > * __def) const** `[inline, inherited]`

Definition at line 999 of file datamap-inline.h.

**6.28.2.110 bool datamap::retrieve_i (const char * __s, int *i*, const matrix< double > *& __b) const** `[inline, inherited]`

Definition at line 994 of file datamap-inline.h.

**6.28.2.111 bool datamap::retrieve_i (const char * __s, int *i*, const std::vector< interval > *& __b, const std::vector< interval > * __def) const** `[inline, inherited]`

Definition at line 989 of file datamap-inline.h.

**6.28.2.112 bool datamap::retrieve_i (const char * __s, int *i*, const std::vector< interval > *& __b) const** `[inline, inherited]`

Definition at line 984 of file datamap-inline.h.

**6.28.2.113 bool datamap::retrieve_i (const char * __s, int *i*, const std::vector< double > *& __b, const std::vector< double > * __def) const** `[inline, inherited]`

Definition at line 979 of file datamap-inline.h.

**6.28.2.114 bool datamap::retrieve_i (const char * __s, int i, const std::vector< double > *& __b) const** `[inline, inherited]`

Definition at line 974 of file datamap-inline.h.

**6.28.2.115 bool datamap::retrieve_i (const char * __s, int i, const std::vector< unsigned int > *& __b, const std::vector< unsigned int > * __def) const** `[inline, inherited]`

Definition at line 969 of file datamap-inline.h.

**6.28.2.116 bool datamap::retrieve_i (const char * __s, int i, const std::vector< unsigned int > *& __b) const** `[inline, inherited]`

Definition at line 964 of file datamap-inline.h.

**6.28.2.117 bool datamap::retrieve_i (const char * __s, int i, const std::vector< int > *& __b, const std::vector< int > * __def) const** `[inline, inherited]`

Definition at line 959 of file datamap-inline.h.

**6.28.2.118 bool datamap::retrieve_i (const char * __s, int i, const std::vector< int > *& __b) const** `[inline, inherited]`

Definition at line 954 of file datamap-inline.h.

**6.28.2.119 bool datamap::retrieve_i (const char * __s, int i, const std::vector< bool > *& __b, const std::vector< bool > * __def) const** `[inline, inherited]`

Definition at line 949 of file datamap-inline.h.

**6.28.2.120 bool datamap::retrieve_i (const char * __s, int i, const std::vector< bool > *& __b) const** `[inline, inherited]`

Definition at line 944 of file datamap-inline.h.

**6.28.2.121 bool datamap::retrieve_i (const char * __s, int i, std::string & __b, const std::string & __def) const** `[inline, inherited]`

Definition at line 939 of file datamap-inline.h.

**6.28.2.122 bool datamap::retrieve_i (const char * __s, int i, std::string & __is) const** `[inline, inherited]`

Definition at line 934 of file datamap-inline.h.

**6.28.2.123 bool datamap::retrieve_i (const char * __s, int i, interval & __b, const interval & __def) const** `[inline, inherited]`

Definition at line 929 of file datamap-inline.h.

**6.28.2.124 bool datamap::retrieve_i (const char * __s, int i, interval & __b) const** `[inline, inherited]`

Definition at line 924 of file datamap-inline.h.

**6.28.2.125 bool datamap::retrieve_i (const char * __s, int i, double & __d, double __def) const** `[inline, inherited]`

Definition at line 919 of file datamap-inline.h.

**6.28.2.126 bool datamap::retrieve_i (const char * __s, int i, double & __d) const** `[inline, inherited]`

Definition at line 914 of file datamap-inline.h.

**6.28.2.127 bool datamap::retrieve_i (const char * __s, int i, unsigned int & __d, unsigned int __def) const** `[inline, inherited]`

Definition at line 909 of file datamap-inline.h.

**6.28.2.128 bool datamap::retrieve_i (const char * __s, int i, unsigned int & __d) const** `[inline, inherited]`

Definition at line 904 of file datamap-inline.h.

**6.28.2.129 bool datamap::retrieve_i (const char * __s, int i, int & __d, int __def) const** `[inline, inherited]`

Definition at line 899 of file datamap-inline.h.

**6.28.2.130 bool datamap::retrieve_i (const char * __s, int i, int & __d) const** `[inline, inherited]`

Definition at line 894 of file datamap-inline.h.

**6.28.2.131 bool datamap::retrieve_i (const char * __s, int i, bool & __b, bool __def) const** `[inline, inherited]`

Definition at line 889 of file datamap-inline.h.

**6.28.2.132 bool datamap::retrieve_i (const char * __s, int i, bool & __b) const** `[inline, inherited]`

Definition at line 884 of file datamap-inline.h.

**6.28.2.133 bool datamap::retrieve_i (const std::string & __s, int i, const matrix< interval > *& __b, const matrix< interval > * __def) const** `[inline, inherited]`

Definition at line 874 of file datamap-inline.h.

**6.28.2.134 bool datamap::retrieve_i (const std::string & __s, int *i*, const matrix< interval > *& __b) const** `[inline, inherited]`

Definition at line 863 of file datamap-inline.h.

**6.28.2.135 bool datamap::retrieve_i (const std::string & __s, int *i*, const matrix< int > *& __b, const matrix< int > * __def) const** `[inline, inherited]`

Definition at line 854 of file datamap-inline.h.

**6.28.2.136 bool datamap::retrieve_i (const std::string & __s, int *i*, const matrix< int > *& __b) const** `[inline, inherited]`

Definition at line 843 of file datamap-inline.h.

**6.28.2.137 bool datamap::retrieve_i (const std::string & __s, int *i*, const matrix< double > *& __b, const matrix< double > * __def) const** `[inline, inherited]`

Definition at line 834 of file datamap-inline.h.

**6.28.2.138 bool datamap::retrieve_i (const std::string & __s, int *i*, const matrix< double > *& __b) const** `[inline, inherited]`

Definition at line 823 of file datamap-inline.h.

**6.28.2.139 bool datamap::retrieve_i (const std::string & __s, int *i*, const std::vector< interval > *& __b, const std::vector< interval > * __def) const** `[inline, inherited]`

Definition at line 814 of file datamap-inline.h.

**6.28.2.140 bool datamap::retrieve_i (const std::string & __s, int *i*, const std::vector< interval > *& __b) const** `[inline, inherited]`

Definition at line 803 of file datamap-inline.h.

**6.28.2.141 bool datamap::retrieve_i (const std::string & __s, int *i*, const std::vector< double > *& __b, const std::vector< double > * __def) const** `[inline, inherited]`

Definition at line 794 of file datamap-inline.h.

**6.28.2.142 bool datamap::retrieve_i (const std::string & __s, int *i*, const std::vector< double > *& __b) const** `[inline, inherited]`

Definition at line 783 of file datamap-inline.h.

**6.28.2.143 bool datamap::retrieve_i (const std::string & __s, int *i*, const std::vector< unsigned int > *& __b, const std::vector< unsigned int > * __def) const** `[inline, inherited]`

Definition at line 774 of file datamap-inline.h.

**6.28.2.144 bool datamap::retrieve_i (const std::string & __s, int *i*, const std::vector< unsigned int > *& __b) const** `[inline, inherited]`

Definition at line 763 of file datamap-inline.h.

**6.28.2.145 bool datamap::retrieve_i (const std::string & __s, int *i*, const std::vector< int > *& __b, const std::vector< int > * __def) const** `[inline, inherited]`

Definition at line 754 of file datamap-inline.h.

**6.28.2.146 bool datamap::retrieve_i (const std::string & __s, int *i*, const std::vector< int > *& __b) const** `[inline, inherited]`

Definition at line 743 of file datamap-inline.h.

**6.28.2.147 bool datamap::retrieve_i (const std::string & __s, int *i*, const std::vector< bool > *& __b, const std::vector< bool > * __def) const** `[inline, inherited]`

Definition at line 734 of file datamap-inline.h.

**6.28.2.148 bool datamap::retrieve_i (const std::string & __s, int *i*, const std::vector< bool > *& __b) const** `[inline, inherited]`

Definition at line 723 of file datamap-inline.h.

**6.28.2.149 bool datamap::retrieve_i (const std::string & __s, int *i*, std::string & __is, const std::string & __def) const** `[inline, inherited]`

Definition at line 714 of file datamap-inline.h.

**6.28.2.150 bool datamap::retrieve_i (const std::string & __s, int *i*, std::string & __is) const** `[inline, inherited]`

Definition at line 703 of file datamap-inline.h.

**6.28.2.151 bool datamap::retrieve_i (const std::string & __s, int *i*, interval & __b, const interval & __def) const** `[inline, inherited]`

Definition at line 694 of file datamap-inline.h.

**6.28.2.152 bool datamap::retrieve_i (const std::string & __s, int *i*, interval & __b) const** `[inline, inherited]`

Definition at line 683 of file datamap-inline.h.

**6.28.2.153 bool datamap::retrieve_i (const std::string & __s, int *i*, double & __d, double __def) const** `[inline, inherited]`

Definition at line 674 of file datamap-inline.h.

**6.28.2.154 bool datamap::retrieve_i (const std::string & __s, int *i*, double & __d) const** `[inline, inherited]`

Definition at line 663 of file datamap-inline.h.

**6.28.2.155 bool datamap::retrieve_i (const std::string & __s, int *i*, unsigned int & __d, unsigned int __def) const** `[inline, inherited]`

Definition at line 654 of file datamap-inline.h.

**6.28.2.156 bool datamap::retrieve_i (const std::string & __s, int *i*, unsigned int & __d) const** `[inline, inherited]`

Definition at line 643 of file datamap-inline.h.

**6.28.2.157 bool datamap::retrieve_i (const std::string & __s, int *i*, int & __d, int __def) const** `[inline, inherited]`

Definition at line 634 of file datamap-inline.h.

**6.28.2.158 bool datamap::retrieve_i (const std::string & __s, int *i*, int & __d) const** `[inline, inherited]`

Definition at line 623 of file datamap-inline.h.

**6.28.2.159 bool datamap::retrieve_i (const std::string & __s, int *i*, bool & __b, bool __def) const** `[inline, inherited]`

Definition at line 614 of file datamap-inline.h.

**6.28.2.160 bool datamap::retrieve_i (const std::string & __s, int *i*, bool & __b) const** `[inline, inherited]`

Definition at line 603 of file datamap-inline.h.

**6.28.2.161 const std::string & control_data::service ()** `[inline]`

Definition at line 37 of file cdat-inline.h.

**6.28.2.162 void control_data::service (const char * __s)** `[inline]`

Definition at line 32 of file cdat-inline.h.

**6.28.2.163 void control_data::service (const std::string & __s)** `[inline]`

Definition at line 30 of file cdat-inline.h.

**6.28.2.164 void control_data::set (const char * __cp, int *i*, const additional_info_u & __h)** `[inline]`

Definition at line 78 of file control_data.h.

**6.28.2.165 void control_data::set (const std::string & __s, int i, const additional_info_u & __h) [inline]**

Definition at line 76 of file control_data.h.

**6.28.2.166 void control_data::set (const char * __cp, const additional_info_u & __h) [inline]**

Definition at line 74 of file control_data.h.

**6.28.2.167 void control_data::set (const std::string & __s, const additional_info_u & __h) [inline]**

Definition at line 72 of file control_data.h.

**6.28.2.168 void control_data::set (const info_contents & __i) [inline]**

Definition at line 71 of file control_data.h.

**6.28.2.169 const additional_info_u & datamap::sfind (const char * __cp, int i) const [inline, inherited]**

Definition at line 101 of file datamap-inline.h.

**6.28.2.170 const additional_info_u & datamap::sfind (const std::string & __s, int i) const [inline, inherited]**

Definition at line 92 of file datamap-inline.h.

**6.28.2.171 const additional_info_u & datamap::sfind (const char * __cp) const [inline, inherited]**

Definition at line 87 of file datamap-inline.h.

**6.28.2.172 const additional_info_u & datamap::sfind (const std::string & __s) const [inline, inherited]**

Definition at line 78 of file datamap-inline.h.

**6.28.2.173 bool datamap::sinsert (const char * __cp, int i, const additional_info_u & __h, bool replace) [inline, inherited]**

Definition at line 47 of file datamap-inline.h.

**6.28.2.174 bool datamap::sinsert (const std::string & __s, int i, const additional_info_u & __h, bool replace) [inline, inherited]**

Definition at line 39 of file datamap-inline.h.

**6.28.2.175 bool datamap::sinsert (const char * __cp, const additional_info_u & __h, bool replace) [inline, inherited]**

Definition at line 33 of file datamap-inline.h.

#### 6.28.2.176 bool datamap::sinsert (const std::string & __s, const additional_info_u & __h, bool *replace*) [inherited]

Definition at line 29 of file datamap.cc.

#### 6.28.2.177 void control_data::unset (const char * __cp, int *i*) [inline]

Definition at line 93 of file control_data.h.

#### 6.28.2.178 void control_data::unset (const std::string & __s, int *i*) [inline]

Definition at line 92 of file control_data.h.

#### 6.28.2.179 void control_data::unset (const char * __cp) [inline]

Definition at line 91 of file control_data.h.

#### 6.28.2.180 void control_data::unset (const std::string & __s) [inline]

Definition at line 90 of file control_data.h.

#### 6.28.2.181 bool datamap::which (const char * __cp, std::vector< int > & __idx) const [inline, inherited]

Definition at line 140 of file datamap-inline.h.

#### 6.28.2.182 bool datamap::which (const std::string & __s, std::vector< int > & __idx) const [inline, inherited]

Definition at line 126 of file datamap-inline.h.

#### 6.28.2.183 bool control_data::which_set (const char * __cp, std::vector< int > & __idx) const [inline]

Definition at line 102 of file control_data.h.

#### 6.28.2.184 bool control_data::which_set (const std::string & __s, std::vector< int > & __idx) const [inline]

Definition at line 100 of file control_data.h.

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

- control_data.h
- cdat-inline.h

## 6.29 convex_e Class Reference

`#include <semantics.h>`

### Public Methods

- convex_e ()
- convex_e (const convex_info &__e, uint16_t __t)
- convex_e (const convex_e &__e)
- convex_e (uint16_t __e)
- ~convex_e ()
- const convex_info & i () const
- uint16_t t () const
- convex_e operator- () const
- convex_e & operator= (const convex_e &__c)
- convex_e & operator= (const convex_info &__i)
- convex_e & operator= (unsigned int __t)
- convex_e & operator= (uint16_t __t)
- void read (char *c)
- void merge (const convex_e &__c)

### Friends

- bool operator== (const convex_e &__c, const convex_e &__d)
- bool operator== (const convex_e &__c, const convex_info &__d)
- bool operator== (const convex_info &__c, const convex_e &__d)
- bool operator!= (const convex_e &__c, const convex_e &__d)
- bool operator!= (const convex_e &__c, const convex_info &__d)
- bool operator!= (const convex_info &__c, const convex_e &__d)
- std::ostream & operator<< (std::ostream &o, const convex_e &__s)

### 6.29.1 Constructor & Destructor Documentation

#### 6.29.1.1 convex_e::convex_e () [inline]

Definition at line 78 of file semantics.h.

#### 6.29.1.2 convex_e::convex_e (const convex_info & __e, uint16_t __t) [inline]

Definition at line 79 of file semantics.h.

#### 6.29.1.3 convex_e::convex_e (const convex_e & __e) [inline]

Definition at line 80 of file semantics.h.

#### 6.29.1.4 convex_e::convex_e (uint16_t __e) [inline, explicit]

Definition at line 81 of file semantics.h.

#### 6.29.1.5 convex_e::~convex_e () [inline]

Definition at line 83 of file semantics.h.

### 6.29.2 Member Function Documentation

#### 6.29.2.1 const convex_info& convex_e::i () const [inline]

Definition at line 85 of file semantics.h.

#### 6.29.2.2 void convex_e::merge (const convex_e & __c) [inline]

Definition at line 127 of file semantics.h.

#### 6.29.2.3 convex_e convex_e::operator- () const [inline]

Definition at line 88 of file semantics.h.

#### 6.29.2.4 convex_e& convex_e::operator= (uint16_t __t) [inline]

Definition at line 119 of file semantics.h.

#### 6.29.2.5 convex_e& convex_e::operator= (unsigned int __t) [inline]

Definition at line 113 of file semantics.h.

#### 6.29.2.6 convex_e& convex_e::operator= (const convex_info & __i) [inline]

Definition at line 107 of file semantics.h.

#### 6.29.2.7 convex_e& convex_e::operator= (const convex_e & __c) [inline]

Definition at line 100 of file semantics.h.

#### 6.29.2.8 void convex_e::read (char * c)

Definition at line 29 of file semantics.cc.

#### 6.29.2.9 uint16_t convex_e::t () const [inline]

Definition at line 86 of file semantics.h.

### 6.29.3 Friends And Related Function Documentation

#### 6.29.3.1 bool operator!= (const convex_info & __c, const convex_e & __d) [friend]

Definition at line 177 of file semantics.h.

#### 6.29.3.2 bool operator!= (const convex_e & __c, const convex_info & __d) [friend]

Definition at line 172 of file semantics.h.

#### 6.29.3.3 bool operator!= (const convex_e & __c, const convex_e & __d) [friend]

Definition at line 167 of file semantics.h.

#### 6.29.3.4 std::ostream& operator<< (std::ostream & *o*, const convex_e & *__s*) [friend]

Definition at line 182 of file semantics.h.

#### 6.29.3.5 bool operator== (const convex_info & *__c*, const convex_e & *__d*) [friend]

Definition at line 162 of file semantics.h.

#### 6.29.3.6 bool operator== (const convex_e & *__c*, const convex_info & *__d*) [friend]

Definition at line 157 of file semantics.h.

#### 6.29.3.7 bool operator== (const convex_e & *__c*, const convex_e & *__d*) [friend]

Definition at line 152 of file semantics.h.

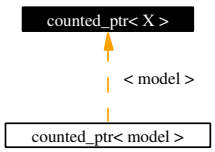
The documentation for this class was generated from the following files:

- semantics.h
- semantics.cc

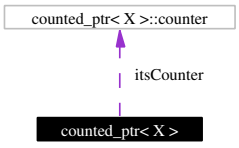
## 6.30 counted_ptr< X > Class Template Reference

`#include <counted_ptr.h>`

Inheritance diagram for counted_ptr< X >:




Collaboration diagram for counted_ptr< X >:




### Public Types

- typedef X element_type

### Public Methods

- counted_ptr (X *p=0)
- ~counted_ptr ()

- counted_ptr (const counted_ptr &r) throw ()
- counted_ptr & operator= (const counted_ptr &r)
- counted_ptr & operator= (element_type *r)
- template<class Y> counted_ptr (const counted_ptr< Y > &r) throw ()
- template<class Y> counted_ptr & operator= (const counted_ptr< Y > &r)
- X & operator * () const throw ()
- X * operator -> () const throw ()
- X * get () const throw ()
- bool unique () const throw ()

**template<class X> class counted_ptr< X >**

### 6.30.1 Member Typedef Documentation

#### 6.30.1.1 template<class X> typedef X counted_ptr< X >::element_type

Definition at line 17 of file counted_ptr.h.

### 6.30.2 Constructor & Destructor Documentation

#### 6.30.2.1 template<class X> counted_ptr< X >::counted_ptr (X * *p* = 0) [inline, explicit]

Definition at line 19 of file counted_ptr.h.

#### 6.30.2.2 template<class X> counted_ptr< X >::~counted_ptr () [inline]

Definition at line 21 of file counted_ptr.h.

#### 6.30.2.3 template<class X> counted_ptr< X >::counted_ptr (const counted_ptr< X > & *r*) throw () [inline]

Definition at line 23 of file counted_ptr.h.

#### 6.30.2.4 template<class X> template<class Y> counted_ptr< X >::counted_ptr (const counted_ptr< Y > & *r*) throw () [inline]

Definition at line 43 of file counted_ptr.h.

### 6.30.3 Member Function Documentation

#### 6.30.3.1 template<class X> X* counted_ptr< X >::get () const throw () [inline]

Definition at line 57 of file counted_ptr.h.

#### 6.30.3.2 template<class X> X& counted_ptr< X >::operator * () const throw () [inline]

Definition at line 55 of file counted_ptr.h.

#### 6.30.3.3 template<class X> X* counted_ptr< X >::operator -> () const throw () [inline]

Definition at line 56 of file counted_ptr.h.

#### 6.30.3.4 template<class X> template<class Y> counted_ptr& counted_ptr< X >::operator= (const counted_ptr< Y > & *r*) [inline]

Definition at line 45 of file counted_ptr.h.

#### 6.30.3.5 template<class X> counted_ptr& counted_ptr< X >::operator= (element_type * *r*) [inline]

Definition at line 33 of file counted_ptr.h.

#### 6.30.3.6 template<class X> counted_ptr& counted_ptr< X >::operator= (const counted_ptr< X > & *r*) [inline]

Definition at line 25 of file counted_ptr.h.

#### 6.30.3.7 template<class X> bool counted_ptr< X >::unique () const throw () [inline]

Definition at line 58 of file counted_ptr.h.

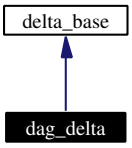
The documentation for this class was generated from the following file:

- counted_ptr.h

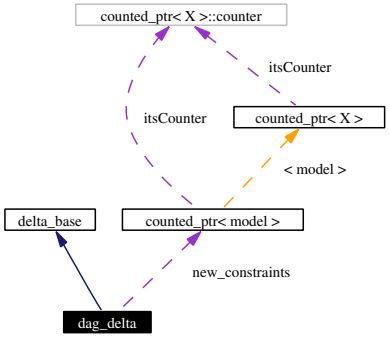
## 6.31 dag_delta Class Reference

`#include <dag_delta.h>`

Inheritance diagram for dag_delta:




Collaboration diagram for dag_delta:



### Public Methods

- dag_delta (const std::string &__a, bool full=false)
- dag_delta (const std::string &__a, model *__nc, bool full=false)
- dag_delta (const dag_delta &__d)
- ~dag_delta ()
- dag_delta * new_copy () const
- void destroy_copy (dag_delta *__d)
- void add_new (model *__m)
- void add_new (model &__m)
- void remove (const walker &_nn)
- void remove (const std::vector< walker > &_nn)
- bool apply (work_node &x, undelta_base *&_u) const
- virtual void destroy_copy (delta_base *__d)
- delta make_delta (const std::string &a)
- const std::string & get_action () const
- virtual void convert (work_node &_x, delta_base *&_d)
- virtual bool apply3 (work_node &_x, const work_node &_y, undelta_base *&_u) const

### Public Attributes

- counted_ptr< model > new_constraints
- std::vector< walker > rm_nodes
- bool is_full_delta

### Protected Attributes

- std::string _action

### Friends

- class dag_undelta

#### 6.31.1 Constructor & Destructor Documentation

##### 6.31.1.1 dag_delta::dag_delta (const std::string & __a, bool full = false) [inline]

Definition at line 128 of file dag_delta.h.

##### 6.31.1.2 dag_delta::dag_delta (const std::string & __a, model * __nc, bool full = false) [inline]

Definition at line 133 of file dag_delta.h.

##### 6.31.1.3 dag_delta::dag_delta (const dag_delta & __d) [inline]

Definition at line 139 of file dag_delta.h.

##### 6.31.1.4 dag_delta::~dag_delta () [inline]

Definition at line 149 of file dag_delta.h.

### 6.31.2 Member Function Documentation

#### 6.31.2.1 void dag_delta::add_new (model & __m) [inline]

Definition at line 155 of file dag_delta.h.

#### 6.31.2.2 void dag_delta::add_new (model * __m) [inline]

Definition at line 154 of file dag_delta.h.

#### 6.31.2.3 bool dag_delta::apply (work_node & x, undelta_base *& _u) const [virtual]

Reimplemented from delta_base.

Definition at line 107 of file dag_delta.cc.

#### 6.31.2.4 bool delta_base::apply3 (work_node & _x, const work_node & _y, undelta_base *& _u) const [inline, virtual, inherited]

Definition at line 63 of file api_delta.h.

#### 6.31.2.5 virtual void delta_base::convert (work_node & _x, delta_base *& _d) [inline, virtual, inherited]

Reimplemented in table_delta.

Definition at line 107 of file api_deltabase.h.

#### 6.31.2.6 virtual void delta_base::destroy_copy (delta_base * __d) [inline, virtual, inherited]

Definition at line 95 of file api_deltabase.h.

#### 6.31.2.7 void dag_delta::destroy_copy (dag_delta * __d) [inline]

Definition at line 152 of file dag_delta.h.

#### 6.31.2.8 const std::string& delta_base::get_action () const [inline, inherited]

Definition at line 105 of file api_deltabase.h.

#### 6.31.2.9 delta delta_base::make_delta (const std::string & a) [inline, inherited]

Definition at line 99 of file api_deltabase.h.

#### 6.31.2.10 dag_delta* dag_delta::new_copy () const [inline, virtual]

Reimplemented from delta_base.

Definition at line 151 of file dag_delta.h.

#### 6.31.2.11 void dag_delta::remove (const std::vector< walker > & _nn) [inline]

Definition at line 158 of file dag_delta.h.

#### 6.31.2.12 void dag_delta::remove (const walker & *_nn*) [inline]

Definition at line 157 of file dag_delta.h.

### 6.31.3 Friends And Related Function Documentation

#### 6.31.3.1 friend class dag_undelta [friend]

Definition at line 198 of file dag_delta.h.

### 6.31.4 Member Data Documentation

#### 6.31.4.1 std::string delta_base::_action [protected, inherited]

Definition at line 86 of file api_deltabase.h.

#### 6.31.4.2 bool dag_delta::is_full_delta

Definition at line 124 of file dag_delta.h.

#### 6.31.4.3 counted_ptr<model> dag_delta::new_constraints

Definition at line 119 of file dag_delta.h.

#### 6.31.4.4 std::vector<walker> dag_delta::rm_nodes

Definition at line 121 of file dag_delta.h.

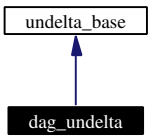
The documentation for this class was generated from the following files:

- dag_delta.h
- dag_delta.cc

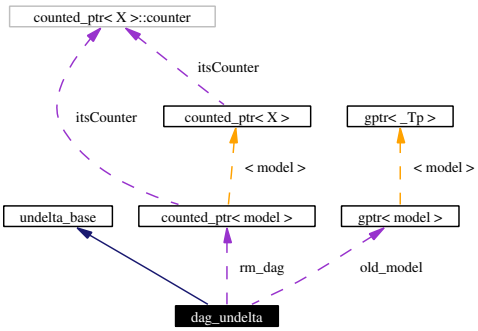
## 6.32 dag_undelta Class Reference

`#include <dag_delta.h>`

Inheritance diagram for dag_undelta:




Collaboration diagram for dag_undelta:




### Public Methods

- dag_undelta (bool full=false)
- dag_undelta (gptr< model > *__nc)
- dag_undelta (const dag_undelta &_du)
- ~dag_undelta ()
- dag_undelta * new_copy () const
- void destroy_copy (dag_undelta *__d)
- bool unapply (work_node &x) const
- virtual void destroy_copy (undelta_base *__d)
- undelta make_undelta ()
- virtual bool unapply3 (work_node &_x, const work_node &_y) const

### Public Attributes

- gptr< model > * old_model
- counted_ptr< model > rm_dag
- std::vector< **model::enhanced_edge** > rm_e
- std::vector< walker > added_nodes
- std::vector< walker > added_constraints
- std::vector< walker > added_ghosts
- std::vector< walker > added_vars
- std::map< unsigned int, interval > bounds_chgd
- bool is_full_undelta

### Friends

- class dag_delta

#### 6.32.1 Constructor & Destructor Documentation

##### 6.32.1.1 dag_undelta::dag_undelta (bool *full* = false) [inline]

Definition at line 72 of file dag_delta.h.

##### 6.32.1.2 dag_undelta::dag_undelta (gptr< model > * *__nc*) [inline]

Definition at line 80 of file dag_delta.h.

#### 6.32.1.3 dag_undelta::dag_undelta (const dag_undelta & *_du*) [inline]

Definition at line 88 of file dag_delta.h.

#### 6.32.1.4 dag_undelta::~dag_undelta () [inline]

Definition at line 103 of file dag_delta.h.

### 6.32.2 Member Function Documentation

#### 6.32.2.1 virtual void undelta_base::destroy_copy (undelta_base * *__d*) [inline, virtual, inherited]

Definition at line 146 of file api_deltabase.h.

#### 6.32.2.2 void dag_undelta::destroy_copy (dag_undelta * *__d*) [inline]

Definition at line 106 of file dag_delta.h.

#### 6.32.2.3 undelta undelta_base::make_undelta () [inline, inherited]

Definition at line 150 of file api_deltabase.h.

#### 6.32.2.4 dag_undelta* dag_undelta::new_copy () const [inline, virtual]

Reimplemented from undelta_base.

Definition at line 105 of file dag_delta.h.

#### 6.32.2.5 bool dag_undelta::unapply (work_node & *x*) const [virtual]

Reimplemented from undelta_base.

Definition at line 259 of file dag_delta.cc.

#### 6.32.2.6 bool undelta_base::unapply3 (work_node & *_x*, const work_node & *_y*) const [inline, virtual, inherited]

Definition at line 69 of file api_delta.h.

### 6.32.3 Friends And Related Function Documentation

#### 6.32.3.1 friend class dag_delta [friend]

Definition at line 110 of file dag_delta.h.

### 6.32.4 Member Data Documentation

#### 6.32.4.1 std::vector<walker> dag_undelta::added_constraints

Definition at line 44 of file dag_delta.h.

#### 6.32.4.2 std::vector<walker> dag_undelta::added_ghosts

Definition at line 45 of file dag_delta.h.

#### 6.32.4.3 std::vector<walker> dag_undelta::added_nodes

Definition at line 43 of file dag_delta.h.

#### 6.32.4.4 std::vector<walker> dag_undelta::added_vars

Definition at line 46 of file dag_delta.h.

#### 6.32.4.5 std::map<unsigned int, interval> dag_undelta::bounds_chgd

Definition at line 48 of file dag_delta.h.

#### 6.32.4.6 bool dag_undelta::is_full_undelta

Definition at line 50 of file dag_delta.h.

#### 6.32.4.7 gptr<model>* dag_undelta::old_model

Definition at line 38 of file dag_delta.h.

#### 6.32.4.8 counted_ptr<model> dag_undelta::rm_dag

Definition at line 40 of file dag_delta.h.

#### 6.32.4.9 std::vector<model::enhanced_edge> dag_undelta::rm_e

Definition at line 41 of file dag_delta.h.

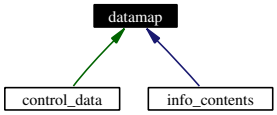
The documentation for this class was generated from the following files:

- dag_delta.h
- dag_delta.cc

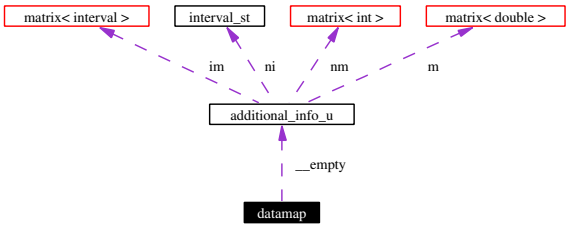
## 6.33 datamap Class Reference

`#include <datamap.h>`

Inheritance diagram for datamap:




Collaboration diagram for datamap:




### Public Methods

- datamap ()
- datamap (const std::string &__n, const additional_info_u &__v)
- datamap (const char *__n, const additional_info_u &__v)
- datamap (const datamap &__c)
- virtual ~datamap ()
- bool sinsert (const std::string &__s, const additional_info_u &__h, bool replace)
- bool sinsert (const char *__cp, const additional_info_u &__h, bool replace)
- bool sinsert (const std::string &__s, int i, const additional_info_u &__h, bool replace)
- bool sinsert (const char *__cp, int i, const additional_info_u &__h, bool replace)
- const additional_info_u & sfind (const std::string &__s) const
- const additional_info_u & sfind (const char *__cp) const
- const additional_info_u & sfind (const std::string &__s, int i) const
- const additional_info_u & sfind (const char *__cp, int i) const
- void remove (const std::string &__s)
- void remove (const char *__cp)
- void remove (const std::string &__s, int i)
- void remove (const char *__cp, int i)
- bool defd (const std::string &__s) const
- bool defd (const char *__cp) const
- bool defd (const std::string &__s, int i) const
- bool defd (const char *__cp, int i) const
- bool which (const std::string &__s, std::vector< int > &__idx) const
- bool which (const char *__cp, std::vector< int > &__idx) const
- bool retrieve (const std::string &__s, bool &__b) const
- bool retrieve (const std::string &__s, bool &__b, bool __def) const
- bool retrieve (const std::string &__s, int &__d) const
- bool retrieve (const std::string &__s, int &__d, int __def) const
- bool retrieve (const std::string &__s, unsigned int &__d) const
- bool retrieve (const std::string &__s, unsigned int &__d, unsigned int __def) const
- bool retrieve (const std::string &__s, double &__d) const
- bool retrieve (const std::string &__s, double &__d, double __def) const
- bool retrieve (const std::string &__s, interval &__b) const
- bool retrieve (const std::string &__s, interval &__b, const interval &__def) const
- bool retrieve (const std::string &__s, std::string &__is) const
- bool retrieve (const std::string &__s, std::string &__is, const std::string &__def) const
- bool retrieve (const std::string &__s, const std::vector< bool > *&__b) const
- bool retrieve (const std::string &__s, const std::vector< bool > *&__b, const std::vector< bool > *__def) const
- bool retrieve (const std::string &__s, const std::vector< int > *&__b) const

- bool retrieve (const std::string &__s, const std::vector< int > *&__b, const std::vector< int > *__def) const
- bool retrieve (const std::string &__s, const std::vector< unsigned int > *&__b) const
- bool retrieve (const std::string &__s, const std::vector< unsigned int > *&__b, const std::vector< unsigned int > *__def) const
- bool retrieve (const std::string &__s, const std::vector< double > *&__b) const
- bool retrieve (const std::string &__s, const std::vector< double > *&__b, const std::vector< double > *__def) const
- bool retrieve (const std::string &__s, const std::vector< interval > *&__b) const
- bool retrieve (const std::string &__s, const std::vector< interval > *&__b, const std::vector< interval > *__def) const
- bool retrieve (const std::string &__s, const matrix< double > *&__b) const
- bool retrieve (const std::string &__s, const matrix< double > *&__b, const matrix< double > *__def) const
- bool retrieve (const std::string &__s, const matrix< int > *&__b) const
- bool retrieve (const std::string &__s, const matrix< int > *&__b, const matrix< int > *__def) const
- bool retrieve (const std::string &__s, const matrix< interval > *&__b) const
- bool retrieve (const std::string &__s, const matrix< interval > *&__b, const matrix< interval > *__def) const
- bool retrieve (const char *__s, bool &__b) const
- bool retrieve (const char *__s, bool &__b, bool __def) const
- bool retrieve (const char *__s, int &__d) const
- bool retrieve (const char *__s, int &__d, int __def) const
- bool retrieve (const char *__s, unsigned int &__d) const
- bool retrieve (const char *__s, unsigned int &__d, unsigned int __def) const
- bool retrieve (const char *__s, double &__d) const
- bool retrieve (const char *__s, double &__d, double __def) const
- bool retrieve (const char *__s, interval &__b) const
- bool retrieve (const char *__s, interval &__b, const interval &__def) const
- bool retrieve (const char *__s, std::string &__is) const
- bool retrieve (const char *__s, std::string &__b, const std::string &__def) const
- bool retrieve (const char *__s, const std::vector< bool > *&__b) const
- bool retrieve (const char *__s, const std::vector< bool > *&__b, const std::vector< bool > *__def) const
- bool retrieve (const char *__s, const std::vector< unsigned int > *&__b) const
- bool retrieve (const char *__s, const std::vector< unsigned int > *&__b, const std::vector< unsigned int > *__def) const
- bool retrieve (const char *__s, const std::vector< int > *&__b) const
- bool retrieve (const char *__s, const std::vector< int > *&__b, const std::vector< int > *__def) const
- bool retrieve (const char *__s, const std::vector< double > *&__b) const
- bool retrieve (const char *__s, const std::vector< double > *&__b, const std::vector< double > *__def) const
- bool retrieve (const char *__s, const std::vector< interval > *&__b) const
- bool retrieve (const char *__s, const std::vector< interval > *&__b, const std::vector< interval > *__def) const
- bool retrieve (const char *__s, const matrix< double > *&__b) const
- bool retrieve (const char *__s, const matrix< double > *&__b, const matrix< double > *__def) const
- bool retrieve (const char *__s, const matrix< int > *&__b) const
- bool retrieve (const char *__s, const matrix< int > *&__b, const matrix< int > *__def) const
- bool retrieve (const char *__s, const matrix< interval > *&__b) const

- bool retrieve (const char *__s, const matrix< interval > *&__b, const matrix< interval > *__def) const
- bool retrieve_i (const std::string &__s, int i, bool &__b) const
- bool retrieve_i (const std::string &__s, int i, bool &__b, bool __def) const
- bool retrieve_i (const std::string &__s, int i, int &__d) const
- bool retrieve_i (const std::string &__s, int i, int &__d, int __def) const
- bool retrieve_i (const std::string &__s, int i, unsigned int &__d) const
- bool retrieve_i (const std::string &__s, int i, unsigned int &__d, unsigned int __def) const
- bool retrieve_i (const std::string &__s, int i, double &__d) const
- bool retrieve_i (const std::string &__s, int i, double &__d, double __def) const
- bool retrieve_i (const std::string &__s, int i, interval &__b) const
- bool retrieve_i (const std::string &__s, int i, interval &__b, const interval &__def) const
- bool retrieve_i (const std::string &__s, int i, std::string &__is) const
- bool retrieve_i (const std::string &__s, int i, std::string &__is, const std::string &__def) const
- bool retrieve_i (const std::string &__s, int i, const std::vector< bool > *&__b) const
- bool retrieve_i (const std::string &__s, int i, const std::vector< bool > *&__b, const std::vector< bool > *__def) const
- bool retrieve_i (const std::string &__s, int i, const std::vector< int > *&__b) const
- bool retrieve_i (const std::string &__s, int i, const std::vector< int > *&__b, const std::vector< int > *__def) const
- bool retrieve_i (const std::string &__s, int i, const std::vector< unsigned int > *&__b) const
- bool retrieve_i (const std::string &__s, int i, const std::vector< unsigned int > *&__b, const std::vector< unsigned int > *__def) const
- bool retrieve_i (const std::string &__s, int i, const std::vector< double > *&__b) const
- bool retrieve_i (const std::string &__s, int i, const std::vector< double > *&__b, const std::vector< double > *__def) const
- bool retrieve_i (const std::string &__s, int i, const std::vector< interval > *&__b) const
- bool retrieve_i (const std::string &__s, int i, const std::vector< interval > *&__b, const std::vector< interval > *__def) const
- bool retrieve_i (const std::string &__s, int i, const matrix< double > *&__b) const
- bool retrieve_i (const std::string &__s, int i, const matrix< double > *&__b, const matrix< double > *__def) const
- bool retrieve_i (const std::string &__s, int i, const matrix< int > *&__b) const
- bool retrieve_i (const std::string &__s, int i, const matrix< int > *&__b, const matrix< int > *__def) const
- bool retrieve_i (const std::string &__s, int i, const matrix< interval > *&__b) const
- bool retrieve_i (const std::string &__s, int i, const matrix< interval > *&__b, const matrix< interval > *__def) const
- bool retrieve_i (const char *__s, int i, bool &__b) const
- bool retrieve_i (const char *__s, int i, bool &__b, bool __def) const
- bool retrieve_i (const char *__s, int i, int &__d) const
- bool retrieve_i (const char *__s, int i, int &__d, int __def) const
- bool retrieve_i (const char *__s, int i, unsigned int &__d) const
- bool retrieve_i (const char *__s, int i, unsigned int &__d, unsigned int __def) const
- bool retrieve_i (const char *__s, int i, double &__d) const
- bool retrieve_i (const char *__s, int i, double &__d, double __def) const
- bool retrieve_i (const char *__s, int i, interval &__b) const
- bool retrieve_i (const char *__s, int i, interval &__b, const interval &__def) const
- bool retrieve_i (const char *__s, int i, std::string &__is) const
- bool retrieve_i (const char *__s, int i, std::string &__b, const std::string &__def) const
- bool retrieve_i (const char *__s, int i, const std::vector< bool > *&__b) const

- bool retrieve_i (const char *__s, int i, const std::vector< bool > *&__b, const std::vector< bool > *__def) const
- bool retrieve_i (const char *__s, int i, const std::vector< int > *&__b) const
- bool retrieve_i (const char *__s, int i, const std::vector< int > *&__b, const std::vector< int > *__def) const
- bool retrieve_i (const char *__s, int i, const std::vector< unsigned int > *&__b) const
- bool retrieve_i (const char *__s, int i, const std::vector< unsigned int > *&__b, const std::vector< unsigned int > *__def) const
- bool retrieve_i (const char *__s, int i, const std::vector< double > *&__b) const
- bool retrieve_i (const char *__s, int i, const std::vector< double > *&__b, const std::vector< double > *__def) const
- bool retrieve_i (const char *__s, int i, const std::vector< interval > *&__b) const
- bool retrieve_i (const char *__s, int i, const std::vector< interval > *&__b, const std::vector< interval > *__def) const
- bool retrieve_i (const char *__s, int i, const matrix< double > *&__b) const
- bool retrieve_i (const char *__s, int i, const matrix< double > *&__b, const matrix< double > *__def) const
- bool retrieve_i (const char *__s, int i, const matrix< int > *&__b) const
- bool retrieve_i (const char *__s, int i, const matrix< int > *&__b, const matrix< int > *__def) const
- bool retrieve_i (const char *__s, int i, const matrix< interval > *&__b) const
- bool retrieve_i (const char *__s, int i, const matrix< interval > *&__b, const matrix< interval > *__def) const

### 6.33.1 Constructor & Destructor Documentation

#### 6.33.1.1 datamap::datamap () [inline]

Definition at line 47 of file datamap.h.

#### 6.33.1.2 datamap::datamap (const std::string & *__n*, const additional_info_u & *__v*) [inline]

Definition at line 48 of file datamap.h.

#### 6.33.1.3 datamap::datamap (const char * *__n*, const additional_info_u & *__v*) [inline]

Definition at line 51 of file datamap.h.

#### 6.33.1.4 datamap::datamap (const datamap & *__c*) [inline]

Definition at line 54 of file datamap.h.

#### 6.33.1.5 virtual datamap::~datamap () [inline, virtual]

Definition at line 55 of file datamap.h.

### 6.33.2 Member Function Documentation

#### 6.33.2.1 bool datamap::defd (const char * *__cp*, int *i*) const [inline]

Definition at line 121 of file datamap-inline.h.

#### 6.33.2.2 bool datamap::defd (const std::string & __s, int i) const [inline]

Definition at line 116 of file datamap-inline.h.

#### 6.33.2.3 bool datamap::defd (const char * __cp) const [inline]

Definition at line 111 of file datamap-inline.h.

#### 6.33.2.4 bool datamap::defd (const std::string & __s) const [inline]

Definition at line 106 of file datamap-inline.h.

#### 6.33.2.5 void datamap::remove (const char * __cp, int i) [inline]

Definition at line 73 of file datamap-inline.h.

#### 6.33.2.6 void datamap::remove (const std::string & __s, int i) [inline]

Definition at line 67 of file datamap-inline.h.

#### 6.33.2.7 void datamap::remove (const char * __cp) [inline]

Definition at line 62 of file datamap-inline.h.

#### 6.33.2.8 void datamap::remove (const std::string & __s) [inline]

Definition at line 54 of file datamap-inline.h.

#### 6.33.2.9 bool datamap::retrieve (const char * __s, const matrix< interval > *& __b, const matrix< interval > * __def) const [inline]

Definition at line 597 of file datamap-inline.h.

#### 6.33.2.10 bool datamap::retrieve (const char * __s, const matrix< interval > *& __b) const [inline]

Definition at line 592 of file datamap-inline.h.

#### 6.33.2.11 bool datamap::retrieve (const char * __s, const matrix< int > *& __b, const matrix< int > * __def) const [inline]

Definition at line 586 of file datamap-inline.h.

#### 6.33.2.12 bool datamap::retrieve (const char * __s, const matrix< int > *& __b) const [inline]

Definition at line 581 of file datamap-inline.h.

#### 6.33.2.13 bool datamap::retrieve (const char * __s, const matrix< double > *& __b, const matrix< double > * __def) const [inline]

Definition at line 575 of file datamap-inline.h.

**6.33.2.14 bool datamap::retrieve (const char * _s, const matrix< double > *& _b) const** `[inline]`

Definition at line 570 of file datamap-inline.h.

**6.33.2.15 bool datamap::retrieve (const char * _s, const std::vector< interval > *& _b, const std::vector< interval > * _def) const** `[inline]`

Definition at line 565 of file datamap-inline.h.

**6.33.2.16 bool datamap::retrieve (const char * _s, const std::vector< interval > *& _b) const** `[inline]`

Definition at line 560 of file datamap-inline.h.

**6.33.2.17 bool datamap::retrieve (const char * _s, const std::vector< double > *& _b, const std::vector< double > * _def) const** `[inline]`

Definition at line 555 of file datamap-inline.h.

**6.33.2.18 bool datamap::retrieve (const char * _s, const std::vector< double > *& _b) const** `[inline]`

Definition at line 550 of file datamap-inline.h.

**6.33.2.19 bool datamap::retrieve (const char * _s, const std::vector< int > *& _b, const std::vector< int > * _def) const** `[inline]`

Definition at line 535 of file datamap-inline.h.

**6.33.2.20 bool datamap::retrieve (const char * _s, const std::vector< int > *& _b) const** `[inline]`

Definition at line 530 of file datamap-inline.h.

**6.33.2.21 bool datamap::retrieve (const char * _s, const std::vector< unsigned int > *& _b, const std::vector< unsigned int > * _def) const** `[inline]`

Definition at line 545 of file datamap-inline.h.

**6.33.2.22 bool datamap::retrieve (const char * _s, const std::vector< unsigned int > *& _b) const** `[inline]`

Definition at line 540 of file datamap-inline.h.

**6.33.2.23 bool datamap::retrieve (const char * _s, const std::vector< bool > *& _b, const std::vector< bool > * _def) const** `[inline]`

Definition at line 525 of file datamap-inline.h.

#### 6.33.2.24 bool datamap::retrieve (const char * __s, const std::vector< bool > *& __b) const [inline]

Definition at line 520 of file datamap-inline.h.

#### 6.33.2.25 bool datamap::retrieve (const char * __s, std::string & __b, const std::string & __def) const [inline]

Definition at line 515 of file datamap-inline.h.

#### 6.33.2.26 bool datamap::retrieve (const char * __s, std::string & __is) const [inline]

Definition at line 510 of file datamap-inline.h.

#### 6.33.2.27 bool datamap::retrieve (const char * __s, interval & __b, const interval & __def) const [inline]

Definition at line 505 of file datamap-inline.h.

#### 6.33.2.28 bool datamap::retrieve (const char * __s, interval & __b) const [inline]

Definition at line 500 of file datamap-inline.h.

#### 6.33.2.29 bool datamap::retrieve (const char * __s, double & __d, double __def) const [inline]

Definition at line 495 of file datamap-inline.h.

#### 6.33.2.30 bool datamap::retrieve (const char * __s, double & __d) const [inline]

Definition at line 490 of file datamap-inline.h.

#### 6.33.2.31 bool datamap::retrieve (const char * __s, unsigned int & __d, unsigned int __def) const [inline]

Definition at line 485 of file datamap-inline.h.

#### 6.33.2.32 bool datamap::retrieve (const char * __s, unsigned int & __d) const [inline]

Definition at line 480 of file datamap-inline.h.

#### 6.33.2.33 bool datamap::retrieve (const char * __s, int & __d, int __def) const [inline]

Definition at line 475 of file datamap-inline.h.

#### 6.33.2.34 bool datamap::retrieve (const char * __s, int & __d) const [inline]

Definition at line 470 of file datamap-inline.h.

#### 6.33.2.35 bool datamap::retrieve (const char * __s, bool & __b, bool __def) const [inline]

Definition at line 465 of file datamap-inline.h.

#### 6.33.2.36 bool datamap::retrieve (const char * __s, bool & __b) const [inline]

Definition at line 460 of file datamap-inline.h.

#### 6.33.2.37 bool datamap::retrieve (const std::string & __s, const matrix< interval > *& __b, const matrix< interval > * __def) const [inline]

Definition at line 449 of file datamap-inline.h.

#### 6.33.2.38 bool datamap::retrieve (const std::string & __s, const matrix< interval > *& __b) const [inline]

Definition at line 437 of file datamap-inline.h.

#### 6.33.2.39 bool datamap::retrieve (const std::string & __s, const matrix< int > *& __b, const matrix< int > * __def) const [inline]

Definition at line 427 of file datamap-inline.h.

#### 6.33.2.40 bool datamap::retrieve (const std::string & __s, const matrix< int > *& __b) const [inline]

Definition at line 415 of file datamap-inline.h.

#### 6.33.2.41 bool datamap::retrieve (const std::string & __s, const matrix< double > *& __b, const matrix< double > * __def) const [inline]

Definition at line 405 of file datamap-inline.h.

#### 6.33.2.42 bool datamap::retrieve (const std::string & __s, const matrix< double > *& __b) const [inline]

Definition at line 393 of file datamap-inline.h.

#### 6.33.2.43 bool datamap::retrieve (const std::string & __s, const std::vector< interval > *& __b, const std::vector< interval > * __def) const [inline]

Definition at line 383 of file datamap-inline.h.

#### 6.33.2.44 bool datamap::retrieve (const std::string & __s, const std::vector< interval > *& __b) const [inline]

Definition at line 371 of file datamap-inline.h.

#### 6.33.2.45 bool datamap::retrieve (const std::string & __s, const std::vector< double > *& __b, const std::vector< double > * __def) const [inline]

Definition at line 361 of file datamap-inline.h.

#### 6.33.2.46 bool datamap::retrieve (const std::string & __s, const std::vector< double > *& __b) const [inline]

Definition at line 349 of file datamap-inline.h.

**6.33.2.47 bool datamap::retrieve (const std::string & __s, const std::vector< unsigned int > *& __b, const std::vector< unsigned int > * __def) const** `[inline]`

Definition at line 339 of file datamap-inline.h.

**6.33.2.48 bool datamap::retrieve (const std::string & __s, const std::vector< unsigned int > *& __b) const** `[inline]`

Definition at line 327 of file datamap-inline.h.

**6.33.2.49 bool datamap::retrieve (const std::string & __s, const std::vector< int > *& __b, const std::vector< int > * __def) const** `[inline]`

Definition at line 317 of file datamap-inline.h.

**6.33.2.50 bool datamap::retrieve (const std::string & __s, const std::vector< int > *& __b) const** `[inline]`

Definition at line 305 of file datamap-inline.h.

**6.33.2.51 bool datamap::retrieve (const std::string & __s, const std::vector< bool > *& __b, const std::vector< bool > * __def) const** `[inline]`

Definition at line 295 of file datamap-inline.h.

**6.33.2.52 bool datamap::retrieve (const std::string & __s, const std::vector< bool > *& __b) const** `[inline]`

Definition at line 283 of file datamap-inline.h.

**6.33.2.53 bool datamap::retrieve (const std::string & __s, std::string & __is, const std::string & __def) const** `[inline]`

Definition at line 273 of file datamap-inline.h.

**6.33.2.54 bool datamap::retrieve (const std::string & __s, std::string & __is) const** `[inline]`

Definition at line 261 of file datamap-inline.h.

**6.33.2.55 bool datamap::retrieve (const std::string & __s, interval & __b, const interval & __def) const** `[inline]`

Definition at line 251 of file datamap-inline.h.

**6.33.2.56 bool datamap::retrieve (const std::string & __s, interval & __b) const** `[inline]`

Definition at line 239 of file datamap-inline.h.

**6.33.2.57 bool datamap::retrieve (const std::string & __s, double & __d, double __def) const** `[inline]`

Definition at line 229 of file datamap-inline.h.

#### 6.33.2.58 bool datamap::retrieve (const std::string & __s, double & __d) const [inline]

Definition at line 217 of file datamap-inline.h.

#### 6.33.2.59 bool datamap::retrieve (const std::string & __s, unsigned int & __d, unsigned int __def) const [inline]

Definition at line 207 of file datamap-inline.h.

#### 6.33.2.60 bool datamap::retrieve (const std::string & __s, unsigned int & __d) const [inline]

Definition at line 195 of file datamap-inline.h.

#### 6.33.2.61 bool datamap::retrieve (const std::string & __s, int & __d, int __def) const [inline]

Definition at line 185 of file datamap-inline.h.

#### 6.33.2.62 bool datamap::retrieve (const std::string & __s, int & __d) const [inline]

Definition at line 173 of file datamap-inline.h.

#### 6.33.2.63 bool datamap::retrieve (const std::string & __s, bool & __b, bool __def) const [inline]

Definition at line 163 of file datamap-inline.h.

#### 6.33.2.64 bool datamap::retrieve (const std::string & __s, bool & __b) const [inline]

Definition at line 146 of file datamap-inline.h.

#### 6.33.2.65 bool datamap::retrieve_i (const char * __s, int i, const matrix< interval > *& __b, const matrix< interval > * __def) const [inline]

Definition at line 1021 of file datamap-inline.h.

#### 6.33.2.66 bool datamap::retrieve_i (const char * __s, int i, const matrix< interval > *& __b) const [inline]

Definition at line 1016 of file datamap-inline.h.

#### 6.33.2.67 bool datamap::retrieve_i (const char * __s, int i, const matrix< int > *& __b, const matrix< int > * __def) const [inline]

Definition at line 1010 of file datamap-inline.h.

#### 6.33.2.68 bool datamap::retrieve_i (const char * __s, int i, const matrix< int > *& __b) const [inline]

Definition at line 1005 of file datamap-inline.h.

**6.33.2.69 bool datamap::retrieve_i (const char * __s, int i, const matrix< double > *& __b, const matrix< double > * __def) const** `[inline]`

Definition at line 999 of file datamap-inline.h.

**6.33.2.70 bool datamap::retrieve_i (const char * __s, int i, const matrix< double > *& __b) const** `[inline]`

Definition at line 994 of file datamap-inline.h.

**6.33.2.71 bool datamap::retrieve_i (const char * __s, int i, const std::vector< interval > *& __b, const std::vector< interval > * __def) const** `[inline]`

Definition at line 989 of file datamap-inline.h.

**6.33.2.72 bool datamap::retrieve_i (const char * __s, int i, const std::vector< interval > *& __b) const** `[inline]`

Definition at line 984 of file datamap-inline.h.

**6.33.2.73 bool datamap::retrieve_i (const char * __s, int i, const std::vector< double > *& __b, const std::vector< double > * __def) const** `[inline]`

Definition at line 979 of file datamap-inline.h.

**6.33.2.74 bool datamap::retrieve_i (const char * __s, int i, const std::vector< double > *& __b) const** `[inline]`

Definition at line 974 of file datamap-inline.h.

**6.33.2.75 bool datamap::retrieve_i (const char * __s, int i, const std::vector< unsigned int > *& __b, const std::vector< unsigned int > * __def) const** `[inline]`

Definition at line 969 of file datamap-inline.h.

**6.33.2.76 bool datamap::retrieve_i (const char * __s, int i, const std::vector< unsigned int > *& __b) const** `[inline]`

Definition at line 964 of file datamap-inline.h.

**6.33.2.77 bool datamap::retrieve_i (const char * __s, int i, const std::vector< int > *& __b, const std::vector< int > * __def) const** `[inline]`

Definition at line 959 of file datamap-inline.h.

**6.33.2.78 bool datamap::retrieve_i (const char * __s, int i, const std::vector< int > *& __b) const** `[inline]`

Definition at line 954 of file datamap-inline.h.

**6.33.2.79 bool datamap::retrieve_i (const char * __s, int i, const std::vector< bool > *& __b, const std::vector< bool > * __def) const** `[inline]`

Definition at line 949 of file datamap-inline.h.

**6.33.2.80 bool datamap::retrieve_i (const char * __s, int i, const std::vector< bool > *& __b) const** `[inline]`

Definition at line 944 of file datamap-inline.h.

**6.33.2.81 bool datamap::retrieve_i (const char * __s, int i, std::string & __b, const std::string & __def) const** `[inline]`

Definition at line 939 of file datamap-inline.h.

**6.33.2.82 bool datamap::retrieve_i (const char * __s, int i, std::string & __is) const** `[inline]`

Definition at line 934 of file datamap-inline.h.

**6.33.2.83 bool datamap::retrieve_i (const char * __s, int i, interval & __b, const interval & __def) const** `[inline]`

Definition at line 929 of file datamap-inline.h.

**6.33.2.84 bool datamap::retrieve_i (const char * __s, int i, interval & __b) const** `[inline]`

Definition at line 924 of file datamap-inline.h.

**6.33.2.85 bool datamap::retrieve_i (const char * __s, int i, double & __d, double __def) const** `[inline]`

Definition at line 919 of file datamap-inline.h.

**6.33.2.86 bool datamap::retrieve_i (const char * __s, int i, double & __d) const** `[inline]`

Definition at line 914 of file datamap-inline.h.

**6.33.2.87 bool datamap::retrieve_i (const char * __s, int i, unsigned int & __d, unsigned int __def) const** `[inline]`

Definition at line 909 of file datamap-inline.h.

**6.33.2.88 bool datamap::retrieve_i (const char * __s, int i, unsigned int & __d) const** `[inline]`

Definition at line 904 of file datamap-inline.h.

**6.33.2.89 bool datamap::retrieve_i (const char * __s, int i, int & __d, int __def) const** `[inline]`

Definition at line 899 of file datamap-inline.h.

**6.33.2.90 bool datamap::retrieve_i (const char * __s, int i, int & __d) const** `[inline]`

Definition at line 894 of file datamap-inline.h.

#### 6.33.2.91 bool datamap::retrieve_i (const char * __s, int i, bool & __b, bool __def) const [inline]

Definition at line 889 of file datamap-inline.h.

#### 6.33.2.92 bool datamap::retrieve_i (const char * __s, int i, bool & __b) const [inline]

Definition at line 884 of file datamap-inline.h.

#### 6.33.2.93 bool datamap::retrieve_i (const std::string & __s, int i, const matrix< interval > *& __b, const matrix< interval > * __def) const [inline]

Definition at line 874 of file datamap-inline.h.

#### 6.33.2.94 bool datamap::retrieve_i (const std::string & __s, int i, const matrix< interval > *& __b) const [inline]

Definition at line 863 of file datamap-inline.h.

#### 6.33.2.95 bool datamap::retrieve_i (const std::string & __s, int i, const matrix< int > *& __b, const matrix< int > * __def) const [inline]

Definition at line 854 of file datamap-inline.h.

#### 6.33.2.96 bool datamap::retrieve_i (const std::string & __s, int i, const matrix< int > *& __b) const [inline]

Definition at line 843 of file datamap-inline.h.

#### 6.33.2.97 bool datamap::retrieve_i (const std::string & __s, int i, const matrix< double > *& __b, const matrix< double > * __def) const [inline]

Definition at line 834 of file datamap-inline.h.

#### 6.33.2.98 bool datamap::retrieve_i (const std::string & __s, int i, const matrix< double > *& __b) const [inline]

Definition at line 823 of file datamap-inline.h.

#### 6.33.2.99 bool datamap::retrieve_i (const std::string & __s, int i, const std::vector< interval > *& __b, const std::vector< interval > * __def) const [inline]

Definition at line 814 of file datamap-inline.h.

#### 6.33.2.100 bool datamap::retrieve_i (const std::string & __s, int i, const std::vector< interval > *& __b) const [inline]

Definition at line 803 of file datamap-inline.h.

#### 6.33.2.101 bool datamap::retrieve_i (const std::string & __s, int i, const std::vector< double > *& __b, const std::vector< double > * __def) const [inline]

Definition at line 794 of file datamap-inline.h.

**6.33.2.102 bool datamap::retrieve_i (const std::string & __s, int i, const std::vector< double > *& __b) const** `[inline]`

Definition at line 783 of file datamap-inline.h.

**6.33.2.103 bool datamap::retrieve_i (const std::string & __s, int i, const std::vector< unsigned int > *& __b, const std::vector< unsigned int > * __def) const** `[inline]`

Definition at line 774 of file datamap-inline.h.

**6.33.2.104 bool datamap::retrieve_i (const std::string & __s, int i, const std::vector< unsigned int > *& __b) const** `[inline]`

Definition at line 763 of file datamap-inline.h.

**6.33.2.105 bool datamap::retrieve_i (const std::string & __s, int i, const std::vector< int > *& __b, const std::vector< int > * __def) const** `[inline]`

Definition at line 754 of file datamap-inline.h.

**6.33.2.106 bool datamap::retrieve_i (const std::string & __s, int i, const std::vector< int > *& __b) const** `[inline]`

Definition at line 743 of file datamap-inline.h.

**6.33.2.107 bool datamap::retrieve_i (const std::string & __s, int i, const std::vector< bool > *& __b, const std::vector< bool > * __def) const** `[inline]`

Definition at line 734 of file datamap-inline.h.

**6.33.2.108 bool datamap::retrieve_i (const std::string & __s, int i, const std::vector< bool > *& __b) const** `[inline]`

Definition at line 723 of file datamap-inline.h.

**6.33.2.109 bool datamap::retrieve_i (const std::string & __s, int i, std::string & __is, const std::string & __def) const** `[inline]`

Definition at line 714 of file datamap-inline.h.

**6.33.2.110 bool datamap::retrieve_i (const std::string & __s, int i, std::string & __is) const** `[inline]`

Definition at line 703 of file datamap-inline.h.

**6.33.2.111 bool datamap::retrieve_i (const std::string & __s, int i, interval & __b, const interval & __def) const** `[inline]`

Definition at line 694 of file datamap-inline.h.

#### 6.33.2.112 bool datamap::retrieve_i (const std::string & __s, int i, interval & __b) const [inline]

Definition at line 683 of file datamap-inline.h.

#### 6.33.2.113 bool datamap::retrieve_i (const std::string & __s, int i, double & __d, double __def) const [inline]

Definition at line 674 of file datamap-inline.h.

#### 6.33.2.114 bool datamap::retrieve_i (const std::string & __s, int i, double & __d) const [inline]

Definition at line 663 of file datamap-inline.h.

#### 6.33.2.115 bool datamap::retrieve_i (const std::string & __s, int i, unsigned int & __d, unsigned int __def) const [inline]

Definition at line 654 of file datamap-inline.h.

#### 6.33.2.116 bool datamap::retrieve_i (const std::string & __s, int i, unsigned int & __d) const [inline]

Definition at line 643 of file datamap-inline.h.

#### 6.33.2.117 bool datamap::retrieve_i (const std::string & __s, int i, int & __d, int __def) const [inline]

Definition at line 634 of file datamap-inline.h.

#### 6.33.2.118 bool datamap::retrieve_i (const std::string & __s, int i, int & __d) const [inline]

Definition at line 623 of file datamap-inline.h.

#### 6.33.2.119 bool datamap::retrieve_i (const std::string & __s, int i, bool & __b, bool __def) const [inline]

Definition at line 614 of file datamap-inline.h.

#### 6.33.2.120 bool datamap::retrieve_i (const std::string & __s, int i, bool & __b) const [inline]

Definition at line 603 of file datamap-inline.h.

#### 6.33.2.121 const additional_info_u & datamap::sfind (const char * __cp, int i) const [inline]

Definition at line 101 of file datamap-inline.h.

#### 6.33.2.122 const additional_info_u & datamap::sfind (const std::string & __s, int i) const [inline]

Definition at line 92 of file datamap-inline.h.

#### 6.33.2.123 const additional_info_u & datamap::sfind (const char * __cp) const [inline]

Definition at line 87 of file datamap-inline.h.

#### 6.33.2.124 const additional_info_u & datamap::sfind (const std::string & __s) const [inline]

Definition at line 78 of file datamap-inline.h.

#### 6.33.2.125 bool datamap::sinsert (const char * __cp, int i, const additional_info_u & __h, bool replace) [inline]

Definition at line 47 of file datamap-inline.h.

#### 6.33.2.126 bool datamap::sinsert (const std::string & __s, int i, const additional_info_u & __h, bool replace) [inline]

Definition at line 39 of file datamap-inline.h.

#### 6.33.2.127 bool datamap::sinsert (const char * __cp, const additional_info_u & __h, bool replace) [inline]

Definition at line 33 of file datamap-inline.h.

#### 6.33.2.128 bool datamap::sinsert (const std::string & __s, const additional_info_u & __h, bool replace)

Definition at line 29 of file datamap.cc.

#### 6.33.2.129 bool datamap::which (const char * __cp, std::vector< int > & __idx) const [inline]

Definition at line 140 of file datamap-inline.h.

#### 6.33.2.130 bool datamap::which (const std::string & __s, std::vector< int > & __idx) const [inline]

Definition at line 126 of file datamap-inline.h.

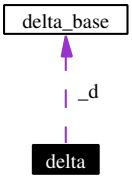
The documentation for this class was generated from the following files:

- datamap.h
- datamap-inline.h
- datamap.cc

## 6.34 delta Class Reference

`#include <api_deltabase.h>`

Collaboration diagram for delta:




### Public Methods

- delta ()
- delta (const delta_base &__d)
- delta (const delta &__d)
- ~delta ()
- const std::string & get_action () const
- const delta_base * get_base () const
- bool apply (work_node &_x, const delta_id &_d) const
- bool apply3 (work_node &_x, const work_node &_y, const delta_id &_d) const
- void convert (work_node &_x)
- delta_id store (work_node &_x)
- delta & operator= (const delta &_d)

### Friends

- class delta_base
- class ie_return_type
- std::ostream & operator<< (std::ostream &o, const delta &t)

#### 6.34.1 Constructor & Destructor Documentation

##### 6.34.1.1 delta::delta () [inline]

Definition at line 53 of file api_deltabase.h.

##### 6.34.1.2 delta::delta (const delta_base & __d) [inline]

Definition at line 87 of file api_delta.h.

##### 6.34.1.3 delta::delta (const delta & __d) [inline]

Definition at line 89 of file api_delta.h.

##### 6.34.1.4 delta::~delta () [inline]

Definition at line 95 of file api_delta.h.

#### 6.34.2 Member Function Documentation

##### 6.34.2.1 bool delta::apply (work_node & _x, const delta_id & _d) const [inline]

Definition at line 114 of file api_delta.h.

#### 6.34.2.2 bool delta::apply3 (work_node & *_x*, const work_node & *_y*, const delta_id & *_d*) const [inline]

Definition at line 126 of file api_delta.h.

#### 6.34.2.3 void delta::convert (work_node & *_x*) [inline]

Definition at line 139 of file api_delta.h.

#### 6.34.2.4 const std::string & delta::get_action () [inline]

Definition at line 111 of file api_delta.h.

#### 6.34.2.5 const delta_base * delta::get_base () [inline]

Definition at line 112 of file api_delta.h.

#### 6.34.2.6 delta & delta::operator= (const delta & *_d*) [inline]

Definition at line 101 of file api_delta.h.

#### 6.34.2.7 delta_id delta::store (work_node & *_x*) [inline]

Definition at line 147 of file api_delta.h.

### 6.34.3 Friends And Related Function Documentation

#### 6.34.3.1 friend class delta_base [friend]

Definition at line 73 of file api_deltabase.h.

#### 6.34.3.2 friend class ie_return_type [friend]

Definition at line 74 of file api_deltabase.h.

#### 6.34.3.3 std::ostream& operator<< (std::ostream & *o*, const delta & *t*) [friend]

Definition at line 77 of file api_deltabase.h.

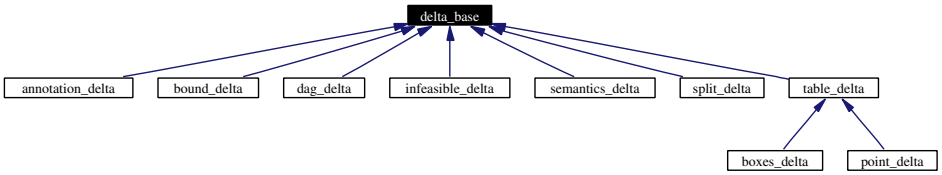
The documentation for this class was generated from the following files:

- api_deltabase.h
- api_delta.h

## 6.35 delta_base Class Reference

`#include <api_deltabase.h>`

Inheritance diagram for delta_base:




### Public Methods

- delta_base ()
- delta_base (const std::string &a)
- delta_base (const char *a)
- delta_base (const delta_base &__d)
- virtual delta_base * new_copy () const
- virtual void destroy_copy (delta_base *__d)
- virtual ~delta_base ()
- delta make_delta (const std::string &a)
- const std::string & get_action () const
- virtual void convert (work_node &_x, delta_base *&_d)
- virtual bool apply (work_node &_x, undelta_base *&_u) const
- virtual bool apply3 (work_node &_x, const work_node &_y, undelta_base *&_u) const

### Protected Attributes

- std::string _action

### 6.35.1 Constructor & Destructor Documentation

#### 6.35.1.1 delta_base::delta_base () [inline]

Definition at line 89 of file api_deltabase.h.

#### 6.35.1.2 delta_base::delta_base (const std::string & *a*) [inline]

Definition at line 90 of file api_deltabase.h.

#### 6.35.1.3 delta_base::delta_base (const char * *a*) [inline]

Definition at line 91 of file api_deltabase.h.

#### 6.35.1.4 delta_base::delta_base (const delta_base & *__d*) [inline]

Definition at line 92 of file api_deltabase.h.

#### 6.35.1.5 virtual delta_base::~delta_base () [inline, virtual]

Definition at line 97 of file api_deltabase.h.

### 6.35.2 Member Function Documentation

#### 6.35.2.1 virtual bool delta_base::apply (work_node & _x, undelta_base *& _u) const [inline, virtual]

Reimplemented in annotation_delta, bound_delta, boxes_delta, dag_delta, infeasible_delta, point_delta, semantics_delta, split_delta, and table_delta.

Definition at line 110 of file api_deltabase.h.

#### 6.35.2.2 bool delta_base::apply3 (work_node & _x, const work_node & _y, undelta_base *& _u) const [inline, virtual]

Definition at line 63 of file api_delta.h.

#### 6.35.2.3 virtual void delta_base::convert (work_node & _x, delta_base *& _d) [inline, virtual]

Reimplemented in table_delta.

Definition at line 107 of file api_deltabase.h.

#### 6.35.2.4 virtual void delta_base::destroy_copy (delta_base * _d) [inline, virtual]

Definition at line 95 of file api_deltabase.h.

#### 6.35.2.5 const std::string& delta_base::get_action () const [inline]

Definition at line 105 of file api_deltabase.h.

#### 6.35.2.6 delta delta_base::make_delta (const std::string & a) [inline]

Definition at line 99 of file api_deltabase.h.

#### 6.35.2.7 virtual delta_base* delta_base::new_copy () const [inline, virtual]

Reimplemented in annotation_delta, bound_delta, boxes_delta, dag_delta, infeasible_delta, point_delta, semantics_delta, split_delta, and table_delta.

Definition at line 94 of file api_deltabase.h.

### 6.35.3 Member Data Documentation

#### 6.35.3.1 std::string delta_base::_action [protected]

Definition at line 86 of file api_deltabase.h.

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

- api_deltabase.h
- api_delta.h

## 6.36 delta_get_action Class Reference

`#include <api_delta.h>`

### Public Types

- typedef vdbl::context context
- typedef std::string return_type

### Public Methods

- delta_get_action (**vdbl::colid** _c)
- delta_get_action (const delta_get_action &_i)
- virtual ~delta_get_action ()
- const std::string & operator() () const
- const std::string & def () const
- void setcontext (const context *c, const vdbl::row *r)

#### 6.36.1 Member Typedef Documentation

##### 6.36.1.1 typedef vdbl::context delta_get_action::context

Definition at line 48 of file api_delta.h.

##### 6.36.1.2 typedef std::string delta_get_action::return_type

Definition at line 49 of file api_delta.h.

#### 6.36.2 Constructor & Destructor Documentation

##### 6.36.2.1 delta_get_action::delta_get_action (vdbl::colid *_c*) [inline]

Definition at line 51 of file api_delta.h.

##### 6.36.2.2 delta_get_action::delta_get_action (const delta_get_action & *_i*) [inline]

Definition at line 52 of file api_delta.h.

##### 6.36.2.3 virtual delta_get_action::~delta_get_action () [inline, virtual]

Definition at line 54 of file api_delta.h.

#### 6.36.3 Member Function Documentation

##### 6.36.3.1 const std::string& delta_get_action::def () const [inline]

Definition at line 57 of file api_delta.h.

##### 6.36.3.2 const std::string & delta_get_action::operator() () [inline]

Definition at line 73 of file api_delta.h.

#### 6.36.3.3 void delta_get_action::setcontext (const context * *c*, const vdbl::row * *r*) [inline]

Definition at line 58 of file api_delta.h.

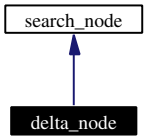
The documentation for this class was generated from the following file:

- api_delta.h

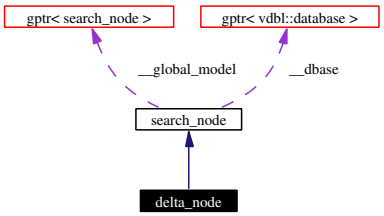
## 6.37 delta_node Class Reference

`#include <search_node.h>`

Inheritance diagram for delta_node:




Collaboration diagram for delta_node:




### Public Methods

- delta_node (const search_node_id &_i, const **vdbl::userid** &_dui, std::vector< delta_id > &__d, gptr< search_node > &_gm, gptr< vdbl::database > &_db, search_node_relation _snr=snr_reduction)
- virtual ~delta_node ()
- bool is_delta () const
- unsigned int n_deltas () const
- delta_id get_delta_id (unsigned int i) const
- delta get_delta (unsigned int i)
- const delta & get_delta (unsigned int i) const
- **vdbl::userid** get_dbuserid () const
- gptr< search_node > * global_model () const
- gptr< vdbl::database > * database () const
- search_node_id get_id () const
- void keep (const annotation &_an)
- void keep (const std::vector< annotation > &_anv)
- void unkeep (const annotation &_an)
- void unkeep (const std::vector< annotation > &_anv)

### Protected Methods

- search_node_id node_id () const
- search_node_relation parent_relation () const

### Protected Attributes

- gptr< search_node > * __global_model
- gptr< vdbl::database > * __dbase
- **vdbl::userid** _dbuser
- search_node_relation _snr
- search_node_id _id
- std::vector< annotation > _keep

### Friends

- class search_graph

#### 6.37.1 Constructor & Destructor Documentation

##### 6.37.1.1 delta_node::delta_node (const search_node_id & *_i*, const vdbl::userid & *_dui*, std::vector< delta_id > & *__d*, gptr< search_node > & *_gm*, gptr< vdbl::database > & *_db*, search_node_relation *_snr* = snr_reduction) [inline]

Definition at line 184 of file search_node.h.

##### 6.37.1.2 virtual delta_node::~delta_node () [inline, virtual]

Definition at line 191 of file search_node.h.

#### 6.37.2 Member Function Documentation

##### 6.37.2.1 gptr<vdbl::database>* search_node::database () const [inline, inherited]

Definition at line 147 of file search_node.h.

##### 6.37.2.2 vdbl::userid search_node::get_dbuserid () const [inline, inherited]

Definition at line 143 of file search_node.h.

##### 6.37.2.3 const delta & delta_node::get_delta (unsigned int *i*) const [inline]

Definition at line 652 of file search_node.h.

##### 6.37.2.4 delta delta_node::get_delta (unsigned int *i*) [inline]

Definition at line 639 of file search_node.h.

##### 6.37.2.5 delta_id delta_node::get_delta_id (unsigned int *i*) const [inline]

Definition at line 197 of file search_node.h.

#### 6.37.2.6 search_node_id search_node::get_id () const [inline, inherited]

Definition at line 149 of file search_node.h.

#### 6.37.2.7 gptr<search_node>* search_node::global_model () const [inline, inherited]

Definition at line 145 of file search_node.h.

#### 6.37.2.8 bool delta_node::is_delta () const [inline, virtual]

Reimplemented from search_node.

Definition at line 193 of file search_node.h.

#### 6.37.2.9 void search_node::keep (const std::vector< annotation > & *_anv*) [inline, inherited]

Definition at line 153 of file search_node.h.

#### 6.37.2.10 void search_node::keep (const annotation & *_an*) [inline, inherited]

Definition at line 151 of file search_node.h.

#### 6.37.2.11 unsigned int delta_node::n_deltas () const [inline]

Definition at line 195 of file search_node.h.

#### 6.37.2.12 search_node_id search_node::node_id () const [inline, protected, inherited]

Definition at line 96 of file search_node.h.

#### 6.37.2.13 search_node_relation search_node::parent_relation () const [inline, protected, inherited]

Definition at line 97 of file search_node.h.

#### 6.37.2.14 void search_node::unkeep (const std::vector< annotation > & *_anv*) [inline, inherited]

Definition at line 167 of file search_node.h.

#### 6.37.2.15 void search_node::unkeep (const annotation & *_an*) [inline, inherited]

Definition at line 156 of file search_node.h.

### 6.37.3 Friends And Related Function Documentation

#### 6.37.3.1 friend class search_graph [friend, inherited]

Definition at line 174 of file search_node.h.

### 6.37.4 Member Data Documentation

#### 6.37.4.1 gptr<vdbl::database>* search_node::__dbase [protected, inherited]

Definition at line 89 of file search_node.h.

#### 6.37.4.2 gptr<search_node>* search_node::__global_model [protected, inherited]

Definition at line 88 of file search_node.h.

#### 6.37.4.3 vdbl::userid search_node::_dbuser [protected, inherited]

Definition at line 90 of file search_node.h.

#### 6.37.4.4 search_node_id search_node::_id [protected, inherited]

Definition at line 92 of file search_node.h.

#### 6.37.4.5 std::vector<annotation> search_node::_keep [protected, inherited]

Definition at line 93 of file search_node.h.

#### 6.37.4.6 search_node_relation search_node::_snr [protected, inherited]

Definition at line 91 of file search_node.h.

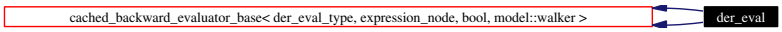
The documentation for this class was generated from the following file:

- search_node.h

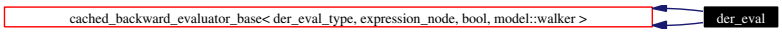
## 6.38 der_eval Class Reference

`#include <der_evaluator.h>`

Inheritance diagram for der_eval:




Collaboration diagram for der_eval:




### Public Types

- typedef cached_evaluator_base< der_eval_type, expression_node, bool, model::walker >::node_data_type node_data_type
- typedef cached_evaluator_base< der_eval_type, expression_node, bool, model::walker >::return_value return_value
- typedef cached_evaluator_base< der_eval_type, expression_node, bool, model::walker >::walker walker
- typedef der_eval_type data_type

### Public Methods

- der_eval (std::vector< std::vector< double > > &__der_data, variable_indicator &__v, const model &__m, std::vector< std::vector< double > > *__d, std::vector< double > &__grad)
- der_eval (const der_eval &__d)
- ~der_eval ()
- void new_point (std::vector< std::vector< double > > &__der_data, const variable_indicator &__v)
- void new_result (std::vector< double > &__grad)
- void set_mult (double scal)
- model::walker short_cut_to (const expression_node &__data)
- void initialize ()
- int calculate (const expression_node &__data)
- void cleanup (const expression_node &__data)
- void retrieve_from_cache (const expression_node &__data)
- int update (const bool &__rval)
- int update (const expression_node &__data, const bool &__rval)
- bool calculate_value (bool eval_all)
- der_eval (std::vector< std::vector< double > > &__der_data, variable_indicator &__v, const model &__m, std::vector< std::vector< double > > *__d, std::vector< double > &__grad)
- der_eval (const der_eval &__d)
- ~der_eval ()
- void new_point (std::vector< std::vector< double > > &__der_data, const variable_indicator &__v)
- void new_result (std::vector< double > &__grad)
- void set_mult (double scal)
- model::walker short_cut_to (const expression_node &__data)
- void initialize ()
- int calculate (const expression_node &__data)
- void cleanup (const expression_node &__data)
- void retrieve_from_cache (const expression_node &__data)
- int update (const bool &__rval)
- int update (const expression_node &__data, const bool &__rval)
- bool calculate_value (bool eval_all)
- int preorder (const node_data_type &__data)
- void postorder (const node_data_type &__data)
- int collect (const node_data_type &__data, const return_value &__rval)
- int vcollect (const return_value &__rval)
- void vinit ()
- return_value value ()
- return_value vvalue ()
- virtual int calculate (const node_data_type &__data)
- virtual void cleanup (const node_data_type &__data)
- virtual void retrieve_from_cache (const node_data_type &__data)
- virtual int update (const node_data_type &__data, const return_value &__rval)
- virtual int update (const return_value &__rval)
- virtual walker short_cut_to (const node_data_type &__data) PURE_VIRTUAL public

### Protected Methods

- bool is_cached (const node_data_type &__data)
- bool is_cached (const node_data_type &__data)

**Protected Attributes**

- const variable_indicator * v_ind
- der_eval_type eval_data

### 6.38.1 Member Typedef Documentation

#### 6.38.1.1 typedef der_eval_type _evaluator_base< der_eval_type, expression_node, bool, model::walker >::data_type [inherited]

Definition at line 245 of file evaluator.h.

#### 6.38.1.2 typedef cached_evaluator_base<der_eval_type, expression_node, bool,model::walker>::node_data_type cached_backward_evaluator_base< der_eval_type, expression_node, bool, model::walker >::node_data_type [inherited]

Reimplemented from cached_evaluator_base< der_eval_type, expression_node, bool, model::walker >.

Definition at line 444 of file evaluator.h.

#### 6.38.1.3 typedef cached_evaluator_base<der_eval_type, expression_node, bool,model::walker>::return_value cached_backward_evaluator_base< der_eval_type, expression_node, bool, model::walker >::return_value [inherited]

Reimplemented from cached_evaluator_base< der_eval_type, expression_node, bool, model::walker >.

Definition at line 446 of file evaluator.h.

#### 6.38.1.4 typedef cached_evaluator_base<der_eval_type, expression_node, bool,model::walker>::walker cached_backward_evaluator_base< der_eval_type, expression_node, bool, model::walker >::walker [inherited]

Reimplemented from cached_evaluator_base< der_eval_type, expression_node, bool, model::walker >.

Definition at line 448 of file evaluator.h.

### 6.38.2 Constructor & Destructor Documentation

#### 6.38.2.1 der_eval::der_eval (std::vector< std::vector< double > > & *__der_data*, variable_indicator & *__v*, const model & *__m*, std::vector< std::vector< double > > * *__d*, std::vector< double > & *__grad*) [inline]

Definition at line 957 of file der_evaluator.h.

#### 6.38.2.2 der_eval::der_eval (const der_eval & *__d*) [inline]

Definition at line 970 of file der_evaluator.h.

#### 6.38.2.3 der_eval::~der_eval () [inline]

Definition at line 972 of file der_evaluator.h.

#### 6.38.2.4 der_eval::der_eval (std::vector< std::vector< double > > & *__der_data*, variable_indicator & *__v*, const model & *__m*, std::vector< std::vector< double > > * *__d*, std::vector< double > & *__grad*) [inline]

Definition at line 1007 of file hess_evaluator.h.

#### 6.38.2.5 der_eval::der_eval (const der_eval & *__d*) [inline]

Definition at line 1020 of file hess_evaluator.h.

#### 6.38.2.6 der_eval::~der_eval () [inline]

Definition at line 1022 of file hess_evaluator.h.

### 6.38.3 Member Function Documentation

#### 6.38.3.1 virtual int cached_backward_evaluator_base< der_eval_type, expression_node, bool, model::walker >::calculate (const node_data_type & *__data*) [inline, virtual, inherited]

Definition at line 476 of file evaluator.h.

#### 6.38.3.2 int der_eval::calculate (const expression_node & *__data*) [inline]

Definition at line 1052 of file hess_evaluator.h.

#### 6.38.3.3 int der_eval::calculate (const expression_node & *__data*) [inline]

Definition at line 1002 of file der_evaluator.h.

#### 6.38.3.4 bool der_eval::calculate_value (bool *eval_all*) [inline, virtual]

Reimplemented from cached_backward_evaluator_base< der_eval_type, expression_node, bool, model::walker >.

Definition at line 1144 of file hess_evaluator.h.

#### 6.38.3.5 bool der_eval::calculate_value (bool *eval_all*) [inline, virtual]

Reimplemented from cached_backward_evaluator_base< der_eval_type, expression_node, bool, model::walker >.

Definition at line 1094 of file der_evaluator.h.

#### 6.38.3.6 virtual void cached_backward_evaluator_base< der_eval_type, expression_node, bool, model::walker >::cleanup (const node_data_type & *__data*) [inline, virtual, inherited]

Definition at line 477 of file evaluator.h.

#### 6.38.3.7 void der_eval::cleanup (const expression_node & *__data*) [inline]

Definition at line 1100 of file hess_evaluator.h.

#### 6.38.3.8 void der_eval::cleanup (const expression_node & *__data*) [inline]

Definition at line 1050 of file der_evaluator.h.

#### 6.38.3.9 int cached_backward_evaluator_base< der_eval_type, expression_node, bool, model::walker >::collect (const node_data_type & *__data*, const return_value & *__rval*) [inline, virtual, inherited]

Reimplemented from _evaluator_base< der_eval_type, expression_node, bool, model::walker >.

Definition at line 466 of file evaluator.h.

#### 6.38.3.10 void der_eval::initialize () [inline, virtual]

Reimplemented from cached_backward_evaluator_base< der_eval_type, expression_node, bool, model::walker >.

Definition at line 1046 of file hess_evaluator.h.

#### 6.38.3.11 void der_eval::initialize () [inline, virtual]

Reimplemented from cached_backward_evaluator_base< der_eval_type, expression_node, bool, model::walker >.

Definition at line 996 of file der_evaluator.h.

#### 6.38.3.12 bool der_eval::is_cached (const node_data_type & *__data*) [inline, protected, virtual]

Reimplemented from cached_backward_evaluator_base< der_eval_type, expression_node, bool, model::walker >.

Definition at line 994 of file hess_evaluator.h.

#### 6.38.3.13 bool der_eval::is_cached (const node_data_type & *__data*) [inline, protected, virtual]

Reimplemented from cached_backward_evaluator_base< der_eval_type, expression_node, bool, model::walker >.

Definition at line 944 of file der_evaluator.h.

#### 6.38.3.14 void der_eval::new_point (std::vector< std::vector< double > > & *__der_data*, const variable_indicator & *__v*) [inline]

Definition at line 1024 of file hess_evaluator.h.

#### 6.38.3.15 void der_eval::new_point (std::vector< std::vector< double > > & *__der_data*, const variable_indicator & *__v*) [inline]

Definition at line 974 of file der_evaluator.h.

#### 6.38.3.16 void der_eval::new_result (std::vector< double > & *__grad*) [inline]

Definition at line 1031 of file hess_evaluator.h.

#### 6.38.3.17 void der_eval::new_result (std::vector< double > & *_grad*) [inline]

Definition at line 981 of file der_evaluator.h.

#### 6.38.3.18 void cached_backward_evaluator_base< der_eval_type, expression_node, bool, model::walker >::postorder (const node_data_type & *_data*) [inline, virtual, inherited]

Reimplemented from _evaluator_base< der_eval_type, expression_node, bool, model::walker >.

Definition at line 465 of file evaluator.h.

#### 6.38.3.19 int cached_backward_evaluator_base< der_eval_type, expression_node, bool, model::walker >::preorder (const node_data_type & *_data*) [inline, virtual, inherited]

Reimplemented from cached_evaluator_base< der_eval_type, expression_node, bool, model::walker >.

Definition at line 456 of file evaluator.h.

#### 6.38.3.20 virtual void cached_backward_evaluator_base< der_eval_type, expression_node, bool, model::walker >::retrieve_from_cache (const node_data_type & *_data*) [inline, virtual, inherited]

Definition at line 478 of file evaluator.h.

#### 6.38.3.21 void der_eval::retrieve_from_cache (const expression_node & *_data*) [inline]

Definition at line 1111 of file hess_evaluator.h.

#### 6.38.3.22 void der_eval::retrieve_from_cache (const expression_node & *_data*) [inline]

Definition at line 1061 of file der_evaluator.h.

#### 6.38.3.23 void der_eval::set_mult (double *scal*) [inline]

Definition at line 1036 of file hess_evaluator.h.

#### 6.38.3.24 void der_eval::set_mult (double *scal*) [inline]

Definition at line 986 of file der_evaluator.h.

#### 6.38.3.25 virtual walker cached_evaluator_base< der_eval_type, expression_node, bool, model::walker >::short_cut_to (const node_data_type & *_data*) [inline, virtual, inherited]

Definition at line 303 of file evaluator.h.

#### 6.38.3.26 model::walker der_eval::short_cut_to (const expression_node & *_data*) [inline]

Definition at line 1042 of file hess_evaluator.h.

#### 6.38.3.27 model::walker der_eval::short_cut_to (const expression_node & *__data*) [inline]

Definition at line 992 of file der_evaluator.h.

#### 6.38.3.28 virtual int cached_backward_evaluator_base< der_eval_type, expression_node, bool, model::walker >::update (const return_value & *__rval*) [inline, virtual, inherited]

Definition at line 481 of file evaluator.h.

#### 6.38.3.29 virtual int cached_backward_evaluator_base< der_eval_type, expression_node, bool, model::walker >::update (const node_data_type & *__data*, const return_value & *__rval*) [inline, virtual, inherited]

Definition at line 479 of file evaluator.h.

#### 6.38.3.30 int der_eval::update (const expression_node & *__data*, const bool & *__rval*) [inline]

Definition at line 1125 of file hess_evaluator.h.

#### 6.38.3.31 int der_eval::update (const bool & *__rval*) [inline]

Definition at line 1118 of file hess_evaluator.h.

#### 6.38.3.32 int der_eval::update (const expression_node & *__data*, const bool & *__rval*) [inline]

Definition at line 1075 of file der_evaluator.h.

#### 6.38.3.33 int der_eval::update (const bool & *__rval*) [inline]

Definition at line 1068 of file der_evaluator.h.

#### 6.38.3.34 return_value cached_backward_evaluator_base< der_eval_type, expression_node, bool, model::walker >::value () [inline, virtual, inherited]

Reimplemented from _evaluator_base< der_eval_type, expression_node, bool, model::walker >.

Definition at line 471 of file evaluator.h.

#### 6.38.3.35 int cached_backward_evaluator_base< der_eval_type, expression_node, bool, model::walker >::vcollect (const return_value & *__rval*) [inline, virtual, inherited]

Reimplemented from _evaluator_base< der_eval_type, expression_node, bool, model::walker >.

Definition at line 468 of file evaluator.h.

#### 6.38.3.36 void cached_backward_evaluator_base< der_eval_type, expression_node, bool, model::walker >::vinit () [inline, inherited]

Definition at line 470 of file evaluator.h.

#### 6.38.3.37 return_value cached_backward_evaluator_base< der_eval_type, expression_node, bool, model::walker >::vvalue () [inline, virtual, inherited]

Reimplemented from _evaluator_base< der_eval_type, expression_node, bool, model::walker >.

Definition at line 472 of file evaluator.h.

### 6.38.4 Member Data Documentation

#### 6.38.4.1 der_eval_type _evaluator_base< der_eval_type, expression_node, bool, model::walker >::eval_data [protected, inherited]

Definition at line 252 of file evaluator.h.

#### 6.38.4.2 const variable_indicator* cached_evaluator_base< der_eval_type, expression_node, bool, model::walker >::v_ind [protected, inherited]

Definition at line 295 of file evaluator.h.

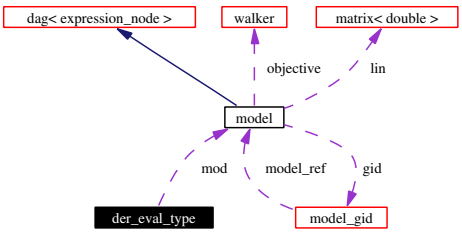
The documentation for this class was generated from the following files:

- der_evaluator.h
- hess_evaluator.h

## 6.39 der_eval_type Struct Reference

`#include <der_evaluator.h>`

Collaboration diagram for der_eval_type:




### Public Attributes

- std::vector< std::vector< double > > * d_data
- std::vector< std::vector< double > > * d_cache
- std::vector< double > * grad_vec
- const model * mod
- double mult
- double mult_trans
- unsigned int child_n
- std::vector< std::vector< double > > * d_data
- std::vector< std::vector< double > > * d_cache
- std::vector< double > * grad_vec
- const model * mod

### 6.39.1 Member Data Documentation

#### 6.39.1.1 unsigned int der_eval_type::child_n

Definition at line 982 of file hess_evaluator.h.

#### 6.39.1.2 std::vector<std::vector<double> >* der_eval_type::d_cache

Definition at line 977 of file hess_evaluator.h.

#### 6.39.1.3 std::vector<std::vector<double> >* der_eval_type::d_cache

Definition at line 927 of file der_evaluator.h.

#### 6.39.1.4 std::vector<std::vector<double> >* der_eval_type::d_data

Definition at line 976 of file hess_evaluator.h.

#### 6.39.1.5 std::vector<std::vector<double> >* der_eval_type::d_data

Definition at line 926 of file der_evaluator.h.

#### 6.39.1.6 std::vector<double>* der_eval_type::grad_vec

Definition at line 978 of file hess_evaluator.h.

#### 6.39.1.7 std::vector<double>* der_eval_type::grad_vec

Definition at line 928 of file der_evaluator.h.

#### 6.39.1.8 const model* der_eval_type::mod

Definition at line 979 of file hess_evaluator.h.

#### 6.39.1.9 const model* der_eval_type::mod

Definition at line 929 of file der_evaluator.h.

#### 6.39.1.10 double der_eval_type::mult

Definition at line 980 of file hess_evaluator.h.

#### 6.39.1.11 double der_eval_type::mult_trans

Definition at line 981 of file hess_evaluator.h.

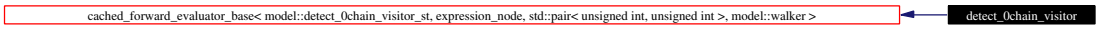
The documentation for this struct was generated from the following files:

- der_evaluator.h
- hess_evaluator.h

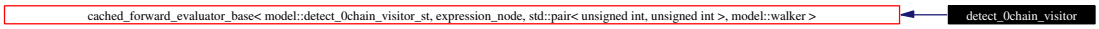
## 6.40 detect_0chain_visitor Class Reference

`#include <model-inline.h>`

Inheritance diagram for detect_0chain_visitor:




Collaboration diagram for detect_0chain_visitor:




### Public Types

- typedef cached_evaluator_base< model::detect_0chain_visitor_st, expression_node, std::pair< unsigned int, unsigned int >, model::walker >::node_data_type node_data_type
- typedef cached_evaluator_base< model::detect_0chain_visitor_st, expression_node, std::pair< unsigned int, unsigned int >, model::walker >::return_value return_value
- typedef cached_evaluator_base< model::detect_0chain_visitor_st, expression_node, std::pair< unsigned int, unsigned int >, model::walker >::walker walker
- typedef model::detect_0chain_visitor_st data_type

### Public Methods

- detect_0chain_visitor (std::vector< unsigned int > &__t, std::vector< unsigned int > &__d, std::vector< unsigned int > &__nn, unsigned int &d)
- detect_0chain_visitor (const detect_0chain_visitor &__x)
- ~detect_0chain_visitor ()
- void initialize ()
- bool is_cached (const expression_node &__data)
- void retrieve_from_cache (const expression_node &__data)
- int initialize (const expression_node &__data)
- void calculate (const expression_node &__data)
- int update (const std::pair< unsigned int, unsigned int > &__rval)
- int update (const expression_node &__data, const std::pair< unsigned int, unsigned int > &__rval)
- std::pair< unsigned int, unsigned int > calculate_value (bool eval_all)
- virtual bool is_cached (const node_data_type &__data)
- int preorder (const node_data_type &__data)
- void postorder (const node_data_type &__data)
- int collect (const node_data_type &__data, const return_value &__rval)
- int vcollect (const return_value &__rval)
- return_value value ()
- return_value vvalue ()
- void vinit ()
- virtual int initialize (const node_data_type &__data)
- virtual void calculate (const node_data_type &__data)
- virtual void retrieve_from_cache (const node_data_type &__data)
- virtual void cleanup (const node_data_type &__data)
- virtual int update (const node_data_type &__data, const return_value &__rval)
- virtual int update (const return_value &__rval)
- virtual walker short_cut_to (const node_data_type &__data) PURE_VIRTUAL public

**Protected Attributes**

- const variable_indicator * v_ind
- model::detect_0chain_visitor_st eval_data

### 6.40.1 Member Typedef Documentation

**6.40.1.1 typedef model::detect_0chain_visitor_st _evaluator_base< model::detect_0chain_visitor_st, expression_node, std::pair< unsigned int, unsigned int >, model::walker >::data_type** `[inherited]`

Definition at line 245 of file evaluator.h.

**6.40.1.2 typedef cached_evaluator_base<model::detect_0chain_visitor_st, expression_node, std::pair< unsigned int, unsigned int >,model::walker>::node_data_type cached_forward_evaluator_base< model::detect_0chain_visitor_st, expression_node, std::pair< unsigned int, unsigned int >, model::walker >::node_data_type** `[inherited]`

Reimplemented from cached_evaluator_base< model::detect_0chain_visitor_st, expression_node, std::pair< unsigned int, unsigned int >, model::walker >.

Definition at line 396 of file evaluator.h.

**6.40.1.3 typedef cached_evaluator_base<model::detect_0chain_visitor_st, expression_node, std::pair< unsigned int, unsigned int >,model::walker>::return_value cached_forward_evaluator_base< model::detect_0chain_visitor_st, expression_node, std::pair< unsigned int, unsigned int >, model::walker >::return_value** `[inherited]`

Reimplemented from cached_evaluator_base< model::detect_0chain_visitor_st, expression_node, std::pair< unsigned int, unsigned int >, model::walker >.

Definition at line 398 of file evaluator.h.

**6.40.1.4 typedef cached_evaluator_base<model::detect_0chain_visitor_st, expression_node, std::pair< unsigned int, unsigned int >,model::walker>::walker cached_forward_evaluator_base< model::detect_0chain_visitor_st, expression_node, std::pair< unsigned int, unsigned int >, model::walker >::walker** `[inherited]`

Reimplemented from cached_evaluator_base< model::detect_0chain_visitor_st, expression_node, std::pair< unsigned int, unsigned int >, model::walker >.

Definition at line 400 of file evaluator.h.

### 6.40.2 Constructor & Destructor Documentation

**6.40.2.1 detect_0chain_visitor::detect_0chain_visitor (std::vector< unsigned int > & *__t*, std::vector< unsigned int > & *__d*, std::vector< unsigned int > & *__nn*, unsigned int & *d*)** `[inline]`

Definition at line 1595 of file model-inline.h.

**6.40.2.2 detect_0chain_visitor::detect_0chain_visitor (const detect_0chain_visitor & *__x*)** `[inline]`

Definition at line 1606 of file model-inline.h.

#### 6.40.2.3 detect_0chain_visitor::~detect_0chain_visitor () [inline]

Definition at line 1609 of file model-inline.h.

### 6.40.3 Member Function Documentation

#### 6.40.3.1 virtual void cached_forward_evaluator_base< model::detect_0chain_visitor_st, expression_node, std::pair< unsigned int, unsigned int >, model::walker >::calculate (const node_data_type & *__data*) [inline, virtual, inherited]

Definition at line 430 of file evaluator.h.

#### 6.40.3.2 void detect_0chain_visitor::calculate (const expression_node & *__data*) [inline]

Definition at line 1638 of file model-inline.h.

#### 6.40.3.3 std::pair<unsigned int,unsigned int> detect_0chain_visitor::calculate_value (bool *eval_all*) [inline, virtual]

Reimplemented from cached_forward_evaluator_base< model::detect_0chain_visitor_st, expression_node, std::pair< unsigned int, unsigned int >, model::walker >.

Definition at line 1667 of file model-inline.h.

#### 6.40.3.4 virtual void cached_forward_evaluator_base< model::detect_0chain_visitor_st, expression_node, std::pair< unsigned int, unsigned int >, model::walker >::cleanup (const node_data_type & *__data*) [inline, virtual, inherited]

Definition at line 432 of file evaluator.h.

#### 6.40.3.5 int cached_forward_evaluator_base< model::detect_0chain_visitor_st, expression_node, std::pair< unsigned int, unsigned int >, model::walker >::collect (const node_data_type & *__data*, const return_value & *__rval*) [inline, virtual, inherited]

Reimplemented from _evaluator_base< model::detect_0chain_visitor_st, expression_node, std::pair< unsigned int, unsigned int >, model::walker >.

Definition at line 419 of file evaluator.h.

#### 6.40.3.6 virtual int cached_forward_evaluator_base< model::detect_0chain_visitor_st, expression_node, std::pair< unsigned int, unsigned int >, model::walker >::initialize (const node_data_type & *__data*) [inline, virtual, inherited]

Definition at line 429 of file evaluator.h.

#### 6.40.3.7 int detect_0chain_visitor::initialize (const expression_node & *__data*) [inline]

Definition at line 1620 of file model-inline.h.

#### 6.40.3.8 void detect_0chain_visitor::initialize () [inline, virtual]

Reimplemented from cached_forward_evaluator_base< model::detect_0chain_visitor_st, expression_node, std::pair< unsigned int, unsigned int >, model::walker >.

Definition at line 1611 of file model-inline.h.

**6.40.3.9 virtual bool cached_forward_evaluator_base< model::detect_0chain_visitor_st, expression_node, std::pair< unsigned int, unsigned int >, model::walker >::is_cached (const node_data_type & __data) [inline, virtual, inherited]**

Definition at line 405 of file evaluator.h.

**6.40.3.10 bool detect_0chain_visitor::is_cached (const expression_node & __data) [inline]**

Definition at line 1613 of file model-inline.h.

**6.40.3.11 void cached_forward_evaluator_base< model::detect_0chain_visitor_st, expression_node, std::pair< unsigned int, unsigned int >, model::walker >::postorder (const node_data_type & __data) [inline, virtual, inherited]**

Reimplemented from _evaluator_base< model::detect_0chain_visitor_st, expression_node, std::pair< unsigned int, unsigned int >, model::walker >.

Definition at line 417 of file evaluator.h.

**6.40.3.12 int cached_forward_evaluator_base< model::detect_0chain_visitor_st, expression_node, std::pair< unsigned int, unsigned int >, model::walker >::preorder (const node_data_type & __data) [inline, virtual, inherited]**

Reimplemented from cached_evaluator_base< model::detect_0chain_visitor_st, expression_node, std::pair< unsigned int, unsigned int >, model::walker >.

Definition at line 408 of file evaluator.h.

**6.40.3.13 virtual void cached_forward_evaluator_base< model::detect_0chain_visitor_st, expression_node, std::pair< unsigned int, unsigned int >, model::walker >::retrieve_from_cache (const node_data_type & __data) [inline, virtual, inherited]**

Definition at line 431 of file evaluator.h.

**6.40.3.14 void detect_0chain_visitor::retrieve_from_cache (const expression_node & __data) [inline]**

Definition at line 1616 of file model-inline.h.

**6.40.3.15 virtual walker cached_evaluator_base< model::detect_0chain_visitor_st, expression_node, std::pair< unsigned int, unsigned int >, model::walker >::short_cut_to (const node_data_type & __data) [inline, virtual, inherited]**

Definition at line 303 of file evaluator.h.

**6.40.3.16 virtual int cached_forward_evaluator_base< model::detect_0chain_visitor_st, expression_node, std::pair< unsigned int, unsigned int >, model::walker >::update (const return_value & __rval) [inline, virtual, inherited]**

Definition at line 435 of file evaluator.h.

**6.40.3.17 virtual int cached_forward_evaluator_base< model::detect_0chain_visitor_st, expression_node, std::pair< unsigned int, unsigned int >, model::walker >::update (const node_data_type & __data, const return_value & __rval) [inline, virtual, inherited]**

Definition at line 433 of file evaluator.h.

#### 6.40.3.18 int detect_0chain_visitor::update (const expression_node & *__data*, const std::pair< unsigned int, unsigned int > & *__rval*) [inline]

Definition at line 1646 of file model-inline.h.

#### 6.40.3.19 int detect_0chain_visitor::update (const std::pair< unsigned int, unsigned int > & *__rval*) [inline]

Definition at line 1644 of file model-inline.h.

#### 6.40.3.20 return_value cached_forward_evaluator_base< model::detect_0chain_visitor_st, expression_node, std::pair< unsigned int, unsigned int >, model::walker >::value () [inline, virtual, inherited]

Reimplemented from _evaluator_base< model::detect_0chain_visitor_st, expression_node, std::pair< unsigned int, unsigned int >, model::walker >.

Definition at line 423 of file evaluator.h.

#### 6.40.3.21 int cached_forward_evaluator_base< model::detect_0chain_visitor_st, expression_node, std::pair< unsigned int, unsigned int >, model::walker >::vcollect (const return_value & *__rval*) [inline, virtual, inherited]

Reimplemented from _evaluator_base< model::detect_0chain_visitor_st, expression_node, std::pair< unsigned int, unsigned int >, model::walker >.

Definition at line 421 of file evaluator.h.

#### 6.40.3.22 void cached_forward_evaluator_base< model::detect_0chain_visitor_st, expression_node, std::pair< unsigned int, unsigned int >, model::walker >::vinit () [inline, inherited]

Definition at line 425 of file evaluator.h.

#### 6.40.3.23 return_value cached_forward_evaluator_base< model::detect_0chain_visitor_st, expression_node, std::pair< unsigned int, unsigned int >, model::walker >::vvalue () [inline, virtual, inherited]

Reimplemented from _evaluator_base< model::detect_0chain_visitor_st, expression_node, std::pair< unsigned int, unsigned int >, model::walker >.

Definition at line 424 of file evaluator.h.

### 6.40.4 Member Data Documentation

#### 6.40.4.1 model::detect_0chain_visitor_st _evaluator_base< model::detect_0chain_visitor_st, expression_node, std::pair< unsigned int, unsigned int >, model::walker >::eval_data [protected, inherited]

Definition at line 252 of file evaluator.h.

#### 6.40.4.2 const variable_indicator* cached_evaluator_base< model::detect_0chain_visitor_st, expression_node, std::pair< unsigned int, unsigned int >, model::walker >::v_ind [protected, inherited]

Definition at line 295 of file evaluator.h.

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

- model-inline.h

## 6.41 detect_0chain_visitor_st Struct Reference

`#include <model-inline.h>`

### Public Attributes

- std::pair< unsigned int, unsigned int > r
- unsigned int * f
- std::vector< unsigned int > * t
- std::vector< unsigned int > * d
- std::vector< unsigned int > * nn
- bool new_number

### 6.41.1 Member Data Documentation

#### 6.41.1.1 std::vector<unsigned int>* detect_0chain_visitor_st::d

Definition at line 1580 of file model-inline.h.

#### 6.41.1.2 unsigned int* detect_0chain_visitor_st::f

Definition at line 1578 of file model-inline.h.

#### 6.41.1.3 bool detect_0chain_visitor_st::new_number

Definition at line 1582 of file model-inline.h.

#### 6.41.1.4 std::vector<unsigned int>* detect_0chain_visitor_st::nn

Definition at line 1581 of file model-inline.h.

#### 6.41.1.5 std::pair<unsigned int,unsigned int> detect_0chain_visitor_st::r

Definition at line 1577 of file model-inline.h.

#### 6.41.1.6 std::vector<unsigned int>* detect_0chain_visitor_st::t

Definition at line 1579 of file model-inline.h.

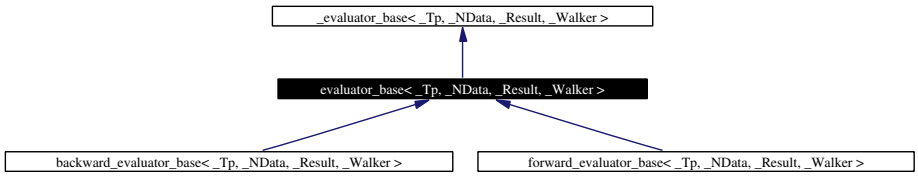
The documentation for this struct was generated from the following file:

- model-inline.h

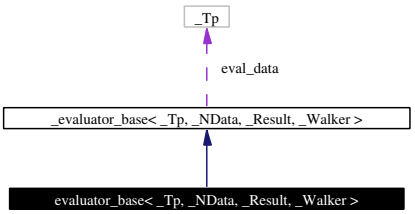
## 6.42 evaluator_base< _Tp, _NData, _Result, _Walker > Class Template Reference

`#include <evaluator.h>`

Inheritance diagram for evaluator_base< _Tp, _NData, _Result, _Walker >:




Collaboration diagram for evaluator_base< _Tp, _NData, _Result, _Walker >:




### Public Types

- typedef _evaluator_base< _Tp, _NData, _Result, _Walker >::node_data_type node_data_type
- typedef _evaluator_base< _Tp, _NData, _Result, _Walker >::return_value return_value
- typedef _evaluator_base< _Tp, _NData, _Result, _Walker >::walker walker
- typedef _Tp data_type

### Public Methods

- virtual int preorder (const node_data_type &__data)
- virtual return_value vvalue ()
- virtual return_value value ()
- virtual int vcollect (const return_value &__cresult)
- virtual int collect (const node_data_type &__data, const return_value &__cresult)
- virtual void postorder (const node_data_type &__data)

### Protected Attributes

- _Tp eval_data

**template<class _Tp, class _NData, class _Result, class _Walker> class evaluator_base< _Tp, _NData, _Result, _Walker >**

### 6.42.1 Member Typedef Documentation

#### 6.42.1.1 template<class _Tp, class _NData, class _Result, class _Walker> typedef _Tp _evaluator_base< _Tp, _NData, _Result, _Walker >::data_type [inherited]

Definition at line 245 of file evaluator.h.

#### 6.42.1.2 template<class _Tp, class _NData, class _Result, class _Walker> typedef _evaluator_base< _Tp, _NData, _Result, _Walker >::node_data_type evaluator_base< _Tp, _NData, _Result, _Walker >::node_data_type

Reimplemented from _evaluator_base< _Tp, _NData, _Result, _Walker >.

Reimplemented in forward_evaluator_base< _Tp, _NData, _Result, _Walker >, and backward_evaluator_base< _Tp, _NData, _Result, _Walker >.

Definition at line 277 of file evaluator.h.

#### 6.42.1.3 template<class _Tp, class _NData, class _Result, class _Walker> typedef _evaluator_base< _Tp, _NData, _Result, _Walker >::return_value evaluator_base< _Tp, _NData, _Result, _Walker >::return_value

Reimplemented from _evaluator_base< _Tp, _NData, _Result, _Walker >.

Reimplemented in forward_evaluator_base< _Tp, _NData, _Result, _Walker >, and backward_evaluator_base< _Tp, _NData, _Result, _Walker >.

Definition at line 279 of file evaluator.h.

#### 6.42.1.4 template<class _Tp, class _NData, class _Result, class _Walker> typedef _evaluator_base< _Tp, _NData, _Result, _Walker >::walker evaluator_base< _Tp, _NData, _Result, _Walker >::walker

Reimplemented from _evaluator_base< _Tp, _NData, _Result, _Walker >.

Reimplemented in forward_evaluator_base< _Tp, _NData, _Result, _Walker >, and backward_evaluator_base< _Tp, _NData, _Result, _Walker >.

Definition at line 281 of file evaluator.h.

### 6.42.2 Member Function Documentation

#### 6.42.2.1 template<class _Tp, class _NData, class _Result, class _Walker> virtual int _evaluator_base< _Tp, _NData, _Result, _Walker >::collect (const node_data_type & *__data*, const return_value & *__cresult*) [inline, virtual, inherited]

Reimplemented in forward_evaluator_base< _Tp, _NData, _Result, _Walker >, backward_evaluator_base< _Tp, _NData, _Result, _Walker >, cached_forward_evaluator_base< _Tp, _NData, _Result, _Walker >, cached_backward_evaluator_base< _Tp, _NData, _Result, _Walker >, cached_forward_evaluator_base< std::vector< std::vector< interval > > *, expression_node, bool, model::walker >, cached_forward_evaluator_base< func_eval_type, expression_node, double, model::walker >, cached_forward_evaluator_base< func_islp_eval_type, expression_node, func_islp_return_type, model::walker >, cached_forward_evaluator_base< cinterval_eval_type, expression_node, cinterval, model::walker >, cached_forward_evaluator_base< interval_eval_type, expression_node, interval, model::walker >, cached_forward_

evaluator_base< func_h_eval_type, expression_node, func_h_eval_ret, model::walker >, cached_forward_evaluator_base< func_d_eval_type, expression_node, double, model::walker >, cached_forward_evaluator_base< std::vector< std::vector< double > > *, expression_node, bool, model::walker >, cached_forward_evaluator_base< model::detect_0chain_visitor_st, expression_node, std::pair< unsigned int, unsigned int >, model::walker >, cached_forward_evaluator_base< infbound_eval_type, expression_node, infbound, model::walker >, cached_forward_evaluator_base< b_interval_eval_type, expression_node, b_interval, model::walker >, cached_forward_evaluator_base< analyticd_eval_type, expression_node, analyticd, model::walker >, cached_forward_evaluator_base< func_id_eval_type, expression_node, interval, model::walker >, cached_backward_evaluator_base< ider_eval_type, expression_node, bool, model::walker >, cached_backward_evaluator_base< islp_eval_type, expression_node, bool, model::walker >, cached_backward_evaluator_base< lincoeff_visitor_st, expression_node, lincoeff_visitor_ret, model::walker >, and cached_backward_evaluator_base< der_eval_type, expression_node, bool, model::walker >.

Definition at line 265 of file evaluator.h.

#### 6.42.2.2 template<class _Tp, class _NData, class _Result, class _Walker> virtual void evaluator_base< _Tp, _NData, _Result, _Walker >::postorder (const node_data_type & *_data*) [inline, virtual, inherited]

Reimplemented in forward_evaluator_base< _Tp, _NData, _Result, _Walker >, backward_evaluator_base< _Tp, _NData, _Result, _Walker >, cached_forward_evaluator_base< _Tp, _NData, _Result, _Walker >, cached_backward_evaluator_base< _Tp, _NData, _Result, _Walker >, cached_forward_evaluator_base< std::vector< std::vector< interval > > *, expression_node, bool, model::walker >, cached_forward_evaluator_base< func_eval_type, expression_node, double, model::walker >, cached_forward_evaluator_base< func_islp_eval_type, expression_node, func_islp_return_type, model::walker >, cached_forward_evaluator_base< cinterval_eval_type, expression_node, cinterval, model::walker >, cached_forward_evaluator_base< interval_eval_type, expression_node, interval, model::walker >, cached_forward_evaluator_base< func_h_eval_type, expression_node, func_h_eval_ret, model::walker >, cached_forward_evaluator_base< func_d_eval_type, expression_node, double, model::walker >, cached_forward_evaluator_base< std::vector< std::vector< double > > *, expression_node, bool, model::walker >, cached_forward_evaluator_base< model::detect_0chain_visitor_st, expression_node, std::pair< unsigned int, unsigned int >, model::walker >, cached_forward_evaluator_base< infbound_eval_type, expression_node, infbound, model::walker >, cached_forward_evaluator_base< b_interval_eval_type, expression_node, b_interval, model::walker >, cached_forward_evaluator_base< analyticd_eval_type, expression_node, analyticd, model::walker >, cached_forward_evaluator_base< func_id_eval_type, expression_node, interval, model::walker >, cached_backward_evaluator_base< ider_eval_type, expression_node, bool, model::walker >, cached_backward_evaluator_base< islp_eval_type, expression_node, bool, model::walker >, cached_backward_evaluator_base< lincoeff_visitor_st, expression_node, lincoeff_visitor_ret, model::walker >, and cached_backward_evaluator_base< der_eval_type, expression_node, bool, model::walker >.

Definition at line 268 of file evaluator.h.

#### 6.42.2.3 template<class _Tp, class _NData, class _Result, class _Walker> virtual int evaluator_base< _Tp, _NData, _Result, _Walker >::preorder (const node_data_type & *_data*) [inline, virtual]

Reimplemented in forward_evaluator_base< _Tp, _NData, _Result, _Walker >, and backward_evaluator_base< _Tp, _NData, _Result, _Walker >.

Definition at line 283 of file evaluator.h.

#### 6.42.2.4 template<class _Tp, class _NData, class _Result, class _Walker> virtual return_value evaluator_base< _Tp, _NData, _Result, _Walker >::value () [inline, virtual, inherited]

Reimplemented in forward_evaluator_base< _Tp, _NData, _Result, _Walker >, backward_evaluator_base< _Tp, _NData, _Result, _Walker >, cached_forward_evaluator_base< _Tp, _NData, _Result, _Walker >, cached_backward_evaluator_base< _Tp, _NData, _Result, _Walker >, cached_forward_evaluator_base< std::vector< std::vector< interval > > *, expression_node, bool, model::walker >, cached_forward_evaluator_base< func_eval_type, expression_node, double, model::walker >, cached_forward_evaluator_base< func_islp_eval_type, expression_node, func_islp_return_type, model::walker >, cached_forward_evaluator_base< cinterval_eval_type, expression_node, cinterval, model::walker >, cached_forward_evaluator_base< interval_eval_type, expression_node, interval, model::walker >, cached_forward_evaluator_base< func_h_eval_type, expression_node, func_h_eval_ret, model::walker >, cached_forward_evaluator_base< func_d_eval_type, expression_node, double, model::walker >, cached_forward_evaluator_base< std::vector< std::vector< double > > *, expression_node, bool, model::walker >, cached_forward_evaluator_base< model::detect_0chain_visitor_st, expression_node, std::pair< unsigned int, unsigned int >, model::walker >, cached_forward_evaluator_base< infbound_eval_type, expression_node, infbound, model::walker >, cached_forward_evaluator_base< b_interval_eval_type, expression_node, b_interval, model::walker >, cached_forward_evaluator_base< analyticd_eval_type, expression_node, analyticd, model::walker >, cached_forward_evaluator_base< func_id_eval_type, expression_node, interval, model::walker >, cached_backward_evaluator_base< ider_eval_type, expression_node, bool, model::walker >, cached_backward_evaluator_base< islp_eval_type, expression_node, bool, model::walker >, cached_backward_evaluator_base< lincoeff_visitor_st, expression_node, lincoeff_visitor_ret, model::walker >, and cached_backward_evaluator_base< der_eval_type, expression_node, bool, model::walker >.

Definition at line 262 of file evaluator.h.

#### 6.42.2.5 template<class _Tp, class _NData, class _Result, class _Walker> virtual int _evaluator_base< _Tp, _NData, _Result, _Walker >::vcollect (const return_value & *__cresult*) [inline, virtual, inherited]

Reimplemented in forward_evaluator_base< _Tp, _NData, _Result, _Walker >, backward_evaluator_base< _Tp, _NData, _Result, _Walker >, cached_forward_evaluator_base< _Tp, _NData, _Result, _Walker >, cached_backward_evaluator_base< _Tp, _NData, _Result, _Walker >, cached_forward_evaluator_base< std::vector< std::vector< interval > > *, expression_node, bool, model::walker >, cached_forward_evaluator_base< func_eval_type, expression_node, double, model::walker >, cached_forward_evaluator_base< func_islp_eval_type, expression_node, func_islp_return_type, model::walker >, cached_forward_evaluator_base< cinterval_eval_type, expression_node, cinterval, model::walker >, cached_forward_evaluator_base< interval_eval_type, expression_node, interval, model::walker >, cached_forward_evaluator_base< func_h_eval_type, expression_node, func_h_eval_ret, model::walker >, cached_forward_evaluator_base< func_d_eval_type, expression_node, double, model::walker >, cached_forward_evaluator_base< std::vector< std::vector< double > > *, expression_node, bool, model::walker >, cached_forward_evaluator_base< model::detect_0chain_visitor_st, expression_node, std::pair< unsigned int, unsigned int >, model::walker >, cached_forward_evaluator_base< infbound_eval_type, expression_node, infbound, model::walker >, cached_forward_evaluator_base< b_interval_eval_type, expression_node, b_interval, model::walker >, cached_forward_evaluator_base< analyticd_eval_type, expression_node, analyticd, model::walker >, cached_forward_evaluator_base< func_id_eval_type, expression_node, interval, model::walker >, cached_backward_evaluator_base< ider_eval_type, expression_node, bool, model::walker >, cached_backward_evaluator_base< islp_eval_type, expression_node, bool, model::walker >, cached_backward_evaluator_base< lincoeff_visitor_st, expression_node, lincoeff_visitor_ret, model::walker >, and cached_backward_evaluator_base< der_eval_type, expression_node, bool, model::walker >.

Definition at line 264 of file evaluator.h.

#### 6.42.2.6 template<class _Tp, class _NData, class _Result, class _Walker> virtual return_value _evaluator_base< _Tp, _NData, _Result, _Walker >::vvalue () [inline, virtual, inherited]

Reimplemented in forward_evaluator_base< _Tp, _NData, _Result, _Walker >, backward_evaluator_base< _Tp, _NData, _Result, _Walker >, cached_forward_evaluator_base< _Tp, _NData, _Result, _Walker >, cached_backward_evaluator_base< _Tp, _NData, _Result, _Walker >, cached_forward_evaluator_base< std::vector< std::vector< interval > > *, expression_node, bool, model::walker >, cached_forward_evaluator_base< func_eval_type, expression_node, double, model::walker >, cached_forward_evaluator_base< func_islp_eval_type, expression_node, func_islp_return_type, model::walker >, cached_forward_evaluator_base< cinterval_eval_type, expression_node, cinterval, model::walker >, cached_forward_evaluator_base< interval_eval_type, expression_node, interval, model::walker >, cached_forward_evaluator_base< func_h_eval_type, expression_node, func_h_eval_ret, model::walker >, cached_forward_evaluator_base< func_d_eval_type, expression_node, double, model::walker >, cached_forward_evaluator_base< std::vector< std::vector< double > > *, expression_node, bool, model::walker >, cached_forward_evaluator_base< model::detect_0chain_visitor_st, expression_node, std::pair< unsigned int, unsigned int >, model::walker >, cached_forward_evaluator_base< infbound_eval_type, expression_node, infbound, model::walker >, cached_forward_evaluator_base< b_interval_eval_type, expression_node, b_interval, model::walker >, cached_forward_evaluator_base< analyticd_eval_type, expression_node, analyticd, model::walker >, cached_forward_evaluator_base< func_id_eval_type, expression_node, interval, model::walker >, cached_backward_evaluator_base< ider_eval_type, expression_node, bool, model::walker >, cached_backward_evaluator_base< islp_eval_type, expression_node, bool, model::walker >, cached_backward_evaluator_base< lincoeff_visitor_st, expression_node, lincoeff_visitor_ret, model::walker >, and cached_backward_evaluator_base< der_eval_type, expression_node, bool, model::walker >.

Definition at line 261 of file evaluator.h.

### 6.42.3 Member Data Documentation

#### 6.42.3.1 template<class _Tp, class _NData, class _Result, class _Walker> _Tp _evaluator_base< _Tp, _NData, _Result, _Walker >::eval_data [protected, inherited]

Definition at line 252 of file evaluator.h.

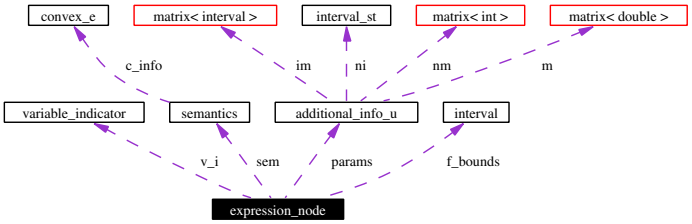
The documentation for this class was generated from the following file:

- evaluator.h

## 6.43 expression_node Class Reference

`#include <expression.h>`

Collaboration diagram for expression_node:




### Public Methods

- expression_node ()
- expression_node (int et, int nn)

- expression_node (const expression_node &__x)
- ~expression_node ()
- expression_node & operator= (const expression_node &__x)
- bool operator< (const expression_node &__x) const
- void merge (const expression_node &__s)
- void set_bounds (interval __i)
- void set_bounds (double __d=0.)
- void set_bounds (int __i)
- void set_bounds (double lo, double up)
- void add_is_var (unsigned int idx)
- void rm_is_var (unsigned int idx)
- bool is (unsigned int __tp) const
- const variable_indicator & var_indicator () const
- double f_evaluate (int argnum, int idx, const std::vector< double > &x, const variable_indicator &v_i, double fold, double fupd, std::vector< double > *cache_data) const
- interval i_evaluate (int argnum, int idx, const std::vector< interval > &x, const variable_indicator &v_i, interval fold, interval fupd, std::vector< interval > *cache_data) const
- interval cp_evaluate (int argnum, int idx, const std::vector< interval > &node_range, const variable_indicator &v_i, interval fold, interval fupd, std::vector< interval > *cache_data) const

**Public Attributes**

- unsigned int node_num
- int operator_type
- unsigned int n_parents
- unsigned int n_children
- std::vector< double > coeffs
- additional_info_u params
- interval f_bounds
- unsigned short is_var
- std::vector< unsigned int > var_idx
- semantics sem
- variable_indicator v_i
- evaluator_v * ev

**Friends**

- std::ostream & operator<< (std::ostream &o, const expression_node &__x)
- std::string & print_C_pre (std::string &__s, const expression_node &__x)
- std::string & print_C_post (std::string &__s, const expression_node &__x)

### 6.43.1 Constructor & Destructor Documentation

#### 6.43.1.1 expression_node::expression_node () [inline]

Definition at line 249 of file expression.h.

#### 6.43.1.2 expression_node::expression_node (int *et*, int *nn*) [inline]

Definition at line 254 of file expression.h.

#### 6.43.1.3 expression_node::expression_node (const expression_node & __x) [inline]

Definition at line 260 of file expression.h.

#### 6.43.1.4 expression_node::~expression_node () [inline]

Definition at line 267 of file expression.h.

### 6.43.2 Member Function Documentation

#### 6.43.2.1 void expression_node::add_is_var (unsigned int *idx*) [inline]

Definition at line 329 of file expression.h.

#### 6.43.2.2 interval expression_node::cp_evaluate (int *argnum*, int *idx*, const std::vector< interval > & *node_range*, const variable_indicator & *v_i*, interval *fold*, interval *fupd*, std::vector< interval > * *cache_data*) const [inline]

Definition at line 362 of file expression.h.

#### 6.43.2.3 double expression_node::f_evaluate (int *argnum*, int *idx*, const std::vector< double > & *x*, const variable_indicator & *v_i*, double *fold*, double *fupd*, std::vector< double > * *cache_data*) const [inline]

Definition at line 360 of file expression.h.

#### 6.43.2.4 interval expression_node::i_evaluate (int *argnum*, int *idx*, const std::vector< interval > & *x*, const variable_indicator & *v_i*, interval *fold*, interval *fupd*, std::vector< interval > * *cache_data*) const [inline]

Definition at line 361 of file expression.h.

#### 6.43.2.5 bool expression_node::is (unsigned int __tp) const

Definition at line 33 of file expression.cc.

#### 6.43.2.6 void expression_node::merge (const expression_node & __s) [inline]

Definition at line 297 of file expression.h.

#### 6.43.2.7 bool expression_node::operator< (const expression_node & __x) const [inline]

Definition at line 204 of file expr-inline.h.

#### 6.43.2.8 expression_node& expression_node::operator= (const expression_node & __x) [inline]

Definition at line 275 of file expression.h.

#### 6.43.2.9 void expression_node::rm_is_var (unsigned int *idx*) [inline]

Definition at line 335 of file expression.h.

#### 6.43.2.10 void expression_node::set_bounds (double *lo*, double *up*) [inline]

Definition at line 324 of file expression.h.

#### 6.43.2.11 void expression_node::set_bounds (int *__i*) [inline]

Definition at line 319 of file expression.h.

#### 6.43.2.12 void expression_node::set_bounds (double *__d* = 0.) [inline]

Definition at line 314 of file expression.h.

#### 6.43.2.13 void expression_node::set_bounds (interval *__i*) [inline]

Definition at line 309 of file expression.h.

#### 6.43.2.14 const variable_indicator& expression_node::var_indicator () const [inline]

Definition at line 357 of file expression.h.

### 6.43.3 Friends And Related Function Documentation

#### 6.43.3.1 std::ostream& operator<< (std::ostream & *o*, const expression_node & *__x*) [friend]

Definition at line 195 of file expression.cc.

#### 6.43.3.2 std::string& print_C_post (std::string & *__s*, const expression_node & *__x*) [friend]

#### 6.43.3.3 std::string& print_C_pre (std::string & *__s*, const expression_node & *__x*) [friend]

### 6.43.4 Member Data Documentation

#### 6.43.4.1 std::vector<double> expression_node::coeffs

Definition at line 221 of file expression.h.

#### 6.43.4.2 evaluator_v* expression_node::ev

Definition at line 236 of file expression.h.

#### 6.43.4.3 interval expression_node::f_bounds

Definition at line 225 of file expression.h.

#### 6.43.4.4 unsigned short expression_node::is_var

Definition at line 227 of file expression.h.

#### 6.43.4.5 unsigned int expression_node::n_children

Definition at line 218 of file expression.h.

#### 6.43.4.6 unsigned int expression_node::n_parents

Definition at line 218 of file expression.h.

#### 6.43.4.7 unsigned int expression_node::node_num

Definition at line 207 of file expression.h.

#### 6.43.4.8 int expression_node::operator_type

Definition at line 209 of file expression.h.

#### 6.43.4.9 additional_info_u expression_node::params

Definition at line 223 of file expression.h.

#### 6.43.4.10 semantics expression_node::sem

Definition at line 230 of file expression.h.

#### 6.43.4.11 variable_indicator expression_node::v_i

Definition at line 234 of file expression.h.

#### 6.43.4.12 std::vector<unsigned int> expression_node::var_idx

Definition at line 228 of file expression.h.

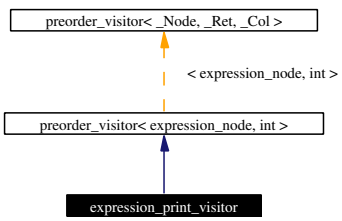
The documentation for this class was generated from the following files:

- expression.h
- expr-inline.h
- expression.cc

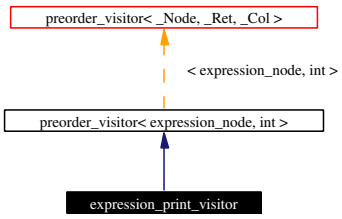
## 6.44 expression_print_visitor Class Reference

`#include <expr-inline.h>`

Inheritance diagram for expression_print_visitor:




Collaboration diagram for expression_print_visitor:




### Public Methods

- expression_print_visitor (std::vector< bool > &__p, std::ostream &__o=std::cout)
- expression_print_visitor (const expression_print_visitor &__p)
- ~expression_print_visitor ()
- bool preorder (const expression_node &r)
- int vvalue ()
- int value ()
- void collect (const expression_node &r, int __r)
- virtual void vinit ()
- virtual void collect (const expression_node &__n, collect_value __r)

### Public Attributes

- int return_value

## 6.44.1 Constructor & Destructor Documentation

#### 6.44.1.1 expression_print_visitor::expression_print_visitor (std::vector< bool > & __p, std::ostream & __o = std::cout) [inline]

Definition at line 217 of file expr-inline.h.

#### 6.44.1.2 expression_print_visitor::expression_print_visitor (const expression_print_visitor & __p) [inline]

Definition at line 220 of file expr-inline.h.

#### 6.44.1.3 expression_print_visitor::~expression_print_visitor () [inline]

Definition at line 222 of file expr-inline.h.

## 6.44.2 Member Function Documentation

#### 6.44.2.1 virtual void preorder_visitor< expression_node, int, _Col >::collect (const expression_node & __n, collect_value __r) [virtual, inherited]

#### 6.44.2.2 void expression_print_visitor::collect (const expression_node & r, int __r) [inline]

Definition at line 241 of file expr-inline.h.

#### 6.44.2.3 bool expression_print_visitor::preorder (const expression_node & *r*) [inline, virtual]

Reimplemented from **preorder_visitor< expression_node, int >**.

Definition at line 224 of file expr-inline.h.

#### 6.44.2.4 int expression_print_visitor::value () [inline]

Definition at line 240 of file expr-inline.h.

#### 6.44.2.5 virtual void preorder_visitor< expression_node, int, _Col >::vinit () [virtual, inherited]

#### 6.44.2.6 int expression_print_visitor::vvalue () [inline, virtual]

Reimplemented from **preorder_visitor< expression_node, int >**.

Definition at line 239 of file expr-inline.h.

### 6.44.3 Member Data Documentation

#### 6.44.3.1 int preorder_visitor< expression_node, int, _Col >::return_value [inherited]

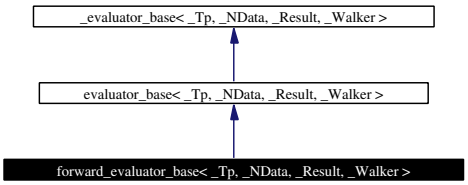
The documentation for this class was generated from the following file:

- expr-inline.h

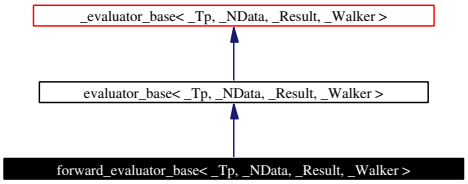
## 6.45 forward_evaluator_base< _Tp, _NData, _Result, _Walker > Class Template Reference

`#include <evaluator.h>`

Inheritance diagram for forward_evaluator_base< _Tp, _NData, _Result, _Walker >:




Collaboration diagram for forward_evaluator_base< _Tp, _NData, _Result, _Walker >:



### Public Types

- typedef _Base::node_data_type node_data_type
- typedef _Base::return_value return_value
- typedef _Base::walker walker
- typedef _Tp data_type

### Public Methods

- int preorder (const node_data_type &__data)
- void postorder (const node_data_type &__data)
- int collect (const node_data_type &__data, const return_value &__rval)
- int vcollect (const return_value &__rval)
- return_value value ()
- return_value vvalue ()
- void vinit ()
- virtual void initialize ()
- virtual int initialize (const node_data_type &__data)
- virtual void calculate (const node_data_type &__data)
- virtual void cleanup (const node_data_type &__data)
- virtual int update (const return_value &__rval)
- virtual int update (const node_data_type &__data, const return_value &__rval)
- virtual return_value calculate_value (bool eval_all)

### Protected Attributes

- _Tp eval_data

**template<class _Tp, class _NData, class _Result, class _Walker> class forward_evaluator_base< _Tp, _NData, _Result, _Walker >**

#### 6.45.1 Member Typedef Documentation

##### 6.45.1.1 template<class _Tp, class _NData, class _Result, class _Walker> typedef _Tp _evaluator_base< _Tp, _NData, _Result, _Walker >::data_type [inherited]

Definition at line 245 of file evaluator.h.

##### 6.45.1.2 template<class _Tp, class _NData, class _Result, class _Walker> typedef _Base::node_data_type forward_evaluator_base< _Tp, _NData, _Result, _Walker >::node_data_type

Reimplemented from evaluator_base< _Tp, _NData, _Result, _Walker >.

Definition at line 322 of file evaluator.h.

##### 6.45.1.3 template<class _Tp, class _NData, class _Result, class _Walker> typedef _Base::return_value forward_evaluator_base< _Tp, _NData, _Result, _Walker >::return_value

Reimplemented from evaluator_base< _Tp, _NData, _Result, _Walker >.

Definition at line 323 of file evaluator.h.

#### 6.45.1.4 template<class _Tp, class _NData, class _Result, class _Walker> typedef _Base::walker forward_evaluator_base< _Tp, _NData, _Result, _Walker >::walker

Reimplemented from evaluator_base< _Tp, _NData, _Result, _Walker >.

Definition at line 324 of file evaluator.h.

### 6.45.2 Member Function Documentation

#### 6.45.2.1 template<class _Tp, class _NData, class _Result, class _Walker> virtual void forward_evaluator_base< _Tp, _NData, _Result, _Walker >::calculate (const node_data_type & *__data*) [inline, virtual]

Definition at line 342 of file evaluator.h.

#### 6.45.2.2 template<class _Tp, class _NData, class _Result, class _Walker> virtual return_value forward_evaluator_base< _Tp, _NData, _Result, _Walker >::calculate_value (bool *eval_all*) [inline, virtual]

Definition at line 347 of file evaluator.h.

#### 6.45.2.3 template<class _Tp, class _NData, class _Result, class _Walker> virtual void forward_evaluator_base< _Tp, _NData, _Result, _Walker >::cleanup (const node_data_type & *__data*) [inline, virtual]

Definition at line 343 of file evaluator.h.

#### 6.45.2.4 template<class _Tp, class _NData, class _Result, class _Walker> int forward_evaluator_base< _Tp, _NData, _Result, _Walker >::collect (const node_data_type & *__data*, const return_value & *__rval*) [inline, virtual]

Reimplemented from _evaluator_base< _Tp, _NData, _Result, _Walker >.

Definition at line 331 of file evaluator.h.

#### 6.45.2.5 template<class _Tp, class _NData, class _Result, class _Walker> virtual int forward_evaluator_base< _Tp, _NData, _Result, _Walker >::initialize (const node_data_type & *__data*) [inline, virtual]

Definition at line 341 of file evaluator.h.

#### 6.45.2.6 template<class _Tp, class _NData, class _Result, class _Walker> virtual void forward_evaluator_base< _Tp, _NData, _Result, _Walker >::initialize () [inline, virtual]

Definition at line 340 of file evaluator.h.

#### 6.45.2.7 template<class _Tp, class _NData, class _Result, class _Walker> void forward_evaluator_base< _Tp, _NData, _Result, _Walker >::postorder (const node_data_type & *__data*) [inline, virtual]

Reimplemented from _evaluator_base< _Tp, _NData, _Result, _Walker >.

Definition at line 329 of file evaluator.h.

#### 6.45.2.8 template<class _Tp, class _NData, class _Result, class _Walker> int forward_evaluator_base< _Tp, _NData, _Result, _Walker >::preorder (const node_data_type & __data) [inline, virtual]

Reimplemented from evaluator_base< _Tp, _NData, _Result, _Walker >.

Definition at line 327 of file evaluator.h.

#### 6.45.2.9 template<class _Tp, class _NData, class _Result, class _Walker> virtual int forward_evaluator_base< _Tp, _NData, _Result, _Walker >::update (const node_data_type & __data, const return_value & __rval) [inline, virtual]

Definition at line 345 of file evaluator.h.

#### 6.45.2.10 template<class _Tp, class _NData, class _Result, class _Walker> virtual int forward_evaluator_base< _Tp, _NData, _Result, _Walker >::update (const return_value & __rval) [inline, virtual]

Definition at line 344 of file evaluator.h.

#### 6.45.2.11 template<class _Tp, class _NData, class _Result, class _Walker> return_value forward_evaluator_base< _Tp, _NData, _Result, _Walker >::value () [inline, virtual]

Reimplemented from _evaluator_base< _Tp, _NData, _Result, _Walker >.

Definition at line 335 of file evaluator.h.

#### 6.45.2.12 template<class _Tp, class _NData, class _Result, class _Walker> int forward_evaluator_base< _Tp, _NData, _Result, _Walker >::vcollect (const return_value & __rval) [inline, virtual]

Reimplemented from _evaluator_base< _Tp, _NData, _Result, _Walker >.

Definition at line 333 of file evaluator.h.

#### 6.45.2.13 template<class _Tp, class _NData, class _Result, class _Walker> void forward_evaluator_base< _Tp, _NData, _Result, _Walker >::vinit () [inline]

Definition at line 337 of file evaluator.h.

#### 6.45.2.14 template<class _Tp, class _NData, class _Result, class _Walker> return_value forward_evaluator_base< _Tp, _NData, _Result, _Walker >::vvalue () [inline, virtual]

Reimplemented from _evaluator_base< _Tp, _NData, _Result, _Walker >.

Definition at line 336 of file evaluator.h.

### 6.45.3 Member Data Documentation

#### 6.45.3.1 template<class _Tp, class _NData, class _Result, class _Walker> _Tp _evaluator_base< _Tp, _NData, _Result, _Walker >::eval_data [protected, inherited]

Definition at line 252 of file evaluator.h.

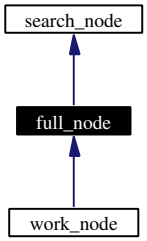
The documentation for this class was generated from the following file:

- evaluator.h

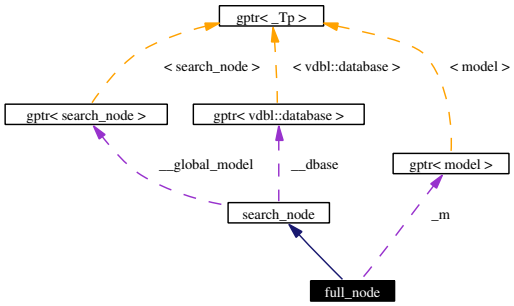
## 6.46 full_node Class Reference

`#include <search_node.h>`

Inheritance diagram for full_node:




Collaboration diagram for full_node:




### Public Methods

- full_node (const search_node_id &_i, const **vdbl::userid** &_dui, gptr< model > &__mod, gptr< search_node > &_gm, gptr< vdbl::database > &_db, search_node_relation _snr=snr_reduction)
- full_node (const search_node_id &_i, const **vdbl::userid** &_dui, gptr< model > &__mod, gptr< search_node > &_gm, gptr< vdbl::database > &_db, const std::vector< annotation > &_a, search_node_relation _snr=snr_reduction)
- virtual ~full_node ()
- bool is_delta () const
- const annotation & get_annotation (unsigned int i) const
- const std::vector< annotation > & get_annotations () const
- const model * get_model () const
- const vdbl::database * get_database () const
- model * get_model_ptr () const
- vdbl::database * get_database_ptr () const
- **vdbl::userid** get_dbuserid () const
- gptr< search_node > * global_model () const
- gptr< vdbl::database > * database () const
- search_node_id get_id () const

- void keep (const annotation &_an)
- void keep (const std::vector< annotation > &_anv)
- void unkeep (const annotation &_an)
- void unkeep (const std::vector< annotation > &_anv)

### Public Attributes

- std::vector< annotation > _ann

### Protected Methods

- full_node (const search_node_id &_i, const **vdbl::userid** &_dui, gptr< model > &__mod, gptr< search_node > *_gm, gptr< vdbl::database > &_db, search_node_relation _snr=snr_reduction)
- full_node (const search_node_id &_i, const **vdbl::userid** &_dui, gptr< model > &__mod, gptr< search_node > *_gm, gptr< vdbl::database > &_db, const std::vector< annotation > &_a, search_node_relation _snr=snr_reduction)
- search_node_id node_id () const
- search_node_relation parent_relation () const

### Protected Attributes

- gptr< model > * _m
- gptr< search_node > * __global_model
- gptr< vdbl::database > * __dbase
- **vdbl::userid** _dbuser
- search_node_relation _snr
- search_node_id _id
- std::vector< annotation > _keep

### Friends

- class delta_base
- class dag_delta
- class dag_undelta
- class search_graph

#### 6.46.1 Constructor & Destructor Documentation

##### 6.46.1.1 full_node::full_node (const search_node_id & *_i*, const vdbl::userid & *_dui*, gptr< model > & *__mod*, gptr< search_node > * *_gm*, gptr< vdbl::database > & *_db*, search_node_relation *_snr* = snr_reduction) [inline, protected]

Definition at line 216 of file search_node.h.

##### 6.46.1.2 full_node::full_node (const search_node_id & *_i*, const vdbl::userid & *_dui*, gptr< model > & *__mod*, gptr< search_node > * *_gm*, gptr< vdbl::database > & *_db*, const std::vector< annotation > & *_a*, search_node_relation *_snr* = snr_reduction) [inline, protected]

Definition at line 225 of file search_node.h.

#### 6.46.1.3 full_node::full_node (const search_node_id & _i, const vdbl::userid & _dui, gptr< model > & __mod, gptr< search_node > & _gm, gptr< vdbl::database > & _db, search_node_relation _snr = snr_reduction) [inline]

Definition at line 236 of file search_node.h.

#### 6.46.1.4 full_node::full_node (const search_node_id & _i, const vdbl::userid & _dui, gptr< model > & __mod, gptr< search_node > & _gm, gptr< vdbl::database > & _db, const std::vector< annotation > & _a, search_node_relation _snr = snr_reduction) [inline]

Definition at line 245 of file search_node.h.

#### 6.46.1.5 virtual full_node::~full_node () [inline, virtual]

Definition at line 254 of file search_node.h.

### 6.46.2 Member Function Documentation

#### 6.46.2.1 gptr<vdbl::database>* search_node::database () const [inline, inherited]

Definition at line 147 of file search_node.h.

#### 6.46.2.2 const annotation& full_node::get_annotation (unsigned int i) const [inline]

Definition at line 258 of file search_node.h.

#### 6.46.2.3 const std::vector<annotation>& full_node::get_annotations () const [inline]

Definition at line 263 of file search_node.h.

#### 6.46.2.4 const vdbl::database* full_node::get_database () const [inline]

Definition at line 270 of file search_node.h.

#### 6.46.2.5 vdbl::database* full_node::get_database_ptr () const [inline]

Definition at line 278 of file search_node.h.

#### 6.46.2.6 vdbl::userid search_node::get_dbuserid () const [inline, inherited]

Definition at line 143 of file search_node.h.

#### 6.46.2.7 search_node_id search_node::get_id () const [inline, inherited]

Definition at line 149 of file search_node.h.

#### 6.46.2.8 const model* full_node::get_model () const [inline]

Reimplemented in work_node.

Definition at line 268 of file search_node.h.

#### 6.46.2.9 model* full_node::get_model_ptr () const [inline]

Definition at line 273 of file search_node.h.

#### 6.46.2.10 gptr<search_node>* search_node::global_model () const [inline, inherited]

Definition at line 145 of file search_node.h.

#### 6.46.2.11 bool full_node::is_delta () const [inline, virtual]

Reimplemented from search_node.

Definition at line 256 of file search_node.h.

#### 6.46.2.12 void search_node::keep (const std::vector< annotation > & *_anv*) [inline, inherited]

Definition at line 153 of file search_node.h.

#### 6.46.2.13 void search_node::keep (const annotation & *_an*) [inline, inherited]

Definition at line 151 of file search_node.h.

#### 6.46.2.14 search_node_id search_node::node_id () const [inline, protected, inherited]

Definition at line 96 of file search_node.h.

#### 6.46.2.15 search_node_relation search_node::parent_relation () const [inline, protected, inherited]

Definition at line 97 of file search_node.h.

#### 6.46.2.16 void search_node::unkeep (const std::vector< annotation > & *_anv*) [inline, inherited]

Definition at line 167 of file search_node.h.

#### 6.46.2.17 void search_node::unkeep (const annotation & *_an*) [inline, inherited]

Definition at line 156 of file search_node.h.

### 6.46.3 Friends And Related Function Documentation

#### 6.46.3.1 friend class dag_delta [friend]

Definition at line 284 of file search_node.h.

#### 6.46.3.2 friend class dag_undelta [friend]

Definition at line 285 of file search_node.h.

#### 6.46.3.3 friend class delta_base [friend]

Definition at line 283 of file search_node.h.

#### 6.46.3.4 friend class search_graph [friend, inherited]

Definition at line 174 of file search_node.h.

### 6.46.4 Member Data Documentation

#### 6.46.4.1 gptr<vdbl::database>* search_node::__dbase [protected, inherited]

Definition at line 89 of file search_node.h.

#### 6.46.4.2 gptr<search_node>* search_node::__global_model [protected, inherited]

Definition at line 88 of file search_node.h.

#### 6.46.4.3 std::vector<annotation> full_node::_ann

Definition at line 213 of file search_node.h.

#### 6.46.4.4 vdbl::userid search_node::_dbuser [protected, inherited]

Definition at line 90 of file search_node.h.

#### 6.46.4.5 search_node_id search_node::_id [protected, inherited]

Definition at line 92 of file search_node.h.

#### 6.46.4.6 std::vector<annotation> search_node::_keep [protected, inherited]

Definition at line 93 of file search_node.h.

#### 6.46.4.7 gptr<model>* full_node::_m [protected]

Definition at line 210 of file search_node.h.

#### 6.46.4.8 search_node_relation search_node::_snr [protected, inherited]

Definition at line 91 of file search_node.h.

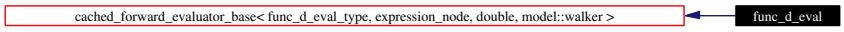
The documentation for this class was generated from the following file:

- search_node.h

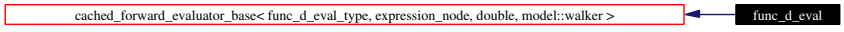
## 6.47 func_d_eval Class Reference

`#include <der_evaluator.h>`

Inheritance diagram for func_d_eval:




Collaboration diagram for func_d_eval:




### Public Types

- typedef cached_evaluator_base< func_d_eval_type, expression_node, double, model::walker >::node_data_type node_data_type
- typedef cached_evaluator_base< func_d_eval_type, expression_node, double, model::walker >::return_value return_value
- typedef cached_evaluator_base< func_d_eval_type, expression_node, double, model::walker >::walker walker
- typedef func_d_eval_type data_type

### Public Methods

- func_d_eval (const std::vector< double > &__x, const variable_indicator &__v, const model &__m, std::vector< std::vector< double > > &__d, std::vector< double > *__c)
- func_d_eval (const func_d_eval &__x)
- ~func_d_eval ()
- model::walker short_cut_to (const expression_node &__data)
- void new_point (const std::vector< double > &__x, const variable_indicator &__v)
- void initialize ()
- int initialize (const expression_node &__data)
- void calculate (const expression_node &__data)
- void retrieve_from_cache (const expression_node &__data)
- int update (const double &__rval)
- int update (const expression_node &__data, const double &__rval)
- double calculate_value (bool eval_all)
- int preorder (const node_data_type &__data)
- void postorder (const node_data_type &__data)
- int collect (const node_data_type &__data, const return_value &__rval)
- int vcollect (const return_value &__rval)
- return_value value ()
- return_value vvalue ()
- void vinit ()
- virtual int initialize (const node_data_type &__data)
- virtual void calculate (const node_data_type &__data)
- virtual void retrieve_from_cache (const node_data_type &__data)
- virtual void cleanup (const node_data_type &__data)
- virtual int update (const node_data_type &__data, const return_value &__rval)
- virtual int update (const return_value &__rval)
- virtual walker short_cut_to (const node_data_type &__data) PURE_VIRTUAL public

### Protected Methods

- bool is_cached (const node_data_type &__data)

**Protected Attributes**

- const variable_indicator * v_ind
- func_d_eval_type eval_data

### 6.47.1 Member Typedef Documentation

#### 6.47.1.1 typedef func_d_eval_type _evaluator_base< func_d_eval_type, expression_node, double, model::walker >::data_type [inherited]

Definition at line 245 of file evaluator.h.

#### 6.47.1.2 typedef cached_evaluator_base<func_d_eval_type, expression_node, double,model::walker>::node_data_type cached_forward_evaluator_base< func_d_eval_type, expression_node, double, model::walker >::node_data_type [inherited]

Reimplemented from cached_evaluator_base< func_d_eval_type, expression_node, double, model::walker >.

Definition at line 396 of file evaluator.h.

#### 6.47.1.3 typedef cached_evaluator_base<func_d_eval_type, expression_node, double,model::walker>::return_value cached_forward_evaluator_base< func_d_eval_type, expression_node, double, model::walker >::return_value [inherited]

Reimplemented from cached_evaluator_base< func_d_eval_type, expression_node, double, model::walker >.

Definition at line 398 of file evaluator.h.

#### 6.47.1.4 typedef cached_evaluator_base<func_d_eval_type, expression_node, double,model::walker>::walker cached_forward_evaluator_base< func_d_eval_type, expression_node, double, model::walker >::walker [inherited]

Reimplemented from cached_evaluator_base< func_d_eval_type, expression_node, double, model::walker >.

Definition at line 400 of file evaluator.h.

### 6.47.2 Constructor & Destructor Documentation

#### 6.47.2.1 func_d_eval::func_d_eval (const std::vector< double > & *__x*, const variable_indicator & *__v*, const model & *__m*, std::vector< std::vector< double > > & *__d*, std::vector< double > * *__c*) [inline]

Definition at line 188 of file der_evaluator.h.

#### 6.47.2.2 func_d_eval::func_d_eval (const func_d_eval & *__x*) [inline]

Definition at line 202 of file der_evaluator.h.

#### 6.47.2.3 func_d_eval::~func_d_eval () [inline]

Definition at line 204 of file der_evaluator.h.

### 6.47.3 Member Function Documentation

#### 6.47.3.1 virtual void cached_forward_evaluator_base< func_d_eval_type, expression_node, double, model::walker >::calculate (const node_data_type & *__data*) [inline, virtual, inherited]

Definition at line 430 of file evaluator.h.

#### 6.47.3.2 void func_d_eval::calculate (const expression_node & *__data*) [inline]

Definition at line 277 of file der_evaluator.h.

#### 6.47.3.3 double func_d_eval::calculate_value (bool *eval_all*) [inline, virtual]

Reimplemented from cached_forward_evaluator_base< func_d_eval_type, expression_node, double, model::walker >.

Definition at line 918 of file der_evaluator.h.

#### 6.47.3.4 virtual void cached_forward_evaluator_base< func_d_eval_type, expression_node, double, model::walker >::cleanup (const node_data_type & *__data*) [inline, virtual, inherited]

Definition at line 432 of file evaluator.h.

#### 6.47.3.5 int cached_forward_evaluator_base< func_d_eval_type, expression_node, double, model::walker >::collect (const node_data_type & *__data*, const return_value & *__rval*) [inline, virtual, inherited]

Reimplemented from _evaluator_base< func_d_eval_type, expression_node, double, model::walker >.

Definition at line 419 of file evaluator.h.

#### 6.47.3.6 virtual int cached_forward_evaluator_base< func_d_eval_type, expression_node, double, model::walker >::initialize (const node_data_type & *__data*) [inline, virtual, inherited]

Definition at line 429 of file evaluator.h.

#### 6.47.3.7 int func_d_eval::initialize (const expression_node & *__data*) [inline]

Definition at line 217 of file der_evaluator.h.

#### 6.47.3.8 void func_d_eval::initialize () [inline, virtual]

Reimplemented from cached_forward_evaluator_base< func_d_eval_type, expression_node, double, model::walker >.

Definition at line 215 of file der_evaluator.h.

#### 6.47.3.9 bool func_d_eval::is_cached (const node_data_type & *__data*) [inline, protected, virtual]

Reimplemented from cached_forward_evaluator_base< func_d_eval_type, expression_node, double, model::walker >.

Definition at line 118 of file der_evaluator.h.

#### 6.47.3.10 void func_d_eval::new_point (const std::vector< double > & *__x*, const variable_indicator & *__v*) [inline]

Definition at line 209 of file der_evaluator.h.

#### 6.47.3.11 void cached_forward_evaluator_base< func_d_eval_type, expression_node, double, model::walker >::postorder (const node_data_type & *__data*) [inline, virtual, inherited]

Reimplemented from _evaluator_base< func_d_eval_type, expression_node, double, model::walker >.

Definition at line 417 of file evaluator.h.

#### 6.47.3.12 int cached_forward_evaluator_base< func_d_eval_type, expression_node, double, model::walker >::preorder (const node_data_type & *__data*) [inline, virtual, inherited]

Reimplemented from cached_evaluator_base< func_d_eval_type, expression_node, double, model::walker >.

Definition at line 408 of file evaluator.h.

#### 6.47.3.13 virtual void cached_forward_evaluator_base< func_d_eval_type, expression_node, double, model::walker >::retrieve_from_cache (const node_data_type & *__data*) [inline, virtual, inherited]

Definition at line 431 of file evaluator.h.

#### 6.47.3.14 void func_d_eval::retrieve_from_cache (const expression_node & *__data*) [inline]

Definition at line 287 of file der_evaluator.h.

#### 6.47.3.15 virtual walker cached_evaluator_base< func_d_eval_type, expression_node, double, model::walker >::short_cut_to (const node_data_type & *__data*) [inline, virtual, inherited]

Definition at line 303 of file evaluator.h.

#### 6.47.3.16 model::walker func_d_eval::short_cut_to (const expression_node & *__data*) [inline]

Definition at line 206 of file der_evaluator.h.

#### 6.47.3.17 virtual int cached_forward_evaluator_base< func_d_eval_type, expression_node, double, model::walker >::update (const return_value & *__rval*) [inline, virtual, inherited]

Definition at line 435 of file evaluator.h.

#### 6.47.3.18 virtual int cached_forward_evaluator_base< func_d_eval_type, expression_node, double, model::walker >::update (const node_data_type & *__data*, const return_value & *__rval*) [inline, virtual, inherited]

Definition at line 433 of file evaluator.h.

#### 6.47.3.19 int func_d_eval::update (const expression_node & *__data*, const double & *__rval*) [inline]

Definition at line 320 of file der_evaluator.h.

#### 6.47.3.20 int func_d_eval::update (const double & *__rval*) [inline]

Definition at line 314 of file der_evaluator.h.

#### 6.47.3.21 return_value cached_forward_evaluator_base< func_d_eval_type, expression_node, double, model::walker >::value () [inline, virtual, inherited]

Reimplemented from _evaluator_base< func_d_eval_type, expression_node, double, model::walker >.

Definition at line 423 of file evaluator.h.

#### 6.47.3.22 int cached_forward_evaluator_base< func_d_eval_type, expression_node, double, model::walker >::vcollect (const return_value & *__rval*) [inline, virtual, inherited]

Reimplemented from _evaluator_base< func_d_eval_type, expression_node, double, model::walker >.

Definition at line 421 of file evaluator.h.

#### 6.47.3.23 void cached_forward_evaluator_base< func_d_eval_type, expression_node, double, model::walker >::vinit () [inline, inherited]

Definition at line 425 of file evaluator.h.

#### 6.47.3.24 return_value cached_forward_evaluator_base< func_d_eval_type, expression_node, double, model::walker >::vvalue () [inline, virtual, inherited]

Reimplemented from _evaluator_base< func_d_eval_type, expression_node, double, model::walker >.

Definition at line 424 of file evaluator.h.

### 6.47.4 Member Data Documentation

#### 6.47.4.1 func_d_eval_type _evaluator_base< func_d_eval_type, expression_node, double, model::walker >::eval_data [protected, inherited]

Definition at line 252 of file evaluator.h.

#### 6.47.4.2 const variable_indicator* cached_evaluator_base< func_d_eval_type, expression_node, double, model::walker >::v_ind [protected, inherited]

Definition at line 295 of file evaluator.h.

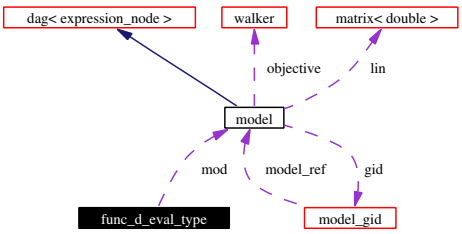
The documentation for this class was generated from the following file:

- der_evaluator.h

## 6.48 func_d_eval_type Struct Reference

`#include <der_evaluator.h>`

Collaboration diagram for func_d_eval_type:




### Public Attributes

- const std::vector< double > * x
- std::vector< double > * f_cache
- std::vector< std::vector< double > > * d_data
- const model * mod
- union {
  void * p
  double d
  } u
- double r
- unsigned int n
- unsigned int info

### 6.48.1 Member Data Documentation

#### 6.48.1.1 double func_d_eval_type::d

Definition at line 103 of file der_evaluator.h.

#### 6.48.1.2 std::vector<std::vector<double> >* func_d_eval_type::d_data

Definition at line 101 of file der_evaluator.h.

#### 6.48.1.3 std::vector<double>* func_d_eval_type::f_cache

Definition at line 100 of file der_evaluator.h.

#### 6.48.1.4 unsigned int func_d_eval_type::info

Definition at line 105 of file der_evaluator.h.

#### 6.48.1.5 const model* func_d_eval_type::mod

Definition at line 102 of file der_evaluator.h.

#### 6.48.1.6 unsigned int func_d_eval_type::n

Definition at line 105 of file der_evaluator.h.

#### 6.48.1.7 void* func_d_eval_type::p

Definition at line 103 of file der_evaluator.h.

#### 6.48.1.8 double func_d_eval_type::r

Definition at line 104 of file der_evaluator.h.

#### 6.48.1.9 union { ... } func_d_eval_type::u

#### 6.48.1.10 const std::vector<double>* func_d_eval_type::x

Definition at line 99 of file der_evaluator.h.

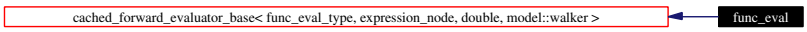
The documentation for this struct was generated from the following file:

- der_evaluator.h

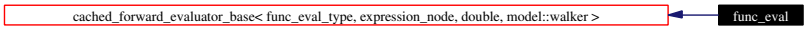
## 6.49 func_eval Class Reference

`#include <func_evaluator.h>`

Inheritance diagram for func_eval:




Collaboration diagram for func_eval:




### Public Types

- typedef cached_evaluator_base< func_eval_type, expression_node, double, model::walker >::node_data_type node_data_type
- typedef cached_evaluator_base< func_eval_type, expression_node, double, model::walker >::return_value return_value
- typedef cached_evaluator_base< func_eval_type, expression_node, double, model::walker >::walker walker
- typedef func_eval_type data_type

### Public Methods

- func_eval (const std::vector< double > &__x, const variable_indicator &__v, const model &__m, std::vector< double > *__c)
- func_eval (const func_eval &__v)

- ~func_eval ()
- model::walker short_cut_to (const expression_node &__data)
- void new_point (const std::vector< double > &__x, const variable_indicator &__v)
- void initialize ()
- int initialize (const expression_node &__data)
- void calculate (const expression_node &__data)
- void retrieve_from_cache (const expression_node &__data)
- int update (const double &__rval)
- int update (const expression_node &__data, const double &__rval)
- double calculate_value (bool eval_all)
- int preorder (const node_data_type &__data)
- void postorder (const node_data_type &__data)
- int collect (const node_data_type &__data, const return_value &__rval)
- int vcollect (const return_value &__rval)
- return_value value ()
- return_value vvalue ()
- void vinit ()
- virtual int initialize (const node_data_type &__data)
- virtual void calculate (const node_data_type &__data)
- virtual void retrieve_from_cache (const node_data_type &__data)
- virtual void cleanup (const node_data_type &__data)
- virtual int update (const node_data_type &__data, const return_value &__rval)
- virtual int update (const return_value &__rval)
- virtual walker short_cut_to (const node_data_type &__data) PURE_VIRTUAL public

**Protected Methods**

- bool is_cached (const node_data_type &__data)

**Protected Attributes**

- const variable_indicator * v_ind
- func_eval_type eval_data

### 6.49.1 Member Typedef Documentation

#### 6.49.1.1 typedef func_eval_type _evaluator_base< func_eval_type, expression_node, double, model::walker >::data_type [inherited]

Definition at line 245 of file evaluator.h.

#### 6.49.1.2 typedef cached_evaluator_base<func_eval_type, expression_node, double,model::walker>::node_data_type cached_forward_evaluator_base< func_eval_type, expression_node, double, model::walker >::node_data_type [inherited]

Reimplemented from cached_evaluator_base< func_eval_type, expression_node, double, model::walker >.

Definition at line 396 of file evaluator.h.

#### 6.49.1.3 typedef cached_evaluator_base<func_eval_type, expression_node, double,model::walker>::return_value cached_forward_evaluator_base< func_eval_type, expression_node, double, model::walker >::return_value [inherited]

Reimplemented from cached_evaluator_base< func_eval_type, expression_node, double, model::walker >.

Definition at line 398 of file evaluator.h.

#### 6.49.1.4 typedef cached_evaluator_base<func_eval_type, expression_node, double,model::walker>::walker cached_forward_evaluator_base< func_eval_type, expression_node, double, model::walker >::walker [inherited]

Reimplemented from cached_evaluator_base< func_eval_type, expression_node, double, model::walker >.

Definition at line 400 of file evaluator.h.

### 6.49.2 Constructor & Destructor Documentation

#### 6.49.2.1 func_eval::func_eval (const std::vector< double > & *__x*, const variable_indicator & *__v*, const model & *__m*, std::vector< double > * *__c*) [inline]

Definition at line 112 of file func_evaluator.h.

#### 6.49.2.2 func_eval::func_eval (const func_eval & *__v*) [inline]

Definition at line 123 of file func_evaluator.h.

#### 6.49.2.3 func_eval::~func_eval () [inline]

Definition at line 125 of file func_evaluator.h.

### 6.49.3 Member Function Documentation

#### 6.49.3.1 virtual void cached_forward_evaluator_base< func_eval_type, expression_node, double, model::walker >::calculate (const node_data_type & *__data*) [inline, virtual, inherited]

Definition at line 430 of file evaluator.h.

#### 6.49.3.2 void func_eval::calculate (const expression_node & *__data*) [inline]

Definition at line 199 of file func_evaluator.h.

#### 6.49.3.3 double func_eval::calculate_value (bool *eval_all*) [inline, virtual]

Reimplemented from cached_forward_evaluator_base< func_eval_type, expression_node, double, model::walker >.

Definition at line 583 of file func_evaluator.h.

#### 6.49.3.4 virtual void cached_forward_evaluator_base< func_eval_type, expression_node, double, model::walker >::cleanup (const node_data_type & *__data*) [inline, virtual, inherited]

Definition at line 432 of file evaluator.h.

#### 6.49.3.5 int cached_forward_evaluator_base< func_eval_type, expression_node, double, model::walker >::collect (const node_data_type & *__data*, const return_value & *__rval*) [inline, virtual, inherited]

Reimplemented from _evaluator_base< func_eval_type, expression_node, double, model::walker >.

Definition at line 419 of file evaluator.h.

#### 6.49.3.6 virtual int cached_forward_evaluator_base< func_eval_type, expression_node, double, model::walker >::initialize (const node_data_type & *__data*) [inline, virtual, inherited]

Definition at line 429 of file evaluator.h.

#### 6.49.3.7 int func_eval::initialize (const expression_node & *__data*) [inline]

Definition at line 139 of file func_evaluator.h.

#### 6.49.3.8 void func_eval::initialize () [inline, virtual]

Reimplemented from cached_forward_evaluator_base< func_eval_type, expression_node, double, model::walker >.

Definition at line 137 of file func_evaluator.h.

#### 6.49.3.9 bool func_eval::is_cached (const node_data_type & *__data*) [inline, protected, virtual]

Reimplemented from cached_forward_evaluator_base< func_eval_type, expression_node, double, model::walker >.

Definition at line 62 of file func_evaluator.h.

#### 6.49.3.10 void func_eval::new_point (const std::vector< double > & *__x*, const variable_indicator & *__v*) [inline]

Definition at line 131 of file func_evaluator.h.

#### 6.49.3.11 void cached_forward_evaluator_base< func_eval_type, expression_node, double, model::walker >::postorder (const node_data_type & *__data*) [inline, virtual, inherited]

Reimplemented from _evaluator_base< func_eval_type, expression_node, double, model::walker >.

Definition at line 417 of file evaluator.h.

#### 6.49.3.12 int cached_forward_evaluator_base< func_eval_type, expression_node, double, model::walker >::preorder (const node_data_type & *__data*) [inline, virtual, inherited]

Reimplemented from cached_evaluator_base< func_eval_type, expression_node, double, model::walker >.

Definition at line 408 of file evaluator.h.

#### 6.49.3.13 virtual void cached_forward_evaluator_base< func_eval_type, expression_node, double, model::walker >::retrieve_from_cache (const node_data_type & *__data*) [inline, virtual, inherited]

Definition at line 431 of file evaluator.h.

#### 6.49.3.14 void func_eval::retrieve_from_cache (const expression_node & *__data*) [inline]

Definition at line 208 of file func_evaluator.h.

#### 6.49.3.15 virtual walker cached_evaluator_base< func_eval_type, expression_node, double, model::walker >::short_cut_to (const node_data_type & *__data*) [inline, virtual, inherited]

Definition at line 303 of file evaluator.h.

#### 6.49.3.16 model::walker func_eval::short_cut_to (const expression_node & *__data*) [inline]

Definition at line 127 of file func_evaluator.h.

#### 6.49.3.17 virtual int cached_forward_evaluator_base< func_eval_type, expression_node, double, model::walker >::update (const return_value & *__rval*) [inline, virtual, inherited]

Definition at line 435 of file evaluator.h.

#### 6.49.3.18 virtual int cached_forward_evaluator_base< func_eval_type, expression_node, double, model::walker >::update (const node_data_type & *__data*, const return_value & *__rval*) [inline, virtual, inherited]

Definition at line 433 of file evaluator.h.

#### 6.49.3.19 int func_eval::update (const expression_node & *__data*, const double & *__rval*) [inline]

Definition at line 232 of file func_evaluator.h.

#### 6.49.3.20 int func_eval::update (const double & *__rval*) [inline]

Definition at line 226 of file func_evaluator.h.

#### 6.49.3.21 return_value cached_forward_evaluator_base< func_eval_type, expression_node, double, model::walker >::value () [inline, virtual, inherited]

Reimplemented from _evaluator_base< func_eval_type, expression_node, double, model::walker >.

Definition at line 423 of file evaluator.h.

#### 6.49.3.22 int cached_forward_evaluator_base< func_eval_type, expression_node, double, model::walker >::vcollect (const return_value & *__rval*) [inline, virtual, inherited]

Reimplemented from _evaluator_base< func_eval_type, expression_node, double, model::walker >.

Definition at line 421 of file evaluator.h.

#### 6.49.3.23 void cached_forward_evaluator_base< func_eval_type, expression_node, double, model::walker >::vinit () [inline, inherited]

Definition at line 425 of file evaluator.h.

#### 6.49.3.24 return_value cached_forward_evaluator_base< func_eval_type, expression_node, double, model::walker >::vvalue () [inline, virtual, inherited]

Reimplemented from _evaluator_base< func_eval_type, expression_node, double, model::walker >.

Definition at line 424 of file evaluator.h.

### 6.49.4 Member Data Documentation

#### 6.49.4.1 func_eval_type _evaluator_base< func_eval_type, expression_node, double, model::walker >::eval_data [protected, inherited]

Definition at line 252 of file evaluator.h.

#### 6.49.4.2 const variable_indicator* cached_evaluator_base< func_eval_type, expression_node, double, model::walker >::v_ind [protected, inherited]

Definition at line 295 of file evaluator.h.

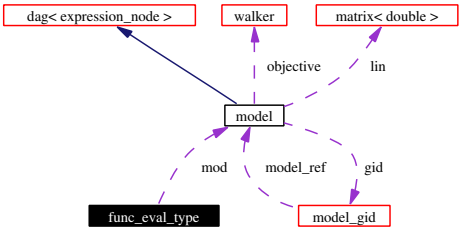
The documentation for this class was generated from the following file:

- func_evaluator.h

## 6.50 func_eval_type Struct Reference

`#include <func_evaluator.h>`

Collaboration diagram for func_eval_type:




### Public Attributes

- const std::vector< double > * x
- std::vector< double > * cache
- const model * mod
- union {
  void * p
  double d
  } u

- double r
- unsigned int n

### 6.50.1 Member Data Documentation

#### 6.50.1.1 std::vector<double>* func_eval_type::cache

Definition at line 46 of file func_evaluator.h.

#### 6.50.1.2 double func_eval_type::d

Definition at line 48 of file func_evaluator.h.

#### 6.50.1.3 const model* func_eval_type::mod

Definition at line 47 of file func_evaluator.h.

#### 6.50.1.4 unsigned int func_eval_type::n

Definition at line 50 of file func_evaluator.h.

#### 6.50.1.5 void* func_eval_type::p

Definition at line 48 of file func_evaluator.h.

#### 6.50.1.6 double func_eval_type::r

Definition at line 49 of file func_evaluator.h.

#### 6.50.1.7 union { ... } func_eval_type::u

#### 6.50.1.8 const std::vector<double>* func_eval_type::x

Definition at line 45 of file func_evaluator.h.

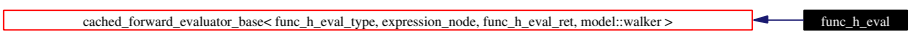
The documentation for this struct was generated from the following file:

- func_evaluator.h

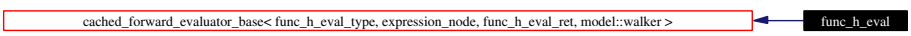
## 6.51 func_h_eval Class Reference

`#include <hess_evaluator.h>`

Inheritance diagram for func_h_eval:




Collaboration diagram for func_h_eval:



### Public Types

- typedef cached_evaluator_base< func_h_eval_type, expression_node, func_h_eval_ret, model::walker >::node_data_type node_data_type
- typedef cached_evaluator_base< func_h_eval_type, expression_node, func_h_eval_ret, model::walker >::return_value return_value
- typedef cached_evaluator_base< func_h_eval_type, expression_node, func_h_eval_ret, model::walker >::walker walker
- typedef func_h_eval_type data_type

### Public Methods

- func_h_eval (const std::vector< double > &__x, const variable_indicator &__v, const model &__m, std::vector< std::vector< double > > &__d, std::vector< matrix< double > > &__h, std::vector< unsigned int > &__t, std::vector< bool > &__b, std::vector< double > *__c)
- func_h_eval (const func_h_eval &__x)
- ~func_h_eval ()
- model::walker short_cut_to (const expression_node &__data)
- void new_point (const std::vector< double > &__x, const variable_indicator &__v)
- void initialize ()
- int initialize (const expression_node &__data)
- void calculate (const expression_node &__data)
- void retrieve_from_cache (const expression_node &__data)
- int update (const func_h_eval_ret &__rval)
- int update (const expression_node &__data, const func_h_eval_ret &__rval)
- func_h_eval_ret calculate_value (bool eval_all)
- int preorder (const node_data_type &__data)
- void postorder (const node_data_type &__data)
- int collect (const node_data_type &__data, const return_value &__rval)
- int vcollect (const return_value &__rval)
- return_value value ()
- return_value vvalue ()
- void vinit ()
- virtual int initialize (const node_data_type &__data)
- virtual void calculate (const node_data_type &__data)
- virtual void retrieve_from_cache (const node_data_type &__data)
- virtual void cleanup (const node_data_type &__data)
- virtual int update (const node_data_type &__data, const return_value &__rval)
- virtual int update (const return_value &__rval)
- virtual walker short_cut_to (const node_data_type &__data) PURE_VIRTUAL public

### Protected Methods

- bool is_cached (const node_data_type &__data)

### Protected Attributes

- const variable_indicator * v_ind
- func_h_eval_type eval_data

### 6.51.1 Member Typedef Documentation

#### 6.51.1.1 typedef func_h_eval_type _evaluator_base< func_h_eval_type, expression_node, func_h_eval_ret, model::walker >::data_type [inherited]

Definition at line 245 of file evaluator.h.

#### 6.51.1.2 typedef cached_evaluator_base<func_h_eval_type, expression_node, func_h_eval_ret,model::walker>::node_data_type cached_forward_evaluator_base< func_h_eval_type, expression_node, func_h_eval_ret, model::walker >::node_data_type [inherited]

Reimplemented from cached_evaluator_base< func_h_eval_type, expression_node, func_h_eval_ret, model::walker >.

Definition at line 396 of file evaluator.h.

#### 6.51.1.3 typedef cached_evaluator_base<func_h_eval_type, expression_node, func_h_eval_ret,model::walker>::return_value cached_forward_evaluator_base< func_h_eval_type, expression_node, func_h_eval_ret, model::walker >::return_value [inherited]

Reimplemented from cached_evaluator_base< func_h_eval_type, expression_node, func_h_eval_ret, model::walker >.

Definition at line 398 of file evaluator.h.

#### 6.51.1.4 typedef cached_evaluator_base<func_h_eval_type, expression_node, func_h_eval_ret,model::walker>::walker cached_forward_evaluator_base< func_h_eval_type, expression_node, func_h_eval_ret, model::walker >::walker [inherited]

Reimplemented from cached_evaluator_base< func_h_eval_type, expression_node, func_h_eval_ret, model::walker >.

Definition at line 400 of file evaluator.h.

### 6.51.2 Constructor & Destructor Documentation

#### 6.51.2.1 func_h_eval::func_h_eval (const std::vector< double > & *__x*, const variable_indicator & *__v*, const model & *__m*, std::vector< std::vector< double > > & *__d*, std::vector< matrix< double > > & *__h*, std::vector< unsigned int > & *__t*, std::vector< bool > & *__b*, std::vector< double > * *__c*) [inline]

Definition at line 214 of file hess_evaluator.h.

#### 6.51.2.2 func_h_eval::func_h_eval (const func_h_eval & *__x*) [inline]

Definition at line 233 of file hess_evaluator.h.

#### 6.51.2.3 func_h_eval::~func_h_eval () [inline]

Definition at line 235 of file hess_evaluator.h.

### 6.51.3 Member Function Documentation

#### 6.51.3.1 virtual void cached_forward_evaluator_base< func_h_eval_type, expression_node, func_h_eval_ret, model::walker >::calculate (const node_data_type & *__data*) [inline, virtual, inherited]

Definition at line 430 of file evaluator.h.

#### 6.51.3.2 void func_h_eval::calculate (const expression_node & *__data*) [inline]

Definition at line 310 of file hess_evaluator.h.

#### 6.51.3.3 func_h_eval_ret func_h_eval::calculate_value (bool *eval_all*) [inline, virtual]

Reimplemented from cached_forward_evaluator_base< func_h_eval_type, expression_node, func_h_eval_ret, model::walker >.

Definition at line 968 of file hess_evaluator.h.

#### 6.51.3.4 virtual void cached_forward_evaluator_base< func_h_eval_type, expression_node, func_h_eval_ret, model::walker >::cleanup (const node_data_type & *__data*) [inline, virtual, inherited]

Definition at line 432 of file evaluator.h.

#### 6.51.3.5 int cached_forward_evaluator_base< func_h_eval_type, expression_node, func_h_eval_ret, model::walker >::collect (const node_data_type & *__data*, const return_value & *__rval*) [inline, virtual, inherited]

Reimplemented from _evaluator_base< func_h_eval_type, expression_node, func_h_eval_ret, model::walker >.

Definition at line 419 of file evaluator.h.

#### 6.51.3.6 virtual int cached_forward_evaluator_base< func_h_eval_type, expression_node, func_h_eval_ret, model::walker >::initialize (const node_data_type & *__data*) [inline, virtual, inherited]

Definition at line 429 of file evaluator.h.

#### 6.51.3.7 int func_h_eval::initialize (const expression_node & *__data*) [inline]

Definition at line 248 of file hess_evaluator.h.

#### 6.51.3.8 void func_h_eval::initialize () [inline, virtual]

Reimplemented from cached_forward_evaluator_base< func_h_eval_type, expression_node, func_h_eval_ret, model::walker >.

Definition at line 246 of file hess_evaluator.h.

#### 6.51.3.9 bool func_h_eval::is_cached (const node_data_type & *__data*) [inline, protected, virtual]

Reimplemented from cached_forward_evaluator_base< func_h_eval_type, expression_node, func_h_eval_ret, model::walker >.

Definition at line 144 of file hess_evaluator.h.

#### 6.51.3.10 void func_h_eval::new_point (const std::vector< double > & *__x*, const variable_indicator & *__v*) [inline]

Definition at line 240 of file hess_evaluator.h.

#### 6.51.3.11 void cached_forward_evaluator_base< func_h_eval_type, expression_node, func_h_eval_ret, model::walker >::postorder (const node_data_type & *__data*) [inline, virtual, inherited]

Reimplemented from _evaluator_base< func_h_eval_type, expression_node, func_h_eval_ret, model::walker >.

Definition at line 417 of file evaluator.h.

#### 6.51.3.12 int cached_forward_evaluator_base< func_h_eval_type, expression_node, func_h_eval_ret, model::walker >::preorder (const node_data_type & *__data*) [inline, virtual, inherited]

Reimplemented from cached_evaluator_base< func_h_eval_type, expression_node, func_h_eval_ret, model::walker >.

Definition at line 408 of file evaluator.h.

#### 6.51.3.13 virtual void cached_forward_evaluator_base< func_h_eval_type, expression_node, func_h_eval_ret, model::walker >::retrieve_from_cache (const node_data_type & *__data*) [inline, virtual, inherited]

Definition at line 431 of file evaluator.h.

#### 6.51.3.14 void func_h_eval::retrieve_from_cache (const expression_node & *__data*) [inline]

Definition at line 326 of file hess_evaluator.h.

#### 6.51.3.15 virtual walker cached_evaluator_base< func_h_eval_type, expression_node, func_h_eval_ret, model::walker >::short_cut_to (const node_data_type & *__data*) [inline, virtual, inherited]

Definition at line 303 of file evaluator.h.

#### 6.51.3.16 model::walker func_h_eval::short_cut_to (const expression_node & *__data*) [inline]

Definition at line 237 of file hess_evaluator.h.

#### 6.51.3.17 virtual int cached_forward_evaluator_base< func_h_eval_type, expression_node, func_h_eval_ret, model::walker >::update (const return_value & *__rval*) [inline, virtual, inherited]

Definition at line 435 of file evaluator.h.

#### 6.51.3.18 virtual int cached_forward_evaluator_base< func_h_eval_type, expression_node, func_h_eval_ret, model::walker >::update (const node_data_type & __data, const return_value & __rval) [inline, virtual, inherited]

Definition at line 433 of file evaluator.h.

#### 6.51.3.19 int func_h_eval::update (const expression_node & __data, const func_h_eval_ret & __rval) [inline]

Definition at line 363 of file hess_evaluator.h.

#### 6.51.3.20 int func_h_eval::update (const func_h_eval_ret & __rval) [inline]

Definition at line 355 of file hess_evaluator.h.

#### 6.51.3.21 return_value cached_forward_evaluator_base< func_h_eval_type, expression_node, func_h_eval_ret, model::walker >::value () [inline, virtual, inherited]

Reimplemented from _evaluator_base< func_h_eval_type, expression_node, func_h_eval_ret, model::walker >.

Definition at line 423 of file evaluator.h.

#### 6.51.3.22 int cached_forward_evaluator_base< func_h_eval_type, expression_node, func_h_eval_ret, model::walker >::vcollect (const return_value & __rval) [inline, virtual, inherited]

Reimplemented from _evaluator_base< func_h_eval_type, expression_node, func_h_eval_ret, model::walker >.

Definition at line 421 of file evaluator.h.

#### 6.51.3.23 void cached_forward_evaluator_base< func_h_eval_type, expression_node, func_h_eval_ret, model::walker >::vinit () [inline, inherited]

Definition at line 425 of file evaluator.h.

#### 6.51.3.24 return_value cached_forward_evaluator_base< func_h_eval_type, expression_node, func_h_eval_ret, model::walker >::vvalue () [inline, virtual, inherited]

Reimplemented from _evaluator_base< func_h_eval_type, expression_node, func_h_eval_ret, model::walker >.

Definition at line 424 of file evaluator.h.

### 6.51.4 Member Data Documentation

#### 6.51.4.1 func_h_eval_type _evaluator_base< func_h_eval_type, expression_node, func_h_eval_ret, model::walker >::eval_data [protected, inherited]

Definition at line 252 of file evaluator.h.

#### 6.51.4.2 const variable_indicator* cached_evaluator_base< func_h_eval_type, expression_node, func_h_eval_ret, model::walker >::v_ind [protected, inherited]

Definition at line 295 of file evaluator.h.

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

- hess_evaluator.h

## 6.52 func_h_eval_ret Struct Reference

`#include <hess_evaluator.h>`

### Public Attributes

- double f
- unsigned int nn
- tristate in_chn

### 6.52.1 Member Data Documentation

#### 6.52.1.1 double func_h_eval_ret::f

Definition at line 115 of file hess_evaluator.h.

#### 6.52.1.2 tristate func_h_eval_ret::in_chn

Definition at line 117 of file hess_evaluator.h.

#### 6.52.1.3 unsigned int func_h_eval_ret::nn

Definition at line 116 of file hess_evaluator.h.

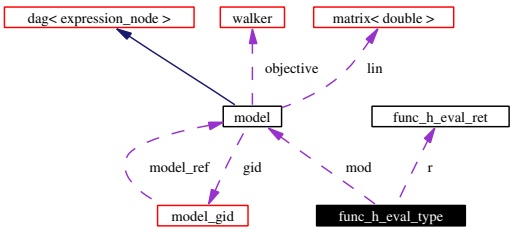
The documentation for this struct was generated from the following file:

- hess_evaluator.h

## 6.53 func_h_eval_type Struct Reference

`#include <hess_evaluator.h>`

Collaboration diagram for func_h_eval_type:



### Public Attributes

- const std::vector< double > * x
- std::vector< double > * f_cache
- std::vector< std::vector< double > > * d_data
- std::vector< matrix< double > > * h_data
- vector< unsigned int > * t
- vector< bool > * b
- func_h_eval_ret r
- const model * mod
- union {
  void * p
  double d
  } u
- unsigned int n
- unsigned int info

## 6.53.1 Member Data Documentation

### 6.53.1.1 vector<bool>* func_h_eval_type::b

Definition at line 127 of file hess_evaluator.h.

### 6.53.1.2 double func_h_eval_type::d

Definition at line 130 of file hess_evaluator.h.

### 6.53.1.3 std::vector<std::vector<double> >* func_h_eval_type::d_data

Definition at line 124 of file hess_evaluator.h.

### 6.53.1.4 std::vector<double>* func_h_eval_type::f_cache

Definition at line 123 of file hess_evaluator.h.

### 6.53.1.5 std::vector<matrix<double> >* func_h_eval_type::h_data

Definition at line 125 of file hess_evaluator.h.

### 6.53.1.6 unsigned int func_h_eval_type::info

Definition at line 131 of file hess_evaluator.h.

### 6.53.1.7 const model* func_h_eval_type::mod

Definition at line 129 of file hess_evaluator.h.

### 6.53.1.8 unsigned int func_h_eval_type::n

Definition at line 131 of file hess_evaluator.h.

#### 6.53.1.9 void* func_h_eval_type::p

Definition at line 130 of file hess_evaluator.h.

#### 6.53.1.10 func_h_eval_ret func_h_eval_type::r

Definition at line 128 of file hess_evaluator.h.

#### 6.53.1.11 vector<unsigned int>* func_h_eval_type::t

Definition at line 126 of file hess_evaluator.h.

#### 6.53.1.12 union { ... } func_h_eval_type::u

#### 6.53.1.13 const std::vector<double>* func_h_eval_type::x

Definition at line 122 of file hess_evaluator.h.

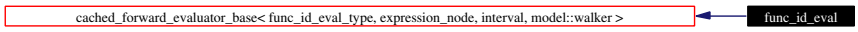
The documentation for this struct was generated from the following file:

- hess_evaluator.h

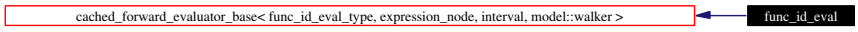
## 6.54 func_id_eval Class Reference

`#include <ider_evaluator.h>`

Inheritance diagram for func_id_eval:




Collaboration diagram for func_id_eval:




### Public Types

- typedef cached_evaluator_base< func_id_eval_type, expression_node, interval, model::walker >::node_data_type node_data_type
- typedef cached_evaluator_base< func_id_eval_type, expression_node, interval, model::walker >::return_value return_value
- typedef cached_evaluator_base< func_id_eval_type, expression_node, interval, model::walker >::walker walker
- typedef func_id_eval_type data_type

### Public Methods

- func_id_eval (const std::vector< interval > &__x, const std::vector< interval > &__rg, const variable_indicator &__v, const model &__m, std::vector< std::vector< interval > > &__d, std::vector< interval > *__c)
- func_id_eval (const func_id_eval &__x)

- ~func_id_eval ()
- model::walker short_cut_to (const expression_node &__data)
- void new_box (const std::vector< interval > &__x, const variable_indicator &__v)
- void new_range (const std::vector< interval > &__rg, const variable_indicator &__v)
- void initialize ()
- int initialize (const expression_node &__data)
- void calculate (const expression_node &__data)
- void retrieve_from_cache (const expression_node &__data)
- int update (const interval &__rval)
- int update (const expression_node &__data, const interval &__rval)
- interval calculate_value (bool eval_all)
- int preorder (const node_data_type &__data)
- void postorder (const node_data_type &__data)
- int collect (const node_data_type &__data, const return_value &__rval)
- int vcollect (const return_value &__rval)
- return_value value ()
- return_value vvalue ()
- void vinit ()
- virtual int initialize (const node_data_type &__data)
- virtual void calculate (const node_data_type &__data)
- virtual void retrieve_from_cache (const node_data_type &__data)
- virtual void cleanup (const node_data_type &__data)
- virtual int update (const node_data_type &__data, const return_value &__rval)
- virtual int update (const return_value &__rval)
- virtual walker short_cut_to (const node_data_type &__data) PURE_VIRTUAL public

**Protected Methods**

- bool is_cached (const node_data_type &__data)

**Protected Attributes**

- const variable_indicator * v_ind
- func_id_eval_type eval_data

### 6.54.1 Member Typedef Documentation

#### 6.54.1.1 typedef func_id_eval_type _evaluator_base< func_id_eval_type, expression_node, interval, model::walker >::data_type [inherited]

Definition at line 245 of file evaluator.h.

#### 6.54.1.2 typedef cached_evaluator_base<func_id_eval_type, expression_node, interval,model::walker>::node_data_type cached_forward_evaluator_base< func_id_eval_type, expression_node, interval, model::walker >::node_data_type [inherited]

Reimplemented from cached_evaluator_base< func_id_eval_type, expression_node, interval, model::walker >.

Definition at line 396 of file evaluator.h.

#### 6.54.1.3 typedef cached_evaluator_base<func_id_eval_type, expression_node, interval,model::walker>::return_value cached_forward_evaluator_base< func_id_eval_type, expression_node, interval, model::walker >::return_value [inherited]

Reimplemented from cached_evaluator_base< func_id_eval_type, expression_node, interval, model::walker >.

Definition at line 398 of file evaluator.h.

#### 6.54.1.4 typedef cached_evaluator_base<func_id_eval_type, expression_node, interval,model::walker>::walker cached_forward_evaluator_base< func_id_eval_type, expression_node, interval, model::walker >::walker [inherited]

Reimplemented from cached_evaluator_base< func_id_eval_type, expression_node, interval, model::walker >.

Definition at line 400 of file evaluator.h.

### 6.54.2 Constructor & Destructor Documentation

#### 6.54.2.1 func_id_eval::func_id_eval (const std::vector< interval > & *__x*, const std::vector< interval > & *__rg*, const variable_indicator & *__v*, const model & *__m*, std::vector< std::vector< interval > > & *__d*, std::vector< interval > * *__c*) [inline]

Definition at line 188 of file ider_evaluator.h.

#### 6.54.2.2 func_id_eval::func_id_eval (const func_id_eval & *__x*) [inline]

Definition at line 206 of file ider_evaluator.h.

#### 6.54.2.3 func_id_eval::~func_id_eval () [inline]

Definition at line 208 of file ider_evaluator.h.

### 6.54.3 Member Function Documentation

#### 6.54.3.1 virtual void cached_forward_evaluator_base< func_id_eval_type, expression_node, interval, model::walker >::calculate (const node_data_type & *__data*) [inline, virtual, inherited]

Definition at line 430 of file evaluator.h.

#### 6.54.3.2 void func_id_eval::calculate (const expression_node & *__data*) [inline]

Definition at line 292 of file ider_evaluator.h.

#### 6.54.3.3 interval func_id_eval::calculate_value (bool *eval_all*) [inline, virtual]

Reimplemented from cached_forward_evaluator_base< func_id_eval_type, expression_node, interval, model::walker >.

Definition at line 1205 of file ider_evaluator.h.

#### 6.54.3.4 virtual void cached_forward_evaluator_base< func_id_eval_type, expression_node, interval, model::walker >::cleanup (const node_data_type & *__data*) [inline, virtual, inherited]

Definition at line 432 of file evaluator.h.

#### 6.54.3.5 int cached_forward_evaluator_base< func_id_eval_type, expression_node, interval, model::walker >::collect (const node_data_type & *__data*, const return_value & *__rval*) [inline, virtual, inherited]

Reimplemented from _evaluator_base< func_id_eval_type, expression_node, interval, model::walker >.

Definition at line 419 of file evaluator.h.

#### 6.54.3.6 virtual int cached_forward_evaluator_base< func_id_eval_type, expression_node, interval, model::walker >::initialize (const node_data_type & *__data*) [inline, virtual, inherited]

Definition at line 429 of file evaluator.h.

#### 6.54.3.7 int func_id_eval::initialize (const expression_node & *__data*) [inline]

Definition at line 227 of file ider_evaluator.h.

#### 6.54.3.8 void func_id_eval::initialize () [inline, virtual]

Reimplemented from cached_forward_evaluator_base< func_id_eval_type, expression_node, interval, model::walker >.

Definition at line 225 of file ider_evaluator.h.

#### 6.54.3.9 bool func_id_eval::is_cached (const node_data_type & *__data*) [inline, protected, virtual]

Reimplemented from cached_forward_evaluator_base< func_id_eval_type, expression_node, interval, model::walker >.

Definition at line 119 of file ider_evaluator.h.

#### 6.54.3.10 void func_id_eval::new_box (const std::vector< interval > & *__x*, const variable_indicator & *__v*) [inline]

Definition at line 213 of file ider_evaluator.h.

#### 6.54.3.11 void func_id_eval::new_range (const std::vector< interval > & *__rg*, const variable_indicator & *__v*) [inline]

Definition at line 219 of file ider_evaluator.h.

#### 6.54.3.12 void cached_forward_evaluator_base< func_id_eval_type, expression_node, interval, model::walker >::postorder (const node_data_type & *__data*) [inline, virtual, inherited]

Reimplemented from _evaluator_base< func_id_eval_type, expression_node, interval, model::walker >.

Definition at line 417 of file evaluator.h.

#### 6.54.3.13 int cached_forward_evaluator_base< func_id_eval_type, expression_node, interval, model::walker >::preorder (const node_data_type & *__data*) [inline, virtual, inherited]

Reimplemented from cached_evaluator_base< func_id_eval_type, expression_node, interval, model::walker >.

Definition at line 408 of file evaluator.h.

#### 6.54.3.14 virtual void cached_forward_evaluator_base< func_id_eval_type, expression_node, interval, model::walker >::retrieve_from_cache (const node_data_type & *__data*) [inline, virtual, inherited]

Definition at line 431 of file evaluator.h.

#### 6.54.3.15 void func_id_eval::retrieve_from_cache (const expression_node & *__data*) [inline]

Definition at line 305 of file ider_evaluator.h.

#### 6.54.3.16 virtual walker cached_evaluator_base< func_id_eval_type, expression_node, interval, model::walker >::short_cut_to (const node_data_type & *__data*) [inline, virtual, inherited]

Definition at line 303 of file evaluator.h.

#### 6.54.3.17 model::walker func_id_eval::short_cut_to (const expression_node & *__data*) [inline]

Definition at line 210 of file ider_evaluator.h.

#### 6.54.3.18 virtual int cached_forward_evaluator_base< func_id_eval_type, expression_node, interval, model::walker >::update (const return_value & *__rval*) [inline, virtual, inherited]

Definition at line 435 of file evaluator.h.

#### 6.54.3.19 virtual int cached_forward_evaluator_base< func_id_eval_type, expression_node, interval, model::walker >::update (const node_data_type & *__data*, const return_value & *__rval*) [inline, virtual, inherited]

Definition at line 433 of file evaluator.h.

#### 6.54.3.20 int func_id_eval::update (const expression_node & *__data*, const interval & *__rval*) [inline]

Definition at line 339 of file ider_evaluator.h.

#### 6.54.3.21 int func_id_eval::update (const interval & *__rval*) [inline]

Definition at line 333 of file ider_evaluator.h.

#### 6.54.3.22 return_value cached_forward_evaluator_base< func_id_eval_type, expression_node, interval, model::walker >::value () [inline, virtual, inherited]

Reimplemented from _evaluator_base< func_id_eval_type, expression_node, interval, model::walker >.

Definition at line 423 of file evaluator.h.

#### 6.54.3.23 int cached_forward_evaluator_base< func_id_eval_type, expression_node, interval, model::walker >::vcollect (const return_value & *__rval*) [inline, virtual, inherited]

Reimplemented from _evaluator_base< func_id_eval_type, expression_node, interval, model::walker >.

Definition at line 421 of file evaluator.h.

#### 6.54.3.24 void cached_forward_evaluator_base< func_id_eval_type, expression_node, interval, model::walker >::vinit () [inline, inherited]

Definition at line 425 of file evaluator.h.

#### 6.54.3.25 return_value cached_forward_evaluator_base< func_id_eval_type, expression_node, interval, model::walker >::vvalue () [inline, virtual, inherited]

Reimplemented from _evaluator_base< func_id_eval_type, expression_node, interval, model::walker >.

Definition at line 424 of file evaluator.h.

### 6.54.4 Member Data Documentation

#### 6.54.4.1 func_id_eval_type _evaluator_base< func_id_eval_type, expression_node, interval, model::walker >::eval_data [protected, inherited]

Definition at line 252 of file evaluator.h.

#### 6.54.4.2 const variable_indicator* cached_evaluator_base< func_id_eval_type, expression_node, interval, model::walker >::v_ind [protected, inherited]

Definition at line 295 of file evaluator.h.

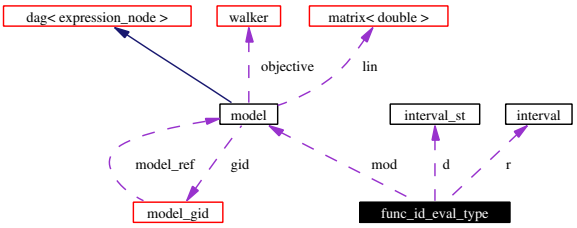
The documentation for this class was generated from the following file:

- ider_evaluator.h

## 6.55 func_id_eval_type Struct Reference

`#include <ider_evaluator.h>`

Collaboration diagram for func_id_eval_type:




## Public Attributes

- const std::vector< interval > * x
- const std::vector< interval > * range
- std::vector< interval > * f
- std::vector< std::vector< interval > > * id_data
- const model * mod
- union {
  void * p
  interval_st d
  unsigned int info
  } u
- interval r
- unsigned int n

### 6.55.1 Member Data Documentation

#### 6.55.1.1 interval_st func_id_eval_type::d

Definition at line 105 of file ider_evaluator.h.

#### 6.55.1.2 std::vector<interval>* func_id_eval_type::f

Definition at line 102 of file ider_evaluator.h.

#### 6.55.1.3 std::vector<std::vector<interval> >* func_id_eval_type::id_data

Definition at line 103 of file ider_evaluator.h.

#### 6.55.1.4 unsigned int func_id_eval_type::info

Definition at line 105 of file ider_evaluator.h.

#### 6.55.1.5 const model* func_id_eval_type::mod

Definition at line 104 of file ider_evaluator.h.

#### 6.55.1.6 unsigned int func_id_eval_type::n

Definition at line 107 of file ider_evaluator.h.

#### 6.55.1.7 void* func_id_eval_type::p

Definition at line 105 of file ider_evaluator.h.

#### 6.55.1.8 interval func_id_eval_type::r

Definition at line 106 of file ider_evaluator.h.

#### 6.55.1.9 const std::vector<interval>* func_id_eval_type::range

Definition at line 101 of file ider_evaluator.h.

#### 6.55.1.10 union { ... } func_id_eval_type::u

#### 6.55.1.11 const std::vector<interval>* func_id_eval_type::x

Definition at line 100 of file ider_evaluator.h.

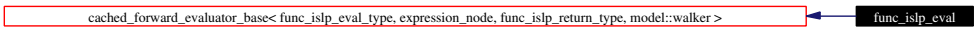
The documentation for this struct was generated from the following file:

- ider_evaluator.h

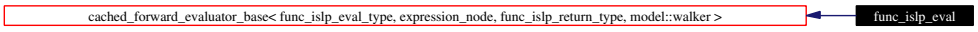
## 6.56 func_islp_eval Class Reference

`#include <islp_evaluator.h>`

Inheritance diagram for func_islp_eval:




Collaboration diagram for func_islp_eval:




### Public Types

- typedef cached_evaluator_base< func_islp_eval_type, expression_node, func_islp_return_type, model::walker >::node_data_type node_data_type
- typedef cached_evaluator_base< func_islp_eval_type, expression_node, func_islp_return_type, model::walker >::return_value return_value
- typedef cached_evaluator_base< func_islp_eval_type, expression_node, func_islp_return_type, model::walker >::walker walker
- typedef func_islp_eval_type data_type

### Public Methods

- func_islp_eval (const std::vector< double > &__z, const std::vector< interval > &__rg, const variable_indicator &__v, const model &__m, std::vector< std::vector< interval > > &__d, std::vector< double > &__f)
- func_islp_eval (const func_islp_eval &__x)

- ~func_islp_eval ()
- model::walker short_cut_to (const expression_node &__data)
- void new_point (const std::vector< double > &__x, const variable_indicator &__v)
- void new_range (const std::vector< interval > &__rg, const variable_indicator &__v)
- void initialize ()
- int initialize (const expression_node &__data)
- void calculate (const expression_node &__data)
- void retrieve_from_cache (const expression_node &__data)
- int update (const func_islp_return_type &__rval)
- int update (const expression_node &__data, const func_islp_return_type &__rval)
- func_islp_return_type calculate_value (bool eval_all)
- int preorder (const node_data_type &__data)
- void postorder (const node_data_type &__data)
- int collect (const node_data_type &__data, const return_value &__rval)
- int vcollect (const return_value &__rval)
- return_value value ()
- return_value vvalue ()
- void vinit ()
- virtual int initialize (const node_data_type &__data)
- virtual void calculate (const node_data_type &__data)
- virtual void retrieve_from_cache (const node_data_type &__data)
- virtual void cleanup (const node_data_type &__data)
- virtual int update (const node_data_type &__data, const return_value &__rval)
- virtual int update (const return_value &__rval)
- virtual walker short_cut_to (const node_data_type &__data) PURE_VIRTUAL public

### Protected Methods

- bool is_cached (const node_data_type &__data)

### Protected Attributes

- const variable_indicator * v_ind
- func_islp_eval_type eval_data

## 6.56.1 Member Typedef Documentation

### 6.56.1.1 typedef func_islp_eval_type _evaluator_base< func_islp_eval_type, expression_node, func_islp_return_type, model::walker >::data_type [inherited]

Definition at line 245 of file evaluator.h.

### 6.56.1.2 typedef cached_evaluator_base<func_islp_eval_type, expression_node, func_islp_return_type,model::walker>::node_data_type cached_forward_evaluator_base< func_islp_eval_type, expression_node, func_islp_return_type, model::walker >::node_data_type [inherited]

Reimplemented from cached_evaluator_base< func_islp_eval_type, expression_node, func_islp_return_type, model::walker >.

Definition at line 396 of file evaluator.h.

#### 6.56.1.3 typedef cached_evaluator_base<func_islp_eval_type, expression_node, func_islp_return_type,model::walker>::return_value cached_forward_evaluator_base< func_islp_eval_type, expression_node, func_islp_return_type, model::walker >::return_value [inherited]

Reimplemented from cached_evaluator_base< func_islp_eval_type, expression_node, func_islp_return_type, model::walker >.

Definition at line 398 of file evaluator.h.

#### 6.56.1.4 typedef cached_evaluator_base<func_islp_eval_type, expression_node, func_islp_return_type,model::walker>::walker cached_forward_evaluator_base< func_islp_eval_type, expression_node, func_islp_return_type, model::walker >::walker [inherited]

Reimplemented from cached_evaluator_base< func_islp_eval_type, expression_node, func_islp_return_type, model::walker >.

Definition at line 400 of file evaluator.h.

### 6.56.2 Constructor & Destructor Documentation

#### 6.56.2.1 func_islp_eval::func_islp_eval (const std::vector< double > & *__z*, const std::vector< interval > & *__rg*, const variable_indicator & *__v*, const model & *__m*, std::vector< std::vector< interval > > & *__d*, std::vector< double > & *__f*) [inline]

Definition at line 142 of file islp_evaluator.h.

#### 6.56.2.2 func_islp_eval::func_islp_eval (const func_islp_eval & *__x*) [inline]

Definition at line 160 of file islp_evaluator.h.

#### 6.56.2.3 func_islp_eval::~func_islp_eval () [inline]

Definition at line 162 of file islp_evaluator.h.

### 6.56.3 Member Function Documentation

#### 6.56.3.1 virtual void cached_forward_evaluator_base< func_islp_eval_type, expression_node, func_islp_return_type, model::walker >::calculate (const node_data_type & *__data*) [inline, virtual, inherited]

Definition at line 430 of file evaluator.h.

#### 6.56.3.2 void func_islp_eval::calculate (const expression_node & *__data*) [inline]

Definition at line 248 of file islp_evaluator.h.

#### 6.56.3.3 func_islp_return_type func_islp_eval::calculate_value (bool *eval_all*) [inline, virtual]

Reimplemented from cached_forward_evaluator_base< func_islp_eval_type, expression_node, func_islp_return_type, model::walker >.

Definition at line 1377 of file islp_evaluator.h.

#### 6.56.3.4 virtual void cached_forward_evaluator_base< func_islp_eval_type, expression_node, func_islp_return_type, model::walker >::cleanup (const node_data_type & __data) [inline, virtual, inherited]

Definition at line 432 of file evaluator.h.

#### 6.56.3.5 int cached_forward_evaluator_base< func_islp_eval_type, expression_node, func_islp_return_type, model::walker >::collect (const node_data_type & __data, const return_value & __rval) [inline, virtual, inherited]

Reimplemented from _evaluator_base< func_islp_eval_type, expression_node, func_islp_return_type, model::walker >.

Definition at line 419 of file evaluator.h.

#### 6.56.3.6 virtual int cached_forward_evaluator_base< func_islp_eval_type, expression_node, func_islp_return_type, model::walker >::initialize (const node_data_type & __data) [inline, virtual, inherited]

Definition at line 429 of file evaluator.h.

#### 6.56.3.7 int func_islp_eval::initialize (const expression_node & __data) [inline]

Definition at line 182 of file islp_evaluator.h.

#### 6.56.3.8 void func_islp_eval::initialize () [inline, virtual]

Reimplemented from cached_forward_evaluator_base< func_islp_eval_type, expression_node, func_islp_return_type, model::walker >.

Definition at line 180 of file islp_evaluator.h.

#### 6.56.3.9 bool func_islp_eval::is_cached (const node_data_type & __data) [inline, protected, virtual]

Reimplemented from cached_forward_evaluator_base< func_islp_eval_type, expression_node, func_islp_return_type, model::walker >.

Definition at line 126 of file islp_evaluator.h.

#### 6.56.3.10 void func_islp_eval::new_point (const std::vector< double > & __x, const variable_indicator & __v) [inline]

Definition at line 167 of file islp_evaluator.h.

#### 6.56.3.11 void func_islp_eval::new_range (const std::vector< interval > & __rg, const variable_indicator & __v) [inline]

Definition at line 173 of file islp_evaluator.h.

#### 6.56.3.12 void cached_forward_evaluator_base< func_islp_eval_type, expression_node, func_islp_return_type, model::walker >::postorder (const node_data_type & __data) [inline, virtual, inherited]

Reimplemented from _evaluator_base< func_islp_eval_type, expression_node, func_islp_return_type, model::walker >.

Definition at line 417 of file evaluator.h.

#### 6.56.3.13 int cached_forward_evaluator_base< func_islp_eval_type, expression_node, func_islp_return_type, model::walker >::preorder (const node_data_type & __data) [inline, virtual, inherited]

Reimplemented from cached_evaluator_base< func_islp_eval_type, expression_node, func_islp_return_type, model::walker >.

Definition at line 408 of file evaluator.h.

#### 6.56.3.14 virtual void cached_forward_evaluator_base< func_islp_eval_type, expression_node, func_islp_return_type, model::walker >::retrieve_from_cache (const node_data_type & __data) [inline, virtual, inherited]

Definition at line 431 of file evaluator.h.

#### 6.56.3.15 void func_islp_eval::retrieve_from_cache (const expression_node & __data) [inline]

Definition at line 261 of file islp_evaluator.h.

#### 6.56.3.16 virtual walker cached_evaluator_base< func_islp_eval_type, expression_node, func_islp_return_type, model::walker >::short_cut_to (const node_data_type & __data) [inline, virtual, inherited]

Definition at line 303 of file evaluator.h.

#### 6.56.3.17 model::walker func_islp_eval::short_cut_to (const expression_node & __data) [inline]

Definition at line 164 of file islp_evaluator.h.

#### 6.56.3.18 virtual int cached_forward_evaluator_base< func_islp_eval_type, expression_node, func_islp_return_type, model::walker >::update (const return_value & __rval) [inline, virtual, inherited]

Definition at line 435 of file evaluator.h.

#### 6.56.3.19 virtual int cached_forward_evaluator_base< func_islp_eval_type, expression_node, func_islp_return_type, model::walker >::update (const node_data_type & __data, const return_value & __rval) [inline, virtual, inherited]

Definition at line 433 of file evaluator.h.

#### 6.56.3.20 int func_islp_eval::update (const expression_node & __data, const func_islp_return_type & __rval) [inline]

Definition at line 290 of file islp_evaluator.h.

#### 6.56.3.21 int func_islp_eval::update (const func_islp_return_type & *__rval*) [inline]

Definition at line 284 of file islp_evaluator.h.

#### 6.56.3.22 return_value cached_forward_evaluator_base< func_islp_eval_type, expression_node, func_islp_return_type, model::walker >::value () [inline, virtual, inherited]

Reimplemented from _evaluator_base< func_islp_eval_type, expression_node, func_islp_return_type, model::walker >.

Definition at line 423 of file evaluator.h.

#### 6.56.3.23 int cached_forward_evaluator_base< func_islp_eval_type, expression_node, func_islp_return_type, model::walker >::vcollect (const return_value & *__rval*) [inline, virtual, inherited]

Reimplemented from _evaluator_base< func_islp_eval_type, expression_node, func_islp_return_type, model::walker >.

Definition at line 421 of file evaluator.h.

#### 6.56.3.24 void cached_forward_evaluator_base< func_islp_eval_type, expression_node, func_islp_return_type, model::walker >::vinit () [inline, inherited]

Definition at line 425 of file evaluator.h.

#### 6.56.3.25 return_value cached_forward_evaluator_base< func_islp_eval_type, expression_node, func_islp_return_type, model::walker >::vvalue () [inline, virtual, inherited]

Reimplemented from _evaluator_base< func_islp_eval_type, expression_node, func_islp_return_type, model::walker >.

Definition at line 424 of file evaluator.h.

### 6.56.4 Member Data Documentation

#### 6.56.4.1 func_islp_eval_type _evaluator_base< func_islp_eval_type, expression_node, func_islp_return_type, model::walker >::eval_data [protected, inherited]

Definition at line 252 of file evaluator.h.

#### 6.56.4.2 const variable_indicator* cached_evaluator_base< func_islp_eval_type, expression_node, func_islp_return_type, model::walker >::v_ind [protected, inherited]

Definition at line 295 of file evaluator.h.

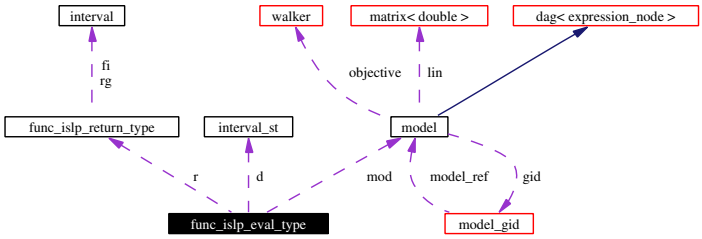
The documentation for this class was generated from the following file:

- islp_evaluator.h

## 6.57 func_islp_eval_type Struct Reference

`#include <islp_evaluator.h>`

Collaboration diagram for func_islp_eval_type:




### Public Attributes

- const std::vector< double > * z
- const std::vector< interval > * range
- std::vector< double > * f
- std::vector< std::vector< interval > > * islp_data
- const model * mod
- union {
  void * p
  interval_st d
  unsigned int info
  } u
- func_islp_return_type r
- unsigned int n

## 6.57.1 Member Data Documentation

### 6.57.1.1 interval_st func_islp_eval_type::d

Definition at line 112 of file islp_evaluator.h.

### 6.57.1.2 std::vector<double>* func_islp_eval_type::f

Definition at line 109 of file islp_evaluator.h.

### 6.57.1.3 unsigned int func_islp_eval_type::info

Definition at line 112 of file islp_evaluator.h.

### 6.57.1.4 std::vector<std::vector<interval> >* func_islp_eval_type::islp_data

Definition at line 110 of file islp_evaluator.h.

### 6.57.1.5 const model* func_islp_eval_type::mod

Definition at line 111 of file islp_evaluator.h.

### 6.57.1.6 unsigned int func_islp_eval_type::n

Definition at line 114 of file islp_evaluator.h.

#### 6.57.1.7 void* func_islp_eval_type::p

Definition at line 112 of file islp_evaluator.h.

#### 6.57.1.8 func_islp_return_type func_islp_eval_type::r

Definition at line 113 of file islp_evaluator.h.

#### 6.57.1.9 const std::vector<interval>* func_islp_eval_type::range

Definition at line 108 of file islp_evaluator.h.

#### 6.57.1.10 union { ... } func_islp_eval_type::u

#### 6.57.1.11 const std::vector<double>* func_islp_eval_type::z

Definition at line 107 of file islp_evaluator.h.

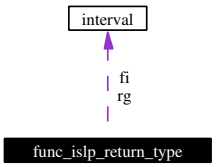
The documentation for this struct was generated from the following file:

- islp_evaluator.h

## 6.58 func_islp_return_type Struct Reference

`#include <islp_evaluator.h>`

Collaboration diagram for func_islp_return_type:




### Public Attributes

- double f
- interval rg
- interval fi

### 6.58.1 Member Data Documentation

#### 6.58.1.1 double func_islp_return_type::f

Definition at line 41 of file islp_evaluator.h.

#### 6.58.1.2 interval func_islp_return_type::fi

Definition at line 43 of file islp_evaluator.h.

#### 6.58.1.3 interval func_islp_return_type::rg

Definition at line 42 of file islp_evaluator.h.

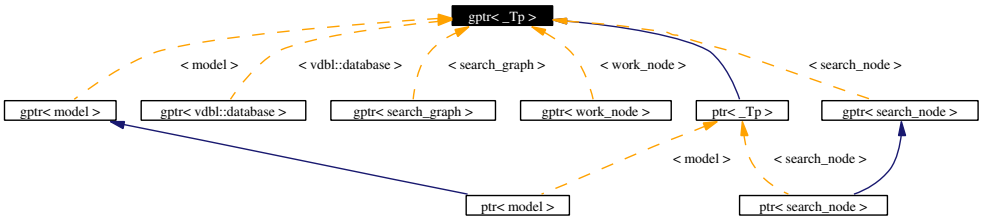
The documentation for this struct was generated from the following file:

- islp_evaluator.h

## 6.59 gptr< _Tp > Class Template Reference

`#include <gptr.h>`

Inheritance diagram for gptr< _Tp >:




**Public Methods**

- virtual ~gptr ()

**template<class _Tp> class gptr< _Tp >**

### 6.59.1 Constructor & Destructor Documentation

#### 6.59.1.1 template<class _Tp> virtual gptr< _Tp >::~gptr () [inline, virtual]

Definition at line 14 of file gptr.h.

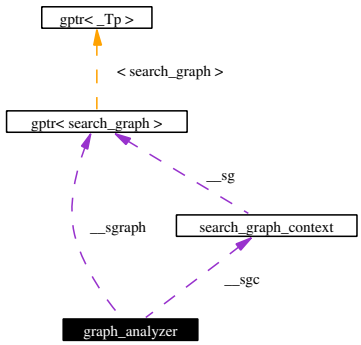
The documentation for this class was generated from the following file:

- gptr.h

## 6.60 graph_analyzer Class Reference

`#include <gr_analyzer.h>`

Collaboration diagram for graph_analyzer:




### Public Methods

- graph_analyzer (const gptr< search_graph > &sgraph, const search_focus &sfoc, const std::string &__n)
- graph_analyzer (const gptr< search_graph > &sgraph, const std::string &__n)
- virtual ~graph_analyzer ()
- virtual bool update_engine (const gptr< work_node > &sgraph, const search_focus &sfoc)
- virtual bool update_engine (const gptr< work_node > &sgraph)
- virtual ga_return_type analyze (const control_data &__c)
- const std::string & get_name () const

### Protected Attributes

- std::string __name
- const gptr< search_graph > * __sgraph
- search_focus * __sfoc
- search_graph_context * __sgc
- vdbl::viewdbase __vdb

#### 6.60.1 Constructor & Destructor Documentation

##### 6.60.1.1 graph_analyzer::graph_analyzer (const gptr< search_graph > & *sgraph*, const search_focus & *sfoc*, const std::string & *__n*) [inline]

Definition at line 58 of file gr_analyzer.h.

##### 6.60.1.2 graph_analyzer::graph_analyzer (const gptr< search_graph > & *sgraph*, const std::string & *__n*) [inline]

Definition at line 67 of file gr_analyzer.h.

##### 6.60.1.3 virtual graph_analyzer::~graph_analyzer () [inline, virtual]

Definition at line 75 of file gr_analyzer.h.

### 6.60.2 Member Function Documentation

#### 6.60.2.1 virtual ga_return_type graph_analyzer::analyze (const control_data & *_c*) [inline, virtual]

Definition at line 92 of file gr_analyzer.h.

#### 6.60.2.2 const std::string& graph_analyzer::get_name () const [inline]

Definition at line 95 of file gr_analyzer.h.

#### 6.60.2.3 virtual bool graph_analyzer::update_engine (const gptr< work_node > & *sgraph*) [inline, virtual]

Definition at line 85 of file gr_analyzer.h.

#### 6.60.2.4 virtual bool graph_analyzer::update_engine (const gptr< work_node > & *sgraph*, const search_focus & *sfoc*) [inline, virtual]

Definition at line 77 of file gr_analyzer.h.

### 6.60.3 Member Data Documentation

#### 6.60.3.1 std::string graph_analyzer::_name [protected]

Definition at line 50 of file gr_analyzer.h.

#### 6.60.3.2 search_focus* graph_analyzer::_sfoc [protected]

Definition at line 52 of file gr_analyzer.h.

#### 6.60.3.3 search_graph_context* graph_analyzer::_sgc [protected]

Definition at line 53 of file gr_analyzer.h.

#### 6.60.3.4 const gptr<search_graph>* graph_analyzer::_sgraph [protected]

Definition at line 51 of file gr_analyzer.h.

#### 6.60.3.5 vdbl::viewdbase graph_analyzer::_vdb [protected]

Definition at line 54 of file gr_analyzer.h.

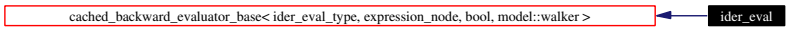
The documentation for this class was generated from the following file:

- gr_analyzer.h

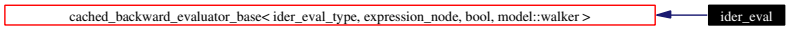
## 6.61 ider_eval Class Reference

`#include <ider_evaluator.h>`

Inheritance diagram for ider_eval:




Collaboration diagram for ider_eval:




### Public Types

- typedef cached_evaluator_base< ider_eval_type, expression_node, bool, model::walker >::node_data_type node_data_type
- typedef cached_evaluator_base< ider_eval_type, expression_node, bool, model::walker >::return_value return_value
- typedef cached_evaluator_base< ider_eval_type, expression_node, bool, model::walker >::walker walker
- typedef ider_eval_type data_type

### Public Methods

- ider_eval (std::vector< interval > __x, std::vector< std::vector< interval > > &__ider_data, variable_indicator &__v, const model &__m, std::vector< std::vector< interval > > *__d, std::vector< interval > &__grad)
- ider_eval (const ider_eval &__d)
- ~ider_eval ()
- void new_box (std::vector< std::vector< interval > > &__ider_data, const variable_indicator &__v)
- void new_result (std::vector< interval > &__grad)
- void set_mult (double scal)
- model::walker short_cut_to (const expression_node &__data)
- void initialize ()
- int calculate (const expression_node &__data)
- void cleanup (const expression_node &__data)
- void retrieve_from_cache (const expression_node &__data)
- int update (const bool &__rval)
- int update (const expression_node &__data, const bool &__rval)
- bool calculate_value (bool eval_all)
- int preorder (const node_data_type &__data)
- void postorder (const node_data_type &__data)
- int collect (const node_data_type &__data, const return_value &__rval)
- int vcollect (const return_value &__rval)
- void vinit ()
- return_value value ()
- return_value vvalue ()
- virtual int calculate (const node_data_type &__data)
- virtual void cleanup (const node_data_type &__data)
- virtual void retrieve_from_cache (const node_data_type &__data)
- virtual int update (const node_data_type &__data, const return_value &__rval)
- virtual int update (const return_value &__rval)
- virtual walker short_cut_to (const node_data_type &__data) PURE_VIRTUAL public

**Protected Methods**

- bool is_cached (const node_data_type &__data)

**Protected Attributes**

- const variable_indicator * v_ind
- ider_eval_type eval_data

### 6.61.1 Member Typedef Documentation

#### 6.61.1.1 typedef ider_eval_type _evaluator_base< ider_eval_type, expression_node, bool, model::walker >::data_type [inherited]

Definition at line 245 of file evaluator.h.

#### 6.61.1.2 typedef cached_evaluator_base<ider_eval_type, expression_node, bool,model::walker>::node_data_type cached_backward_evaluator_base< ider_eval_type, expression_node, bool, model::walker >::node_data_type [inherited]

Reimplemented from cached_evaluator_base< ider_eval_type, expression_node, bool, model::walker >.

Definition at line 444 of file evaluator.h.

#### 6.61.1.3 typedef cached_evaluator_base<ider_eval_type, expression_node, bool,model::walker>::return_value cached_backward_evaluator_base< ider_eval_type, expression_node, bool, model::walker >::return_value [inherited]

Reimplemented from cached_evaluator_base< ider_eval_type, expression_node, bool, model::walker >.

Definition at line 446 of file evaluator.h.

#### 6.61.1.4 typedef cached_evaluator_base<ider_eval_type, expression_node, bool,model::walker>::walker cached_backward_evaluator_base< ider_eval_type, expression_node, bool, model::walker >::walker [inherited]

Reimplemented from cached_evaluator_base< ider_eval_type, expression_node, bool, model::walker >.

Definition at line 448 of file evaluator.h.

### 6.61.2 Constructor & Destructor Documentation

#### 6.61.2.1 ider_eval::ider_eval (std::vector< interval > __x, std::vector< std::vector< interval > > & __ider_data, variable_indicator & __v, const model & __m, std::vector< std::vector< interval > > * __d, std::vector< interval > & __grad) [inline]

Definition at line 1245 of file ider_evaluator.h.

#### 6.61.2.2 ider_eval::ider_eval (const ider_eval & __d) [inline]

Definition at line 1260 of file ider_evaluator.h.

#### 6.61.2.3 ider_eval::~ider_eval () [inline]

Definition at line 1262 of file ider_evaluator.h.

### 6.61.3 Member Function Documentation

#### 6.61.3.1 virtual int cached_backward_evaluator_base< ider_eval_type, expression_node, bool, model::walker >::calculate (const node_data_type & *__data*) [inline, virtual, inherited]

Definition at line 476 of file evaluator.h.

#### 6.61.3.2 int ider_eval::calculate (const expression_node & *__data*) [inline]

Definition at line 1292 of file ider_evaluator.h.

#### 6.61.3.3 bool ider_eval::calculate_value (bool *eval_all*) [inline, virtual]

Reimplemented from cached_backward_evaluator_base< ider_eval_type, expression_node, bool, model::walker >.

Definition at line 1389 of file ider_evaluator.h.

#### 6.61.3.4 virtual void cached_backward_evaluator_base< ider_eval_type, expression_node, bool, model::walker >::cleanup (const node_data_type & *__data*) [inline, virtual, inherited]

Definition at line 477 of file evaluator.h.

#### 6.61.3.5 void ider_eval::cleanup (const expression_node & *__data*) [inline]

Definition at line 1345 of file ider_evaluator.h.

#### 6.61.3.6 int cached_backward_evaluator_base< ider_eval_type, expression_node, bool, model::walker >::collect (const node_data_type & *__data*, const return_value & *__rval*) [inline, virtual, inherited]

Reimplemented from _evaluator_base< ider_eval_type, expression_node, bool, model::walker >.

Definition at line 466 of file evaluator.h.

#### 6.61.3.7 void ider_eval::initialize () [inline, virtual]

Reimplemented from cached_backward_evaluator_base< ider_eval_type, expression_node, bool, model::walker >.

Definition at line 1286 of file ider_evaluator.h.

#### 6.61.3.8 bool ider_eval::is_cached (const node_data_type & *__data*) [inline, protected, virtual]

Reimplemented from cached_backward_evaluator_base< ider_eval_type, expression_node, bool, model::walker >.

Definition at line 1232 of file ider_evaluator.h.

#### 6.61.3.9 void ider_eval::new_box (std::vector< std::vector< interval > > & *__ider_data*, const variable_indicator & *__v*) [inline]

Definition at line 1264 of file ider_evaluator.h.

#### 6.61.3.10 void ider_eval::new_result (std::vector< interval > & *__grad*) [inline]

Definition at line 1271 of file ider_evaluator.h.

#### 6.61.3.11 void cached_backward_evaluator_base< ider_eval_type, expression_node, bool, model::walker >::postorder (const node_data_type & *__data*) [inline, virtual, inherited]

Reimplemented from _evaluator_base< ider_eval_type, expression_node, bool, model::walker >.

Definition at line 465 of file evaluator.h.

#### 6.61.3.12 int cached_backward_evaluator_base< ider_eval_type, expression_node, bool, model::walker >::preorder (const node_data_type & *__data*) [inline, virtual, inherited]

Reimplemented from cached_evaluator_base< ider_eval_type, expression_node, bool, model::walker >.

Definition at line 456 of file evaluator.h.

#### 6.61.3.13 virtual void cached_backward_evaluator_base< ider_eval_type, expression_node, bool, model::walker >::retrieve_from_cache (const node_data_type & *__data*) [inline, virtual, inherited]

Definition at line 478 of file evaluator.h.

#### 6.61.3.14 void ider_eval::retrieve_from_cache (const expression_node & *__data*) [inline]

Definition at line 1356 of file ider_evaluator.h.

#### 6.61.3.15 void ider_eval::set_mult (double *scal*) [inline]

Definition at line 1276 of file ider_evaluator.h.

#### 6.61.3.16 virtual walker cached_evaluator_base< ider_eval_type, expression_node, bool, model::walker >::short_cut_to (const node_data_type & *__data*) [inline, virtual, inherited]

Definition at line 303 of file evaluator.h.

#### 6.61.3.17 model::walker ider_eval::short_cut_to (const expression_node & *__data*) [inline]

Definition at line 1282 of file ider_evaluator.h.

#### 6.61.3.18 virtual int cached_backward_evaluator_base< ider_eval_type, expression_node, bool, model::walker >::update (const return_value & *__rval*) [inline, virtual, inherited]

Definition at line 481 of file evaluator.h.

#### 6.61.3.19 virtual int cached_backward_evaluator_base< ider_eval_type, expression_node, bool, model::walker >::update (const node_data_type & *__data*, const return_value & *__rval*) [inline, virtual, inherited]

Definition at line 479 of file evaluator.h.

#### 6.61.3.20 int ider_eval::update (const expression_node & *__data*, const bool & *__rval*) [inline]

Definition at line 1370 of file ider_evaluator.h.

#### 6.61.3.21 int ider_eval::update (const bool & *__rval*) [inline]

Definition at line 1363 of file ider_evaluator.h.

#### 6.61.3.22 return_value cached_backward_evaluator_base< ider_eval_type, expression_node, bool, model::walker >::value () [inline, virtual, inherited]

Reimplemented from _evaluator_base< ider_eval_type, expression_node, bool, model::walker >.

Definition at line 471 of file evaluator.h.

#### 6.61.3.23 int cached_backward_evaluator_base< ider_eval_type, expression_node, bool, model::walker >::vcollect (const return_value & *__rval*) [inline, virtual, inherited]

Reimplemented from _evaluator_base< ider_eval_type, expression_node, bool, model::walker >.

Definition at line 468 of file evaluator.h.

#### 6.61.3.24 void cached_backward_evaluator_base< ider_eval_type, expression_node, bool, model::walker >::vinit () [inline, inherited]

Definition at line 470 of file evaluator.h.

#### 6.61.3.25 return_value cached_backward_evaluator_base< ider_eval_type, expression_node, bool, model::walker >::vvalue () [inline, virtual, inherited]

Reimplemented from _evaluator_base< ider_eval_type, expression_node, bool, model::walker >.

Definition at line 472 of file evaluator.h.

### 6.61.4 Member Data Documentation

#### 6.61.4.1 ider_eval_type _evaluator_base< ider_eval_type, expression_node, bool, model::walker >::eval_data [protected, inherited]

Definition at line 252 of file evaluator.h.

#### 6.61.4.2 const variable_indicator* cached_evaluator_base< ider_eval_type, expression_node, bool, model::walker >::v_ind [protected, inherited]

Definition at line 295 of file evaluator.h.

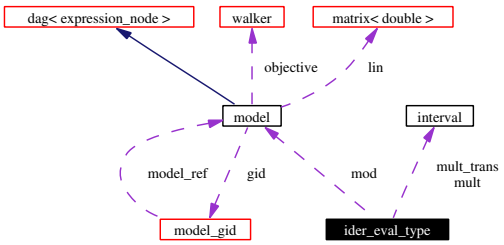
The documentation for this class was generated from the following file:

- ider_evaluator.h

## 6.62 ider_eval_type Struct Reference

`#include <ider_evaluator.h>`

Collaboration diagram for ider_eval_type:




### Public Attributes

- std::vector< std::vector< interval > > * id_data
- std::vector< std::vector< interval > > * d_cache
- std::vector< interval > * grad_vec
- const std::vector< interval > * x
- const model * mod
- interval mult
- interval mult_trans
- unsigned int child_n

### 6.62.1 Member Data Documentation

#### 6.62.1.1 unsigned int ider_eval_type::child_n

Definition at line 1220 of file ider_evaluator.h.

#### 6.62.1.2 std::vector<std::vector<interval> >* ider_eval_type::d_cache

Definition at line 1214 of file ider_evaluator.h.

#### 6.62.1.3 std::vector<interval>* ider_eval_type::grad_vec

Definition at line 1215 of file ider_evaluator.h.

#### 6.62.1.4 std::vector<std::vector<interval> >* ider_eval_type::id_data

Definition at line 1213 of file ider_evaluator.h.

#### 6.62.1.5 const model* ider_eval_type::mod

Definition at line 1217 of file ider_evaluator.h.

#### 6.62.1.6 interval ider_eval_type::mult

Definition at line 1218 of file ider_evaluator.h.

#### 6.62.1.7 interval ider_eval_type::mult_trans

Definition at line 1219 of file ider_evaluator.h.

#### 6.62.1.8 const std::vector<interval>* ider_eval_type::x

Definition at line 1216 of file ider_evaluator.h.

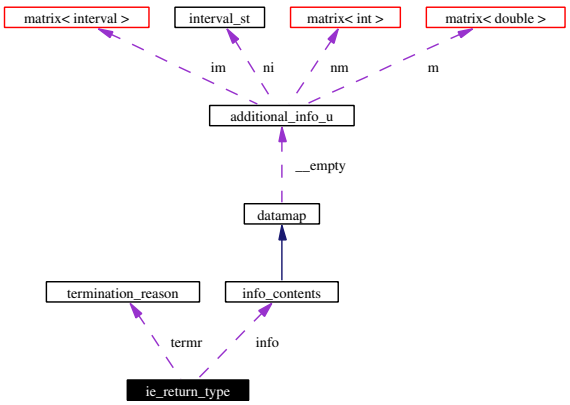
The documentation for this struct was generated from the following file:

- ider_evaluator.h

## 6.63 ie_return_type Class Reference

`#include <ie_rettype.h>`

Collaboration diagram for ie_return_type:




### Public Methods

- ie_return_type ()
- ie_return_type (const termination_reason &__t)
- ie_return_type (const delta_base *__d, double __w)
- ie_return_type (const std::pair< delta_base *, double > &__d)
- ie_return_type (delta __d, double __w)
- ie_return_type (const std::pair< delta, double > &__d)
- ie_return_type (const std::list< delta > &__d, const std::list< double > &__w)
- ie_return_type (const delta_base *__d, double __w, const termination_reason &__t)
- ie_return_type (const std::pair< delta_base *, double > &__d, const termination_reason &__t)
- ie_return_type (delta __d, double __w, const termination_reason &__t)
- ie_return_type (const std::pair< delta, double > &__d, const termination_reason &__t)
- ie_return_type (const std::list< delta > &__d, const std::list< double > &__w, const termination_reason &__t)
- ie_return_type (const ie_return_type &__r)
- virtual ~ie_return_type ()
- void set_termination_reason (const termination_reason &__t)
- const termination_reason & term_reason () const

- bool set_information (const std::string &iname, const additional_info_u &a, bool force=false)
- bool set_information_i (const std::string &iname, int i, const additional_info_u &a, bool force=false)
- const additional_info_u & information (const std::string &iname) const
- const additional_info_u & information (const char *iname) const
- const additional_info_u & information (const std::string &iname, int i) const
- const additional_info_u & information (const char *iname, int i) const
- bool has_information (const std::string &__s) const
- bool has_information (const char *__cp) const
- bool has_information (const std::string &__s, int i) const
- bool has_information (const char *__cp, int i) const
- bool information_indices_set (const std::string &__s, std::vector< int > &__idx) const
- bool information_indices_set (const char *__cp, std::vector< int > &__idx) const
- void unset_information (const std::string &__s)
- void unset_information (const char *__cp)
- void unset_information (const std::string &__s, int i)
- void unset_information (const char *__cp, int i)
- template<class _S> bool retrieve_from_info (const std::string &__s, const std::vector< _S > *&__b) const
- template<class _S> bool retrieve_from_info (const std::string &__s, const matrix< _S > *&__b) const
- template<class _S> bool retrieve_from_info (const std::string &__s, _S &__b) const
- template<class _S> bool retrieve_from_info (const std::string &__s, const std::vector< _S > *&__b, const std::vector< _S > *__def) const
- template<class _S> bool retrieve_from_info (const std::string &__s, const matrix< _S > *&__b, const matrix< _S > *__def) const
- template<class _S> bool retrieve_from_info (const std::string &__s, _S &__b, const _S &__def) const
- template<class _S> bool retrieve_from_info (const char *__s, const std::vector< _S > *&__b) const
- template<class _S> bool retrieve_from_info (const char *__s, const matrix< _S > *&__b) const
- template<class _S> bool retrieve_from_info (const char *__s, _S &__b) const
- template<class _S> bool retrieve_from_info (const char *__s, const std::vector< _S > *&__b, const std::vector< _S > *__def) const
- template<class _S> bool retrieve_from_info (const char *__s, const matrix< _S > *&__b, const matrix< _S > *__def) const
- template<class _S> bool retrieve_from_info (const char *__s, _S &__b, const _S &__def) const
- template<class _S> bool retrieve_from_info_i (const std::string &__s, int i, const std::vector< _S > *&__b) const
- template<class _S> bool retrieve_from_info_i (const std::string &__s, int i, const matrix< _S > *&__b) const
- template<class _S> bool retrieve_from_info_i (const std::string &__s, int i, _S &__b) const
- template<class _S> bool retrieve_from_info_i (const std::string &__s, int i, const std::vector< _S > *&__b, const std::vector< _S > *__def) const
- template<class _S> bool retrieve_from_info_i (const std::string &__s, int i, const matrix< _S > *&__b, const matrix< _S > *__def) const
- template<class _S> bool retrieve_from_info_i (const std::string &__s, int i, _S &__b, const _S &__def) const
- template<class _S> bool retrieve_from_info_i (const char *__s, int i, const std::vector< _S > *&__b) const
- template<class _S> bool retrieve_from_info_i (const char *__s, int i, const matrix< _S > *&__b) const
- template<class _S> bool retrieve_from_info_i (const char *__s, int i, _S &__b) const

- template<class _S> bool retrieve_from_info_i (const char *__s, int i, const std::vector< _S > *&__b, const std::vector< _S > *__def) const
- template<class _S> bool retrieve_from_info_i (const char *__s, int i, const matrix< _S > *&__b, const matrix< _S > *__def) const
- template<class _S> bool retrieve_from_info_i (const char *__s, int i, _S &__b, const _S &__def) const
- ie_return_type operator+ (const std::pair< delta_base *, double > &__d)
- ie_return_type & operator+= (const std::pair< delta_base *, double > &__d)
- ie_return_type & operator= (const ie_return_type &__d)
- unsigned int n_deltas () const
- const std::list< delta_id > & get (work_node &wn, double thresh=-INFINITY)

### 6.63.1 Constructor & Destructor Documentation

#### 6.63.1.1 ie_return_type::ie_return_type () [inline]

Definition at line 47 of file ie_rettype.h.

#### 6.63.1.2 ie_return_type::ie_return_type (const termination_reason & __t) [inline]

Definition at line 48 of file ie_rettype.h.

#### 6.63.1.3 ie_return_type::ie_return_type (const delta_base * __d, double __w) [inline]

Definition at line 50 of file ie_rettype.h.

#### 6.63.1.4 ie_return_type::ie_return_type (const std::pair< delta_base *, double > & __d) [inline]

Definition at line 53 of file ie_rettype.h.

#### 6.63.1.5 ie_return_type::ie_return_type (delta __d, double __w) [inline]

Definition at line 56 of file ie_rettype.h.

#### 6.63.1.6 ie_return_type::ie_return_type (const std::pair< delta, double > & __d) [inline]

Definition at line 58 of file ie_rettype.h.

#### 6.63.1.7 ie_return_type::ie_return_type (const std::list< delta > & __d, const std::list< double > & __w) [inline]

Definition at line 60 of file ie_rettype.h.

#### 6.63.1.8 ie_return_type::ie_return_type (const delta_base * __d, double __w, const termination_reason & __t) [inline]

Definition at line 66 of file ie_rettype.h.

#### 6.63.1.9 ie_return_type::ie_return_type (const std::pair< delta_base *, double > & __d, const termination_reason & __t) [inline]

Definition at line 70 of file ie_rettype.h.

#### 6.63.1.10 ie_return_type::ie_return_type (delta __d, double __w, const termination_reason & __t) [inline]

Definition at line 74 of file ie_rettype.h.

#### 6.63.1.11 ie_return_type::ie_return_type (const std::pair< delta, double > & __d, const termination_reason & __t) [inline]

Definition at line 77 of file ie_rettype.h.

#### 6.63.1.12 ie_return_type::ie_return_type (const std::list< delta > & __d, const std::list< double > & __w, const termination_reason & __t) [inline]

Definition at line 80 of file ie_rettype.h.

#### 6.63.1.13 ie_return_type::ie_return_type (const ie_return_type & __r) [inline]

Definition at line 87 of file ie_rettype.h.

#### 6.63.1.14 virtual ie_return_type::~ie_return_type () [inline, virtual]

Definition at line 91 of file ie_rettype.h.

### 6.63.2 Member Function Documentation

#### 6.63.2.1 const std::list< delta_id > & ie_return_type::get (work_node & wn, double thresh = -INFINITY)

Definition at line 30 of file ie_rettype.cc.

#### 6.63.2.2 bool ie_return_type::has_information (const char * __cp, int i) const [inline]

Definition at line 133 of file ieret-inline.h.

#### 6.63.2.3 bool ie_return_type::has_information (const std::string & __s, int i) const [inline]

Definition at line 128 of file ieret-inline.h.

#### 6.63.2.4 bool ie_return_type::has_information (const char * __cp) const [inline]

Definition at line 123 of file ieret-inline.h.

#### 6.63.2.5 bool ie_return_type::has_information (const std::string & __s) const [inline]

Definition at line 118 of file ieret-inline.h.

#### 6.63.2.6 const additional_info_u & ie_return_type::information (const char * iname, int i) const [inline]

Definition at line 112 of file ieret-inline.h.

#### 6.63.2.7 const additional_info_u & ie_return_type::information (const std::string & *iname*, int *i*) const [inline]

Definition at line 106 of file ieret-inline.h.

#### 6.63.2.8 const additional_info_u & ie_return_type::information (const char * *iname*) const [inline]

Definition at line 100 of file ieret-inline.h.

#### 6.63.2.9 const additional_info_u & ie_return_type::information (const std::string & *iname*) const [inline]

Definition at line 94 of file ieret-inline.h.

#### 6.63.2.10 bool ie_return_type::information_indices_set (const char * *__cp*, std::vector< int > & *__idx*) const [inline]

Definition at line 144 of file ieret-inline.h.

#### 6.63.2.11 bool ie_return_type::information_indices_set (const std::string & *__s*, std::vector< int > & *__idx*) const [inline]

Definition at line 138 of file ieret-inline.h.

#### 6.63.2.12 unsigned int ie_return_type::n_deltas () const [inline]

Definition at line 221 of file ie_rettype.h.

#### 6.63.2.13 ie_return_type ie_return_type::operator+ (const std::pair< delta_base *, double > & *__d*) [inline]

Definition at line 40 of file ieret-inline.h.

#### 6.63.2.14 ie_return_type & ie_return_type::operator+= (const std::pair< delta_base *, double > & *__d*) [inline]

Definition at line 48 of file ieret-inline.h.

#### 6.63.2.15 ie_return_type & ie_return_type::operator= (const ie_return_type & *__d*) [inline]

Definition at line 72 of file ieret-inline.h.

#### 6.63.2.16 template<class _S> bool ie_return_type::retrieve_from_info (const char * *__s*, _S & *__b*, const _S & *__def*) const [inline]

Definition at line 247 of file ieret-inline.h.

#### 6.63.2.17 template<class _S> bool ie_return_type::retrieve_from_info (const char * *__s*, const matrix< _S > *& *__b*, const matrix< _S > * *__def*) const [inline]

Definition at line 240 of file ieret-inline.h.

**6.63.2.18 template<class _S> bool ie_return_type::retrieve_from_info (const char * __s, const std::vector< _S > *& __b, const std::vector< _S > * __def) const** `[inline]`

Definition at line 233 of file ieret-inline.h.

**6.63.2.19 template<class _S> bool ie_return_type::retrieve_from_info (const char * __s, _S & __b) const** `[inline]`

Definition at line 227 of file ieret-inline.h.

**6.63.2.20 template<class _S> bool ie_return_type::retrieve_from_info (const char * __s, const matrix< _S > *& __b) const** `[inline]`

Definition at line 220 of file ieret-inline.h.

**6.63.2.21 template<class _S> bool ie_return_type::retrieve_from_info (const char * __s, const std::vector< _S > *& __b) const** `[inline]`

Definition at line 213 of file ieret-inline.h.

**6.63.2.22 template<class _S> bool ie_return_type::retrieve_from_info (const std::string & __s, _S & __b, const _S & __def) const** `[inline]`

Definition at line 206 of file ieret-inline.h.

**6.63.2.23 template<class _S> bool ie_return_type::retrieve_from_info (const std::string & __s, const matrix< _S > *& __b, const matrix< _S > * __def) const** `[inline]`

Definition at line 199 of file ieret-inline.h.

**6.63.2.24 template<class _S> bool ie_return_type::retrieve_from_info (const std::string & __s, const std::vector< _S > *& __b, const std::vector< _S > * __def) const** `[inline]`

Definition at line 192 of file ieret-inline.h.

**6.63.2.25 template<class _S> bool ie_return_type::retrieve_from_info (const std::string & __s, _S & __b) const** `[inline]`

Definition at line 185 of file ieret-inline.h.

**6.63.2.26 template<class _S> bool ie_return_type::retrieve_from_info (const std::string & __s, const matrix< _S > *& __b) const** `[inline]`

Definition at line 178 of file ieret-inline.h.

**6.63.2.27 template<class _S> bool ie_return_type::retrieve_from_info (const std::string & __s, const std::vector< _S > *& __b) const** `[inline]`

Definition at line 171 of file ieret-inline.h.

**6.63.2.28 template<class _S> bool ie_return_type::retrieve_from_info_i (const char * __s, int i, _S & __b, const _S & __def) const** `[inline]`

Definition at line 331 of file ieret-inline.h.

**6.63.2.29 template<class _S> bool ie_return_type::retrieve_from_info_i (const char * __s, int i, const matrix< _S > *& __b, const matrix< _S > * __def) const** `[inline]`

Definition at line 324 of file ieret-inline.h.

**6.63.2.30 template<class _S> bool ie_return_type::retrieve_from_info_i (const char * __s, int i, const std::vector< _S > *& __b, const std::vector< _S > * __def) const** `[inline]`

Definition at line 317 of file ieret-inline.h.

**6.63.2.31 template<class _S> bool ie_return_type::retrieve_from_info_i (const char * __s, int i, _S & __b) const** `[inline]`

Definition at line 310 of file ieret-inline.h.

**6.63.2.32 template<class _S> bool ie_return_type::retrieve_from_info_i (const char * __s, int i, const matrix< _S > *& __b) const** `[inline]`

Definition at line 303 of file ieret-inline.h.

**6.63.2.33 template<class _S> bool ie_return_type::retrieve_from_info_i (const char * __s, int i, const std::vector< _S > *& __b) const** `[inline]`

Definition at line 296 of file ieret-inline.h.

**6.63.2.34 template<class _S> bool ie_return_type::retrieve_from_info_i (const std::string & __s, int i, _S & __b, const _S & __def) const** `[inline]`

Definition at line 289 of file ieret-inline.h.

**6.63.2.35 template<class _S> bool ie_return_type::retrieve_from_info_i (const std::string & __s, int i, const matrix< _S > *& __b, const matrix< _S > * __def) const** `[inline]`

Definition at line 282 of file ieret-inline.h.

**6.63.2.36 template<class _S> bool ie_return_type::retrieve_from_info_i (const std::string & __s, int i, const std::vector< _S > *& __b, const std::vector< _S > * __def) const** `[inline]`

Definition at line 275 of file ieret-inline.h.

**6.63.2.37 template<class _S> bool ie_return_type::retrieve_from_info_i (const std::string & __s, int i, _S & __b) const** `[inline]`

Definition at line 268 of file ieret-inline.h.

#### 6.63.2.38 template<class _S> bool ie_return_type::retrieve_from_info_i (const std::string & __s, int i, const matrix< _S > *& __b) const [inline]

Definition at line 261 of file ieret-inline.h.

#### 6.63.2.39 template<class _S> bool ie_return_type::retrieve_from_info_i (const std::string & __s, int i, const std::vector< _S > *& __b) const [inline]

Definition at line 254 of file ieret-inline.h.

#### 6.63.2.40 bool ie_return_type::set_information (const std::string & *iname*, const additional_info_u & *a*, bool *force* = false) [inline]

Definition at line 82 of file ieret-inline.h.

#### 6.63.2.41 bool ie_return_type::set_information_i (const std::string & *iname*, int *i*, const additional_info_u & *a*, bool *force* = false) [inline]

Definition at line 88 of file ieret-inline.h.

#### 6.63.2.42 void ie_return_type::set_termination_reason (const termination_reason & __t) [inline]

Definition at line 30 of file ieret-inline.h.

#### 6.63.2.43 const termination_reason & ie_return_type::term_reason () [inline]

Definition at line 35 of file ieret-inline.h.

#### 6.63.2.44 void ie_return_type::unset_information (const char * __cp, int i) [inline]

Definition at line 165 of file ieret-inline.h.

#### 6.63.2.45 void ie_return_type::unset_information (const std::string & __s, int i) [inline]

Definition at line 160 of file ieret-inline.h.

#### 6.63.2.46 void ie_return_type::unset_information (const char * __cp) [inline]

Definition at line 155 of file ieret-inline.h.

#### 6.63.2.47 void ie_return_type::unset_information (const std::string & __s) [inline]

Definition at line 150 of file ieret-inline.h.

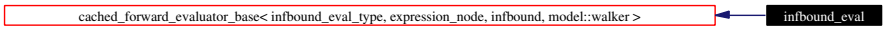
The documentation for this class was generated from the following files:

- ie_rettype.h
- ie_rettype.cc
- ieret-inline.h

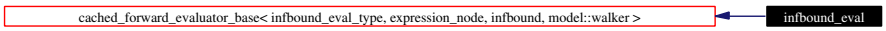
## 6.64 infbound_eval Class Reference

`#include <infb_evaluator.h>`

Inheritance diagram for infbound_eval:




Collaboration diagram for infbound_eval:




### Public Types

- typedef cached_evaluator_base< infbound_eval_type, expression_node, infbound, model::walker >::node_data_type node_data_type
- typedef cached_evaluator_base< infbound_eval_type, expression_node, infbound, model::walker >::return_value return_value
- typedef cached_evaluator_base< infbound_eval_type, expression_node, infbound, model::walker >::walker walker
- typedef infbound_eval_type data_type

### Public Methods

- infbound_eval (const std::vector< infbound > &__x, const variable_indicator &__v, const model &__m, std::vector< infbound > *__c)
- infbound_eval (const infbound_eval &__v)
- ~infbound_eval ()
- model::walker short_cut_to (const expression_node &__data)
- void initialize ()
- int initialize (const expression_node &__data)
- void calculate (const expression_node &__data)
- void retrieve_from_cache (const expression_node &__data)
- int update (const infbound &__rval)
- int update (const expression_node &__data, const infbound &__rval)
- infbound calculate_value (bool eval_all)
- int preorder (const node_data_type &__data)
- void postorder (const node_data_type &__data)
- int collect (const node_data_type &__data, const return_value &__rval)
- int vcollect (const return_value &__rval)
- return_value value ()
- return_value vvalue ()
- void vinit ()
- virtual int initialize (const node_data_type &__data)
- virtual void calculate (const node_data_type &__data)
- virtual void retrieve_from_cache (const node_data_type &__data)
- virtual void cleanup (const node_data_type &__data)
- virtual int update (const node_data_type &__data, const return_value &__rval)
- virtual int update (const return_value &__rval)
- virtual walker short_cut_to (const node_data_type &__data) PURE_VIRTUAL public

**Protected Methods**

- bool is_cached (const node_data_type &__data)

**Protected Attributes**

- const variable_indicator * v_ind
- infbound_eval_type eval_data

### 6.64.1 Member Typedef Documentation

#### 6.64.1.1 typedef infbound_eval_type _evaluator_base< infbound_eval_type, expression_node, infbound, model::walker >::data_type [inherited]

Definition at line 245 of file evaluator.h.

#### 6.64.1.2 typedef cached_evaluator_base<infbound_eval_type, expression_node, infbound,model::walker>::node_data_type cached_forward_evaluator_base< infbound_eval_type, expression_node, infbound, model::walker >::node_data_type [inherited]

Reimplemented from cached_evaluator_base< infbound_eval_type, expression_node, infbound, model::walker >.

Definition at line 396 of file evaluator.h.

#### 6.64.1.3 typedef cached_evaluator_base<infbound_eval_type, expression_node, infbound,model::walker>::return_value cached_forward_evaluator_base< infbound_eval_type, expression_node, infbound, model::walker >::return_value [inherited]

Reimplemented from cached_evaluator_base< infbound_eval_type, expression_node, infbound, model::walker >.

Definition at line 398 of file evaluator.h.

#### 6.64.1.4 typedef cached_evaluator_base<infbound_eval_type, expression_node, infbound,model::walker>::walker cached_forward_evaluator_base< infbound_eval_type, expression_node, infbound, model::walker >::walker [inherited]

Reimplemented from cached_evaluator_base< infbound_eval_type, expression_node, infbound, model::walker >.

Definition at line 400 of file evaluator.h.

### 6.64.2 Constructor & Destructor Documentation

#### 6.64.2.1 infbound_eval::infbound_eval (const std::vector< infbound > & *__x*, const variable_indicator & *__v*, const model & *__m*, std::vector< infbound > * *__c*) [inline]

Definition at line 111 of file infb_evaluator.h.

#### 6.64.2.2 infbound_eval::infbound_eval (const infbound_eval & *__v*) [inline]

Definition at line 123 of file infb_evaluator.h.

#### 6.64.2.3 infbound_eval::~infbound_eval () [inline]

Definition at line 125 of file infb_evaluator.h.

### 6.64.3 Member Function Documentation

#### 6.64.3.1 virtual void cached_forward_evaluator_base< infbound_eval_type, expression_node, infbound, model::walker >::calculate (const node_data_type & *__data*) [inline, virtual, inherited]

Definition at line 430 of file evaluator.h.

#### 6.64.3.2 void infbound_eval::calculate (const expression_node & *__data*) [inline]

Definition at line 188 of file infb_evaluator.h.

#### 6.64.3.3 infbound infbound_eval::calculate_value (bool *eval_all*) [inline, virtual]

Reimplemented from cached_forward_evaluator_base< infbound_eval_type, expression_node, infbound, model::walker >.

Definition at line 736 of file infb_evaluator.h.

#### 6.64.3.4 virtual void cached_forward_evaluator_base< infbound_eval_type, expression_node, infbound, model::walker >::cleanup (const node_data_type & *__data*) [inline, virtual, inherited]

Definition at line 432 of file evaluator.h.

#### 6.64.3.5 int cached_forward_evaluator_base< infbound_eval_type, expression_node, infbound, model::walker >::collect (const node_data_type & *__data*, const return_value & *__rval*) [inline, virtual, inherited]

Reimplemented from _evaluator_base< infbound_eval_type, expression_node, infbound, model::walker >.

Definition at line 419 of file evaluator.h.

#### 6.64.3.6 virtual int cached_forward_evaluator_base< infbound_eval_type, expression_node, infbound, model::walker >::initialize (const node_data_type & *__data*) [inline, virtual, inherited]

Definition at line 429 of file evaluator.h.

#### 6.64.3.7 int infbound_eval::initialize (const expression_node & *__data*) [inline]

Definition at line 132 of file infb_evaluator.h.

#### 6.64.3.8 void infbound_eval::initialize () [inline, virtual]

Reimplemented from cached_forward_evaluator_base< infbound_eval_type, expression_node, infbound, model::walker >.

Definition at line 130 of file infb_evaluator.h.

#### 6.64.3.9 bool infbound_eval::is_cached (const node_data_type & *__data*) [inline, protected, virtual]

Reimplemented from cached_forward_evaluator_base< infbound_eval_type, expression_node, infbound, model::walker >.

Definition at line 65 of file infb_evaluator.h.

#### 6.64.3.10 void cached_forward_evaluator_base< infbound_eval_type, expression_node, infbound, model::walker >::postorder (const node_data_type & *__data*) [inline, virtual, inherited]

Reimplemented from _evaluator_base< infbound_eval_type, expression_node, infbound, model::walker >.

Definition at line 417 of file evaluator.h.

#### 6.64.3.11 int cached_forward_evaluator_base< infbound_eval_type, expression_node, infbound, model::walker >::preorder (const node_data_type & *__data*) [inline, virtual, inherited]

Reimplemented from cached_evaluator_base< infbound_eval_type, expression_node, infbound, model::walker >.

Definition at line 408 of file evaluator.h.

#### 6.64.3.12 virtual void cached_forward_evaluator_base< infbound_eval_type, expression_node, infbound, model::walker >::retrieve_from_cache (const node_data_type & *__data*) [inline, virtual, inherited]

Definition at line 431 of file evaluator.h.

#### 6.64.3.13 void infbound_eval::retrieve_from_cache (const expression_node & *__data*) [inline]

Definition at line 200 of file infb_evaluator.h.

#### 6.64.3.14 virtual walker cached_evaluator_base< infbound_eval_type, expression_node, infbound, model::walker >::short_cut_to (const node_data_type & *__data*) [inline, virtual, inherited]

Definition at line 303 of file evaluator.h.

#### 6.64.3.15 model::walker infbound_eval::short_cut_to (const expression_node & *__data*) [inline]

Definition at line 127 of file infb_evaluator.h.

#### 6.64.3.16 virtual int cached_forward_evaluator_base< infbound_eval_type, expression_node, infbound, model::walker >::update (const return_value & *__rval*) [inline, virtual, inherited]

Definition at line 435 of file evaluator.h.

#### 6.64.3.17 virtual int cached_forward_evaluator_base< infbound_eval_type, expression_node, infbound, model::walker >::update (const node_data_type & *__data*, const return_value & *__rval*) [inline, virtual, inherited]

Definition at line 433 of file evaluator.h.

#### 6.64.3.18 int infbound_eval::update (const expression_node & *__data*, const infbound & *__rval*) [inline]

Definition at line 225 of file infb_evaluator.h.

#### 6.64.3.19 int infbound_eval::update (const infbound & *__rval*) [inline]

Definition at line 219 of file infb_evaluator.h.

#### 6.64.3.20 return_value cached_forward_evaluator_base< infbound_eval_type, expression_node, infbound, model::walker >::value () [inline, virtual, inherited]

Reimplemented from _evaluator_base< infbound_eval_type, expression_node, infbound, model::walker >.

Definition at line 423 of file evaluator.h.

#### 6.64.3.21 int cached_forward_evaluator_base< infbound_eval_type, expression_node, infbound, model::walker >::vcollect (const return_value & *__rval*) [inline, virtual, inherited]

Reimplemented from _evaluator_base< infbound_eval_type, expression_node, infbound, model::walker >.

Definition at line 421 of file evaluator.h.

#### 6.64.3.22 void cached_forward_evaluator_base< infbound_eval_type, expression_node, infbound, model::walker >::vinit () [inline, inherited]

Definition at line 425 of file evaluator.h.

#### 6.64.3.23 return_value cached_forward_evaluator_base< infbound_eval_type, expression_node, infbound, model::walker >::vvalue () [inline, virtual, inherited]

Reimplemented from _evaluator_base< infbound_eval_type, expression_node, infbound, model::walker >.

Definition at line 424 of file evaluator.h.

### 6.64.4 Member Data Documentation

#### 6.64.4.1 infbound_eval_type _evaluator_base< infbound_eval_type, expression_node, infbound, model::walker >::eval_data [protected, inherited]

Definition at line 252 of file evaluator.h.

#### 6.64.4.2 const variable_indicator* cached_evaluator_base< infbound_eval_type, expression_node, infbound, model::walker >::v_ind [protected, inherited]

Definition at line 295 of file evaluator.h.

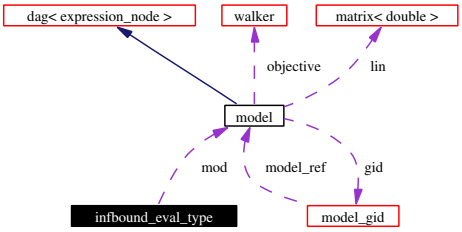
The documentation for this class was generated from the following file:

- infb_evaluator.h

## 6.65 infbound_eval_type Struct Reference

`#include <infb_evaluator.h>`

Collaboration diagram for infbound_eval_type:




### Public Attributes

- const std::vector< infbound > * x
- std::vector< infbound > * cache
- const model * mod
- void * p
- infbound d
- int info
- infbound r
- unsigned int n

### 6.65.1 Member Data Documentation

#### 6.65.1.1 std::vector<infbound>* infbound_eval_type::cache

Definition at line 47 of file infb_evaluator.h.

#### 6.65.1.2 infbound infbound_eval_type::d

Definition at line 50 of file infb_evaluator.h.

#### 6.65.1.3 int infbound_eval_type::info

Definition at line 51 of file infb_evaluator.h.

#### 6.65.1.4 const model* infbound_eval_type::mod

Definition at line 48 of file infb_evaluator.h.

#### 6.65.1.5 unsigned int infbound_eval_type::n

Definition at line 53 of file infb_evaluator.h.

#### 6.65.1.6 void* infbound_eval_type::p

Definition at line 49 of file infb_evaluator.h.

#### 6.65.1.7 infbound infbound_eval_type::r

Definition at line 52 of file infb_evaluator.h.

#### 6.65.1.8 const std::vector<infbound>* infbound_eval_type::x

Definition at line 46 of file infb_evaluator.h.

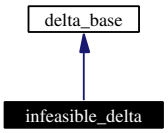
The documentation for this struct was generated from the following file:

- infb_evaluator.h

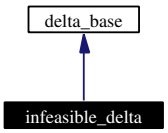
## 6.66 infeasible_delta Class Reference

`#include <infeasible_delta.h>`

Inheritance diagram for infeasible_delta:




Collaboration diagram for infeasible_delta:




### Public Methods

- infeasible_delta ()
- infeasible_delta (const infeasible_delta &__d)
- ~infeasible_delta ()
- infeasible_delta * new_copy () const
- void destroy_copy (infeasible_delta *__d)
- bool apply (work_node &_x, undelta_base *&_u) const
- virtual void destroy_copy (delta_base *__d)
- delta make_delta (const std::string &a)
- const std::string & get_action () const
- virtual void convert (work_node &_x, delta_base *&_d)
- virtual bool apply3 (work_node &_x, const work_node &_y, undelta_base *&_u) const

### Protected Attributes

- std::string _action

### 6.66.1 Constructor & Destructor Documentation

#### 6.66.1.1 infeasible_delta::infeasible_delta () [inline]

Definition at line 62 of file infeasible_delta.h.

#### 6.66.1.2 infeasible_delta::infeasible_delta (const infeasible_delta & *__d*) [inline]

Definition at line 63 of file infeasible_delta.h.

#### 6.66.1.3 infeasible_delta::~infeasible_delta () [inline]

Definition at line 70 of file infeasible_delta.h.

### 6.66.2 Member Function Documentation

#### 6.66.2.1 bool infeasible_delta::apply (work_node & *_x*, undelta_base *& *_u*) const [inline, virtual]

Reimplemented from delta_base.

Definition at line 75 of file infeasible_delta.h.

#### 6.66.2.2 bool delta_base::apply3 (work_node & *_x*, const work_node & *_y*, undelta_base *& *_u*) const [inline, virtual, inherited]

Definition at line 63 of file api_delta.h.

#### 6.66.2.3 virtual void delta_base::convert (work_node & *_x*, delta_base *& *_d*) [inline, virtual, inherited]

Reimplemented in table_delta.

Definition at line 107 of file api_deltabase.h.

#### 6.66.2.4 virtual void delta_base::destroy_copy (delta_base * *__d*) [inline, virtual, inherited]

Definition at line 95 of file api_deltabase.h.

#### 6.66.2.5 void infeasible_delta::destroy_copy (infeasible_delta * *__d*) [inline]

Definition at line 73 of file infeasible_delta.h.

#### 6.66.2.6 const std::string& delta_base::get_action () const [inline, inherited]

Definition at line 105 of file api_deltabase.h.

#### 6.66.2.7 delta delta_base::make_delta (const std::string & *a*) [inline, inherited]

Definition at line 99 of file api_deltabase.h.

#### 6.66.2.8 infeasible_delta* infeasible_delta::new_copy () const [inline, virtual]

Reimplemented from delta_base.

Definition at line 72 of file infeasible_delta.h.

### 6.66.3 Member Data Documentation

#### 6.66.3.1 std::string delta_base::_action [protected, inherited]

Definition at line 86 of file api_deltabase.h.

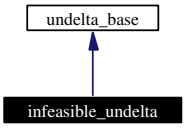
The documentation for this class was generated from the following file:

- infeasible_delta.h

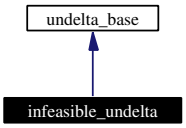
## 6.67 infeasible_undelta Class Reference

`#include <infeasible_delta.h>`

Inheritance diagram for infeasible_undelta:




Collaboration diagram for infeasible_undelta:




### Public Methods

- infeasible_undelta (bool _oi=false)
- infeasible_undelta (const infeasible_undelta &__d)
- ~infeasible_undelta ()
- infeasible_undelta * new_copy () const
- void destroy_copy (infeasible_undelta *__d)
- bool unapply (work_node &_x) const
- virtual void destroy_copy (undelta_base *__d)
- undelta make_undelta ()
- virtual bool unapply3 (work_node &_x, const work_node &_y) const

### 6.67.1 Constructor & Destructor Documentation

#### 6.67.1.1 infeasible_undelta::infeasible_undelta (bool _oi = false) [inline]

Definition at line 38 of file infeasible_delta.h.

#### 6.67.1.2 infeasible_undelta::infeasible_undelta (const infeasible_undelta & __d) [inline]

Definition at line 39 of file infeasible_delta.h.

#### 6.67.1.3 infeasible_undelta::~infeasible_undelta () [inline]

Definition at line 47 of file infeasible_delta.h.

### 6.67.2 Member Function Documentation

#### 6.67.2.1 virtual void undelta_base::destroy_copy (undelta_base * __d) [inline, virtual, inherited]

Definition at line 146 of file api_deltabase.h.

#### 6.67.2.2 void infeasible_undelta::destroy_copy (infeasible_undelta * __d) [inline]

Definition at line 50 of file infeasible_delta.h.

#### 6.67.2.3 undelta undelta_base::make_undelta () [inline, inherited]

Definition at line 150 of file api_deltabase.h.

#### 6.67.2.4 infeasible_undelta* infeasible_undelta::new_copy () const [inline, virtual]

Reimplemented from undelta_base.

Definition at line 49 of file infeasible_delta.h.

#### 6.67.2.5 bool infeasible_undelta::unapply (work_node & _x) const [inline, virtual]

Reimplemented from undelta_base.

Definition at line 52 of file infeasible_delta.h.

#### 6.67.2.6 bool undelta_base::unapply3 (work_node & _x, const work_node & _y) const [inline, virtual, inherited]

Definition at line 69 of file api_delta.h.

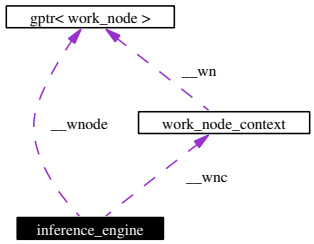
The documentation for this class was generated from the following file:

- infeasible_delta.h

## 6.68 inference_engine Class Reference

`#include <inference_engine.h>`

Collaboration diagram for inference_engine:




### Public Methods

- inference_engine (const gptr< work_node > &wnode, const std::string &__n)
- virtual ~inference_engine ()
- virtual bool update_engine (const gptr< work_node > &wnode)
- std::pair< std::list< delta_id >, std::list< delta_id > > new_deltas ()
- const delta * get_delta (const delta_id &__d) const
- const model * get_model () const
- virtual ie_return_type infer (const control_data &__c)
- const std::string & get_name () const
- virtual statistic_info last_call_stat ()
- virtual statistic_info cumulative_stat ()

### Protected Attributes

- std::string __name
- const gptr< work_node > * __wnode
- work_node_context * __wnc
- vdbl::viewdbase __vdb
- std::vector< delta_id > _old_deltas
- std::vector< delta_id > _new_deltas

## 6.68.1 Constructor & Destructor Documentation

### 6.68.1.1 inference_engine::inference_engine (const gptr< work_node > & *wnode*, const std::string & *__n*) [inline]

Definition at line 58 of file inference_engine.h.

### 6.68.1.2 virtual inference_engine::~inference_engine () [inline, virtual]

Definition at line 69 of file inference_engine.h.

## 6.68.2 Member Function Documentation

### 6.68.2.1 virtual statistic_info inference_engine::cumulative_stat () [inline, virtual]

Definition at line 95 of file inference_engine.h.

#### 6.68.2.2 const delta* inference_engine::get_delta (const delta_id & *__d*) const [inline]

Definition at line 82 of file inference_engine.h.

#### 6.68.2.3 const model* inference_engine::get_model () const [inline]

Definition at line 85 of file inference_engine.h.

#### 6.68.2.4 const std::string& inference_engine::get_name () const [inline]

Definition at line 91 of file inference_engine.h.

#### 6.68.2.5 virtual ie_return_type inference_engine::infer (const control_data & *__c*) [inline, virtual]

Definition at line 88 of file inference_engine.h.

#### 6.68.2.6 virtual statistic_info inference_engine::last_call_stat () [inline, virtual]

Definition at line 94 of file inference_engine.h.

#### 6.68.2.7 std::pair< std::list< delta_id >, std::list< delta_id > > inference_engine::new_deltas ()

Definition at line 41 of file inference_engine.cc.

#### 6.68.2.8 virtual bool inference_engine::update_engine (const gptr< work_node > & *wnode*) [inline, virtual]

Definition at line 71 of file inference_engine.h.

### 6.68.3 Member Data Documentation

#### 6.68.3.1 std::string inference_engine::__name [protected]

Definition at line 47 of file inference_engine.h.

#### 6.68.3.2 vdbl::viewdbase inference_engine::__vdb [protected]

Definition at line 50 of file inference_engine.h.

#### 6.68.3.3 work_node_context* inference_engine::__wnc [protected]

Definition at line 49 of file inference_engine.h.

#### 6.68.3.4 const gptr<work_node>* inference_engine::__wnode [protected]

Definition at line 48 of file inference_engine.h.

#### 6.68.3.5 std::vector<delta_id> inference_engine::_new_deltas [protected]

Definition at line 51 of file inference_engine.h.

#### 6.68.3.6 std::vector<delta_id> inference_engine::_old_deltas [protected]

Definition at line 51 of file inference_engine.h.

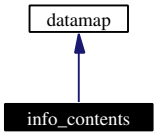
The documentation for this class was generated from the following files:

- inference_engine.h
- inference_engine.cc

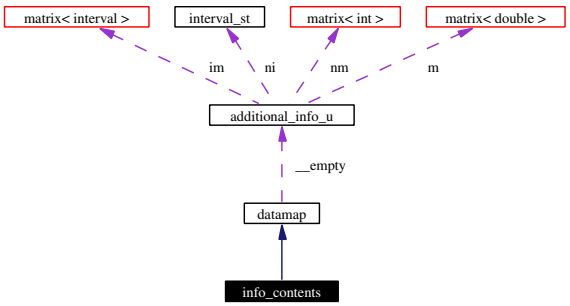
## 6.69 info_contents Class Reference

`#include <info_contents.h>`

Inheritance diagram for info_contents:




Collaboration diagram for info_contents:




### Public Methods

- info_contents ()
- info_contents (const std::string &__n, const additional_info_u &__v)
- info_contents (const char *__n, const additional_info_u &__v)
- info_contents (const info_contents &__c)
- virtual ~info_contents ()
- info_contents & operator= (const info_contents &__c)
- bool sinsert (const std::string &__s, const additional_info_u &__h, bool replace)
- bool sinsert (const char *__cp, const additional_info_u &__h, bool replace)
- bool sinsert (const std::string &__s, int i, const additional_info_u &__h, bool replace)
- bool sinsert (const char *__cp, int i, const additional_info_u &__h, bool replace)
- const additional_info_u & sfind (const std::string &__s) const
- const additional_info_u & sfind (const char *__cp) const
- const additional_info_u & sfind (const std::string &__s, int i) const
- const additional_info_u & sfind (const char *__cp, int i) const

- void remove (const std::string &__s)
- void remove (const char *__cp)
- void remove (const std::string &__s, int i)
- void remove (const char *__cp, int i)
- bool defd (const std::string &__s) const
- bool defd (const char *__cp) const
- bool defd (const std::string &__s, int i) const
- bool defd (const char *__cp, int i) const
- bool which (const std::string &__s, std::vector< int > &__idx) const
- bool which (const char *__cp, std::vector< int > &__idx) const
- bool retrieve (const std::string &__s, bool &__b) const
- bool retrieve (const std::string &__s, bool &__b, bool __def) const
- bool retrieve (const std::string &__s, int &__d) const
- bool retrieve (const std::string &__s, int &__d, int __def) const
- bool retrieve (const std::string &__s, unsigned int &__d) const
- bool retrieve (const std::string &__s, unsigned int &__d, unsigned int __def) const
- bool retrieve (const std::string &__s, double &__d) const
- bool retrieve (const std::string &__s, double &__d, double __def) const
- bool retrieve (const std::string &__s, interval &__b) const
- bool retrieve (const std::string &__s, interval &__b, const interval &__def) const
- bool retrieve (const std::string &__s, std::string &__is) const
- bool retrieve (const std::string &__s, std::string &__is, const std::string &__def) const
- bool retrieve (const std::string &__s, const std::vector< bool > *&__b) const
- bool retrieve (const std::string &__s, const std::vector< bool > *&__b, const std::vector< bool > *__def) const
- bool retrieve (const std::string &__s, const std::vector< int > *&__b) const
- bool retrieve (const std::string &__s, const std::vector< int > *&__b, const std::vector< int > *__def) const
- bool retrieve (const std::string &__s, const std::vector< unsigned int > *&__b) const
- bool retrieve (const std::string &__s, const std::vector< unsigned int > *&__b, const std::vector< unsigned int > *__def) const
- bool retrieve (const std::string &__s, const std::vector< double > *&__b) const
- bool retrieve (const std::string &__s, const std::vector< double > *&__b, const std::vector< double > *__def) const
- bool retrieve (const std::string &__s, const std::vector< interval > *&__b) const
- bool retrieve (const std::string &__s, const std::vector< interval > *&__b, const std::vector< interval > *__def) const
- bool retrieve (const std::string &__s, const matrix< double > *&__b) const
- bool retrieve (const std::string &__s, const matrix< double > *&__b, const matrix< double > *__def) const
- bool retrieve (const std::string &__s, const matrix< int > *&__b) const
- bool retrieve (const std::string &__s, const matrix< int > *&__b, const matrix< int > *__def) const
- bool retrieve (const std::string &__s, const matrix< interval > *&__b) const
- bool retrieve (const std::string &__s, const matrix< interval > *&__b, const matrix< interval > *__-def) const
- bool retrieve (const char *__s, bool &__b) const
- bool retrieve (const char *__s, bool &__b, bool __def) const
- bool retrieve (const char *__s, int &__d) const
- bool retrieve (const char *__s, int &__d, int __def) const
- bool retrieve (const char *__s, unsigned int &__d) const
- bool retrieve (const char *__s, unsigned int &__d, unsigned int __def) const

- bool retrieve (const char *__s, double &__d) const
- bool retrieve (const char *__s, double &__d, double __def) const
- bool retrieve (const char *__s, interval &__b) const
- bool retrieve (const char *__s, interval &__b, const interval &__def) const
- bool retrieve (const char *__s, std::string &__is) const
- bool retrieve (const char *__s, std::string &__b, const std::string &__def) const
- bool retrieve (const char *__s, const std::vector< bool > *&__b) const
- bool retrieve (const char *__s, const std::vector< bool > *&__b, const std::vector< bool > *__def) const
- bool retrieve (const char *__s, const std::vector< unsigned int > *&__b) const
- bool retrieve (const char *__s, const std::vector< unsigned int > *&__b, const std::vector< unsigned int > *__def) const
- bool retrieve (const char *__s, const std::vector< int > *&__b) const
- bool retrieve (const char *__s, const std::vector< int > *&__b, const std::vector< int > *__def) const
- bool retrieve (const char *__s, const std::vector< double > *&__b) const
- bool retrieve (const char *__s, const std::vector< double > *&__b, const std::vector< double > *__def) const
- bool retrieve (const char *__s, const std::vector< interval > *&__b) const
- bool retrieve (const char *__s, const std::vector< interval > *&__b, const std::vector< interval > *__def) const
- bool retrieve (const char *__s, const matrix< double > *&__b) const
- bool retrieve (const char *__s, const matrix< double > *&__b, const matrix< double > *__def) const
- bool retrieve (const char *__s, const matrix< int > *&__b) const
- bool retrieve (const char *__s, const matrix< int > *&__b, const matrix< int > *__def) const
- bool retrieve (const char *__s, const matrix< interval > *&__b) const
- bool retrieve (const char *__s, const matrix< interval > *&__b, const matrix< interval > *__def) const
- bool retrieve_i (const std::string &__s, int i, bool &__b) const
- bool retrieve_i (const std::string &__s, int i, bool &__b, bool __def) const
- bool retrieve_i (const std::string &__s, int i, int &__d) const
- bool retrieve_i (const std::string &__s, int i, int &__d, int __def) const
- bool retrieve_i (const std::string &__s, int i, unsigned int &__d) const
- bool retrieve_i (const std::string &__s, int i, unsigned int &__d, unsigned int __def) const
- bool retrieve_i (const std::string &__s, int i, double &__d) const
- bool retrieve_i (const std::string &__s, int i, double &__d, double __def) const
- bool retrieve_i (const std::string &__s, int i, interval &__b) const
- bool retrieve_i (const std::string &__s, int i, interval &__b, const interval &__def) const
- bool retrieve_i (const std::string &__s, int i, std::string &__is) const
- bool retrieve_i (const std::string &__s, int i, std::string &__is, const std::string &__def) const
- bool retrieve_i (const std::string &__s, int i, const std::vector< bool > *&__b) const
- bool retrieve_i (const std::string &__s, int i, const std::vector< bool > *&__b, const std::vector< bool > *__def) const
- bool retrieve_i (const std::string &__s, int i, const std::vector< int > *&__b) const
- bool retrieve_i (const std::string &__s, int i, const std::vector< int > *&__b, const std::vector< int > *__def) const
- bool retrieve_i (const std::string &__s, int i, const std::vector< unsigned int > *&__b) const
- bool retrieve_i (const std::string &__s, int i, const std::vector< unsigned int > *&__b, const std::vector< unsigned int > *__def) const
- bool retrieve_i (const std::string &__s, int i, const std::vector< double > *&__b) const
- bool retrieve_i (const std::string &__s, int i, const std::vector< double > *&__b, const std::vector< double > *__def) const

- bool retrieve_i (const std::string &__s, int i, const std::vector< interval > *&__b) const
- bool retrieve_i (const std::string &__s, int i, const std::vector< interval > *&__b, const std::vector< interval > *__def) const
- bool retrieve_i (const std::string &__s, int i, const matrix< double > *&__b) const
- bool retrieve_i (const std::string &__s, int i, const matrix< double > *&__b, const matrix< double > *__def) const
- bool retrieve_i (const std::string &__s, int i, const matrix< int > *&__b) const
- bool retrieve_i (const std::string &__s, int i, const matrix< int > *&__b, const matrix< int > *__def) const
- bool retrieve_i (const std::string &__s, int i, const matrix< interval > *&__b) const
- bool retrieve_i (const std::string &__s, int i, const matrix< interval > *&__b, const matrix< interval > *__def) const
- bool retrieve_i (const char *__s, int i, bool &__b) const
- bool retrieve_i (const char *__s, int i, bool &__b, bool __def) const
- bool retrieve_i (const char *__s, int i, int &__d) const
- bool retrieve_i (const char *__s, int i, int &__d, int __def) const
- bool retrieve_i (const char *__s, int i, unsigned int &__d) const
- bool retrieve_i (const char *__s, int i, unsigned int &__d, unsigned int __def) const
- bool retrieve_i (const char *__s, int i, double &__d) const
- bool retrieve_i (const char *__s, int i, double &__d, double __def) const
- bool retrieve_i (const char *__s, int i, interval &__b) const
- bool retrieve_i (const char *__s, int i, interval &__b, const interval &__def) const
- bool retrieve_i (const char *__s, int i, std::string &__is) const
- bool retrieve_i (const char *__s, int i, std::string &__b, const std::string &__def) const
- bool retrieve_i (const char *__s, int i, const std::vector< bool > *&__b) const
- bool retrieve_i (const char *__s, int i, const std::vector< bool > *&__b, const std::vector< bool > *__def) const
- bool retrieve_i (const char *__s, int i, const std::vector< int > *&__b) const
- bool retrieve_i (const char *__s, int i, const std::vector< int > *&__b, const std::vector< int > *__def) const
- bool retrieve_i (const char *__s, int i, const std::vector< unsigned int > *&__b) const
- bool retrieve_i (const char *__s, int i, const std::vector< unsigned int > *&__b, const std::vector< unsigned int > *__def) const
- bool retrieve_i (const char *__s, int i, const std::vector< double > *&__b) const
- bool retrieve_i (const char *__s, int i, const std::vector< double > *&__b, const std::vector< double > *__def) const
- bool retrieve_i (const char *__s, int i, const std::vector< interval > *&__b) const
- bool retrieve_i (const char *__s, int i, const std::vector< interval > *&__b, const std::vector< interval > *__def) const
- bool retrieve_i (const char *__s, int i, const matrix< double > *&__b) const
- bool retrieve_i (const char *__s, int i, const matrix< double > *&__b, const matrix< double > *__def) const
- bool retrieve_i (const char *__s, int i, const matrix< int > *&__b) const
- bool retrieve_i (const char *__s, int i, const matrix< int > *&__b, const matrix< int > *__def) const
- bool retrieve_i (const char *__s, int i, const matrix< interval > *&__b) const
- bool retrieve_i (const char *__s, int i, const matrix< interval > *&__b, const matrix< interval > *__def) const

### 6.69.1 Constructor & Destructor Documentation

#### 6.69.1.1 info_contents::info_contents () [inline]

Definition at line 40 of file info_contents.h.

#### 6.69.1.2 info_contents::info_contents (const std::string & __n, const additional_info_u & __v) [inline]

Definition at line 41 of file info_contents.h.

#### 6.69.1.3 info_contents::info_contents (const char * __n, const additional_info_u & __v) [inline]

Definition at line 43 of file info_contents.h.

#### 6.69.1.4 info_contents::info_contents (const info_contents & __c) [inline]

Definition at line 45 of file info_contents.h.

#### 6.69.1.5 virtual info_contents::~info_contents () [inline, virtual]

Definition at line 46 of file info_contents.h.

### 6.69.2 Member Function Documentation

#### 6.69.2.1 bool datamap::defd (const char * __cp, int i) const [inline, inherited]

Definition at line 121 of file datamap-inline.h.

#### 6.69.2.2 bool datamap::defd (const std::string & __s, int i) const [inline, inherited]

Definition at line 116 of file datamap-inline.h.

#### 6.69.2.3 bool datamap::defd (const char * __cp) const [inline, inherited]

Definition at line 111 of file datamap-inline.h.

#### 6.69.2.4 bool datamap::defd (const std::string & __s) const [inline, inherited]

Definition at line 106 of file datamap-inline.h.

#### 6.69.2.5 info_contents& info_contents::operator= (const info_contents & __c) [inline]

Definition at line 48 of file info_contents.h.

#### 6.69.2.6 void datamap::remove (const char * __cp, int i) [inline, inherited]

Definition at line 73 of file datamap-inline.h.

#### 6.69.2.7 void datamap::remove (const std::string & __s, int i) [inline, inherited]

Definition at line 67 of file datamap-inline.h.

#### 6.69.2.8 void datamap::remove (const char * __cp) [inline, inherited]

Definition at line 62 of file datamap-inline.h.

#### 6.69.2.9 void datamap::remove (const std::string & __s) [inline, inherited]

Definition at line 54 of file datamap-inline.h.

#### 6.69.2.10 bool datamap::retrieve (const char * __s, const matrix< interval > *& __b, const matrix< interval > * __def) const [inline, inherited]

Definition at line 597 of file datamap-inline.h.

#### 6.69.2.11 bool datamap::retrieve (const char * __s, const matrix< interval > *& __b) const [inline, inherited]

Definition at line 592 of file datamap-inline.h.

#### 6.69.2.12 bool datamap::retrieve (const char * __s, const matrix< int > *& __b, const matrix< int > * __def) const [inline, inherited]

Definition at line 586 of file datamap-inline.h.

#### 6.69.2.13 bool datamap::retrieve (const char * __s, const matrix< int > *& __b) const [inline, inherited]

Definition at line 581 of file datamap-inline.h.

#### 6.69.2.14 bool datamap::retrieve (const char * __s, const matrix< double > *& __b, const matrix< double > * __def) const [inline, inherited]

Definition at line 575 of file datamap-inline.h.

#### 6.69.2.15 bool datamap::retrieve (const char * __s, const matrix< double > *& __b) const [inline, inherited]

Definition at line 570 of file datamap-inline.h.

#### 6.69.2.16 bool datamap::retrieve (const char * __s, const std::vector< interval > *& __b, const std::vector< interval > * __def) const [inline, inherited]

Definition at line 565 of file datamap-inline.h.

#### 6.69.2.17 bool datamap::retrieve (const char * __s, const std::vector< interval > *& __b) const [inline, inherited]

Definition at line 560 of file datamap-inline.h.

#### 6.69.2.18 bool datamap::retrieve (const char * __s, const std::vector< double > *& __b, const std::vector< double > * __def) const [inline, inherited]

Definition at line 555 of file datamap-inline.h.

**6.69.2.19 bool datamap::retrieve (const char * __s, const std::vector< double > *& __b) const** `[inline, inherited]`

Definition at line 550 of file datamap-inline.h.

**6.69.2.20 bool datamap::retrieve (const char * __s, const std::vector< int > *& __b, const std::vector< int > * __def) const** `[inline, inherited]`

Definition at line 535 of file datamap-inline.h.

**6.69.2.21 bool datamap::retrieve (const char * __s, const std::vector< int > *& __b) const** `[inline, inherited]`

Definition at line 530 of file datamap-inline.h.

**6.69.2.22 bool datamap::retrieve (const char * __s, const std::vector< unsigned int > *& __b, const std::vector< unsigned int > * __def) const** `[inline, inherited]`

Definition at line 545 of file datamap-inline.h.

**6.69.2.23 bool datamap::retrieve (const char * __s, const std::vector< unsigned int > *& __b) const** `[inline, inherited]`

Definition at line 540 of file datamap-inline.h.

**6.69.2.24 bool datamap::retrieve (const char * __s, const std::vector< bool > *& __b, const std::vector< bool > * __def) const** `[inline, inherited]`

Definition at line 525 of file datamap-inline.h.

**6.69.2.25 bool datamap::retrieve (const char * __s, const std::vector< bool > *& __b) const** `[inline, inherited]`

Definition at line 520 of file datamap-inline.h.

**6.69.2.26 bool datamap::retrieve (const char * __s, std::string & __b, const std::string & __def) const** `[inline, inherited]`

Definition at line 515 of file datamap-inline.h.

**6.69.2.27 bool datamap::retrieve (const char * __s, std::string & __is) const** `[inline, inherited]`

Definition at line 510 of file datamap-inline.h.

**6.69.2.28 bool datamap::retrieve (const char * __s, interval & __b, const interval & __def) const** `[inline, inherited]`

Definition at line 505 of file datamap-inline.h.

**6.69.2.29 bool datamap::retrieve (const char * __s, interval & __b) const** `[inline, inherited]`

Definition at line 500 of file datamap-inline.h.

**6.69.2.30 bool datamap::retrieve (const char * __s, double & __d, double __def) const** `[inline, inherited]`

Definition at line 495 of file datamap-inline.h.

**6.69.2.31 bool datamap::retrieve (const char * __s, double & __d) const** `[inline, inherited]`

Definition at line 490 of file datamap-inline.h.

**6.69.2.32 bool datamap::retrieve (const char * __s, unsigned int & __d, unsigned int __def) const** `[inline, inherited]`

Definition at line 485 of file datamap-inline.h.

**6.69.2.33 bool datamap::retrieve (const char * __s, unsigned int & __d) const** `[inline, inherited]`

Definition at line 480 of file datamap-inline.h.

**6.69.2.34 bool datamap::retrieve (const char * __s, int & __d, int __def) const** `[inline, inherited]`

Definition at line 475 of file datamap-inline.h.

**6.69.2.35 bool datamap::retrieve (const char * __s, int & __d) const** `[inline, inherited]`

Definition at line 470 of file datamap-inline.h.

**6.69.2.36 bool datamap::retrieve (const char * __s, bool & __b, bool __def) const** `[inline, inherited]`

Definition at line 465 of file datamap-inline.h.

**6.69.2.37 bool datamap::retrieve (const char * __s, bool & __b) const** `[inline, inherited]`

Definition at line 460 of file datamap-inline.h.

**6.69.2.38 bool datamap::retrieve (const std::string & __s, const matrix< interval > *& __b, const matrix< interval > * __def) const** `[inline, inherited]`

Definition at line 449 of file datamap-inline.h.

**6.69.2.39 bool datamap::retrieve (const std::string & __s, const matrix< interval > *& __b) const** `[inline, inherited]`

Definition at line 437 of file datamap-inline.h.

**6.69.2.40 bool datamap::retrieve (const std::string & __s, const matrix< int > *& __b, const matrix< int > * __def) const** `[inline, inherited]`

Definition at line 427 of file datamap-inline.h.

**6.69.2.41 bool datamap::retrieve (const std::string & __s, const matrix< int > *& __b) const** `[inline, inherited]`

Definition at line 415 of file datamap-inline.h.

**6.69.2.42 bool datamap::retrieve (const std::string & __s, const matrix< double > *& __b, const matrix< double > * __def) const** `[inline, inherited]`

Definition at line 405 of file datamap-inline.h.

**6.69.2.43 bool datamap::retrieve (const std::string & __s, const matrix< double > *& __b) const** `[inline, inherited]`

Definition at line 393 of file datamap-inline.h.

**6.69.2.44 bool datamap::retrieve (const std::string & __s, const std::vector< interval > *& __b, const std::vector< interval > * __def) const** `[inline, inherited]`

Definition at line 383 of file datamap-inline.h.

**6.69.2.45 bool datamap::retrieve (const std::string & __s, const std::vector< interval > *& __b) const** `[inline, inherited]`

Definition at line 371 of file datamap-inline.h.

**6.69.2.46 bool datamap::retrieve (const std::string & __s, const std::vector< double > *& __b, const std::vector< double > * __def) const** `[inline, inherited]`

Definition at line 361 of file datamap-inline.h.

**6.69.2.47 bool datamap::retrieve (const std::string & __s, const std::vector< double > *& __b) const** `[inline, inherited]`

Definition at line 349 of file datamap-inline.h.

**6.69.2.48 bool datamap::retrieve (const std::string & __s, const std::vector< unsigned int > *& __b, const std::vector< unsigned int > * __def) const** `[inline, inherited]`

Definition at line 339 of file datamap-inline.h.

**6.69.2.49 bool datamap::retrieve (const std::string & __s, const std::vector< unsigned int > *& __b) const** `[inline, inherited]`

Definition at line 327 of file datamap-inline.h.

**6.69.2.50 bool datamap::retrieve (const std::string & __s, const std::vector< int > *& __b, const std::vector< int > * __def) const** `[inline, inherited]`

Definition at line 317 of file datamap-inline.h.

**6.69.2.51 bool datamap::retrieve (const std::string & __s, const std::vector< int > *& __b) const** `[inline, inherited]`

Definition at line 305 of file datamap-inline.h.

**6.69.2.52 bool datamap::retrieve (const std::string & __s, const std::vector< bool > *& __b, const std::vector< bool > * __def) const** `[inline, inherited]`

Definition at line 295 of file datamap-inline.h.

**6.69.2.53 bool datamap::retrieve (const std::string & __s, const std::vector< bool > *& __b) const** `[inline, inherited]`

Definition at line 283 of file datamap-inline.h.

**6.69.2.54 bool datamap::retrieve (const std::string & __s, std::string & __is, const std::string & __def) const** `[inline, inherited]`

Definition at line 273 of file datamap-inline.h.

**6.69.2.55 bool datamap::retrieve (const std::string & __s, std::string & __is) const** `[inline, inherited]`

Definition at line 261 of file datamap-inline.h.

**6.69.2.56 bool datamap::retrieve (const std::string & __s, interval & __b, const interval & __def) const** `[inline, inherited]`

Definition at line 251 of file datamap-inline.h.

**6.69.2.57 bool datamap::retrieve (const std::string & __s, interval & __b) const** `[inline, inherited]`

Definition at line 239 of file datamap-inline.h.

**6.69.2.58 bool datamap::retrieve (const std::string & __s, double & __d, double __def) const** `[inline, inherited]`

Definition at line 229 of file datamap-inline.h.

**6.69.2.59 bool datamap::retrieve (const std::string & __s, double & __d) const** `[inline, inherited]`

Definition at line 217 of file datamap-inline.h.

**6.69.2.60 bool datamap::retrieve (const std::string & __s, unsigned int & __d, unsigned int __def) const** `[inline, inherited]`

Definition at line 207 of file datamap-inline.h.

**6.69.2.61 bool datamap::retrieve (const std::string & __s, unsigned int & __d) const** `[inline, inherited]`

Definition at line 195 of file datamap-inline.h.

**6.69.2.62 bool datamap::retrieve (const std::string & __s, int & __d, int __def) const** `[inline, inherited]`

Definition at line 185 of file datamap-inline.h.

**6.69.2.63 bool datamap::retrieve (const std::string & __s, int & __d) const** `[inline, inherited]`

Definition at line 173 of file datamap-inline.h.

**6.69.2.64 bool datamap::retrieve (const std::string & __s, bool & __b, bool __def) const** `[inline, inherited]`

Definition at line 163 of file datamap-inline.h.

**6.69.2.65 bool datamap::retrieve (const std::string & __s, bool & __b) const** `[inline, inherited]`

Definition at line 146 of file datamap-inline.h.

**6.69.2.66 bool datamap::retrieve_i (const char * __s, int *i*, const matrix< interval > *& __b, const matrix< interval > * __def) const** `[inline, inherited]`

Definition at line 1021 of file datamap-inline.h.

**6.69.2.67 bool datamap::retrieve_i (const char * __s, int *i*, const matrix< interval > *& __b) const** `[inline, inherited]`

Definition at line 1016 of file datamap-inline.h.

**6.69.2.68 bool datamap::retrieve_i (const char * __s, int *i*, const matrix< int > *& __b, const matrix< int > * __def) const** `[inline, inherited]`

Definition at line 1010 of file datamap-inline.h.

**6.69.2.69 bool datamap::retrieve_i (const char * __s, int *i*, const matrix< int > *& __b) const** `[inline, inherited]`

Definition at line 1005 of file datamap-inline.h.

**6.69.2.70 bool datamap::retrieve_i (const char * __s, int i, const matrix< double > *& __b, const matrix< double > * __def) const** `[inline, inherited]`

Definition at line 999 of file datamap-inline.h.

**6.69.2.71 bool datamap::retrieve_i (const char * __s, int i, const matrix< double > *& __b) const** `[inline, inherited]`

Definition at line 994 of file datamap-inline.h.

**6.69.2.72 bool datamap::retrieve_i (const char * __s, int i, const std::vector< interval > *& __b, const std::vector< interval > * __def) const** `[inline, inherited]`

Definition at line 989 of file datamap-inline.h.

**6.69.2.73 bool datamap::retrieve_i (const char * __s, int i, const std::vector< interval > *& __b) const** `[inline, inherited]`

Definition at line 984 of file datamap-inline.h.

**6.69.2.74 bool datamap::retrieve_i (const char * __s, int i, const std::vector< double > *& __b, const std::vector< double > * __def) const** `[inline, inherited]`

Definition at line 979 of file datamap-inline.h.

**6.69.2.75 bool datamap::retrieve_i (const char * __s, int i, const std::vector< double > *& __b) const** `[inline, inherited]`

Definition at line 974 of file datamap-inline.h.

**6.69.2.76 bool datamap::retrieve_i (const char * __s, int i, const std::vector< unsigned int > *& __b, const std::vector< unsigned int > * __def) const** `[inline, inherited]`

Definition at line 969 of file datamap-inline.h.

**6.69.2.77 bool datamap::retrieve_i (const char * __s, int i, const std::vector< unsigned int > *& __b) const** `[inline, inherited]`

Definition at line 964 of file datamap-inline.h.

**6.69.2.78 bool datamap::retrieve_i (const char * __s, int i, const std::vector< int > *& __b, const std::vector< int > * __def) const** `[inline, inherited]`

Definition at line 959 of file datamap-inline.h.

**6.69.2.79 bool datamap::retrieve_i (const char * __s, int i, const std::vector< int > *& __b) const** `[inline, inherited]`

Definition at line 954 of file datamap-inline.h.

**6.69.2.80 bool datamap::retrieve_i (const char * __s, int i, const std::vector< bool > *& __b, const std::vector< bool > * __def) const** `[inline, inherited]`

Definition at line 949 of file datamap-inline.h.

**6.69.2.81 bool datamap::retrieve_i (const char * __s, int i, const std::vector< bool > *& __b) const** `[inline, inherited]`

Definition at line 944 of file datamap-inline.h.

**6.69.2.82 bool datamap::retrieve_i (const char * __s, int i, std::string & __b, const std::string & __def) const** `[inline, inherited]`

Definition at line 939 of file datamap-inline.h.

**6.69.2.83 bool datamap::retrieve_i (const char * __s, int i, std::string & __is) const** `[inline, inherited]`

Definition at line 934 of file datamap-inline.h.

**6.69.2.84 bool datamap::retrieve_i (const char * __s, int i, interval & __b, const interval & __def) const** `[inline, inherited]`

Definition at line 929 of file datamap-inline.h.

**6.69.2.85 bool datamap::retrieve_i (const char * __s, int i, interval & __b) const** `[inline, inherited]`

Definition at line 924 of file datamap-inline.h.

**6.69.2.86 bool datamap::retrieve_i (const char * __s, int i, double & __d, double __def) const** `[inline, inherited]`

Definition at line 919 of file datamap-inline.h.

**6.69.2.87 bool datamap::retrieve_i (const char * __s, int i, double & __d) const** `[inline, inherited]`

Definition at line 914 of file datamap-inline.h.

**6.69.2.88 bool datamap::retrieve_i (const char * __s, int i, unsigned int & __d, unsigned int __def) const** `[inline, inherited]`

Definition at line 909 of file datamap-inline.h.

**6.69.2.89 bool datamap::retrieve_i (const char * __s, int i, unsigned int & __d) const** `[inline, inherited]`

Definition at line 904 of file datamap-inline.h.

**6.69.2.90 bool datamap::retrieve_i (const char * __s, int i, int & __d, int __def) const** `[inline, inherited]`

Definition at line 899 of file datamap-inline.h.

**6.69.2.91 bool datamap::retrieve_i (const char * __s, int i, int & __d) const** `[inline, inherited]`

Definition at line 894 of file datamap-inline.h.

**6.69.2.92 bool datamap::retrieve_i (const char * __s, int i, bool & __b, bool __def) const** `[inline, inherited]`

Definition at line 889 of file datamap-inline.h.

**6.69.2.93 bool datamap::retrieve_i (const char * __s, int i, bool & __b) const** `[inline, inherited]`

Definition at line 884 of file datamap-inline.h.

**6.69.2.94 bool datamap::retrieve_i (const std::string & __s, int i, const matrix< interval > *& __b, const matrix< interval > * __def) const** `[inline, inherited]`

Definition at line 874 of file datamap-inline.h.

**6.69.2.95 bool datamap::retrieve_i (const std::string & __s, int i, const matrix< interval > *& __b) const** `[inline, inherited]`

Definition at line 863 of file datamap-inline.h.

**6.69.2.96 bool datamap::retrieve_i (const std::string & __s, int i, const matrix< int > *& __b, const matrix< int > * __def) const** `[inline, inherited]`

Definition at line 854 of file datamap-inline.h.

**6.69.2.97 bool datamap::retrieve_i (const std::string & __s, int i, const matrix< int > *& __b) const** `[inline, inherited]`

Definition at line 843 of file datamap-inline.h.

**6.69.2.98 bool datamap::retrieve_i (const std::string & __s, int i, const matrix< double > *& __b, const matrix< double > * __def) const** `[inline, inherited]`

Definition at line 834 of file datamap-inline.h.

**6.69.2.99 bool datamap::retrieve_i (const std::string & __s, int i, const matrix< double > *& __b) const** `[inline, inherited]`

Definition at line 823 of file datamap-inline.h.

**6.69.2.100 bool datamap::retrieve_i (const std::string & __s, int i, const std::vector< interval > *& __b, const std::vector< interval > * __def) const** `[inline, inherited]`

Definition at line 814 of file datamap-inline.h.

**6.69.2.101 bool datamap::retrieve_i (const std::string & __s, int i, const std::vector< interval > *& __b) const** `[inline, inherited]`

Definition at line 803 of file datamap-inline.h.

**6.69.2.102 bool datamap::retrieve_i (const std::string & __s, int i, const std::vector< double > *& __b, const std::vector< double > * __def) const** `[inline, inherited]`

Definition at line 794 of file datamap-inline.h.

**6.69.2.103 bool datamap::retrieve_i (const std::string & __s, int i, const std::vector< double > *& __b) const** `[inline, inherited]`

Definition at line 783 of file datamap-inline.h.

**6.69.2.104 bool datamap::retrieve_i (const std::string & __s, int i, const std::vector< unsigned int > *& __b, const std::vector< unsigned int > * __def) const** `[inline, inherited]`

Definition at line 774 of file datamap-inline.h.

**6.69.2.105 bool datamap::retrieve_i (const std::string & __s, int i, const std::vector< unsigned int > *& __b) const** `[inline, inherited]`

Definition at line 763 of file datamap-inline.h.

**6.69.2.106 bool datamap::retrieve_i (const std::string & __s, int i, const std::vector< int > *& __b, const std::vector< int > * __def) const** `[inline, inherited]`

Definition at line 754 of file datamap-inline.h.

**6.69.2.107 bool datamap::retrieve_i (const std::string & __s, int i, const std::vector< int > *& __b) const** `[inline, inherited]`

Definition at line 743 of file datamap-inline.h.

**6.69.2.108 bool datamap::retrieve_i (const std::string & __s, int i, const std::vector< bool > *& __b, const std::vector< bool > * __def) const** `[inline, inherited]`

Definition at line 734 of file datamap-inline.h.

**6.69.2.109 bool datamap::retrieve_i (const std::string & __s, int i, const std::vector< bool > *& __b) const** `[inline, inherited]`

Definition at line 723 of file datamap-inline.h.

**6.69.2.110 bool datamap::retrieve_i (const std::string & __s, int *i*, std::string & __is, const std::string & __def) const** `[inline, inherited]`

Definition at line 714 of file datamap-inline.h.

**6.69.2.111 bool datamap::retrieve_i (const std::string & __s, int *i*, std::string & __is) const** `[inline, inherited]`

Definition at line 703 of file datamap-inline.h.

**6.69.2.112 bool datamap::retrieve_i (const std::string & __s, int *i*, interval & __b, const interval & __def) const** `[inline, inherited]`

Definition at line 694 of file datamap-inline.h.

**6.69.2.113 bool datamap::retrieve_i (const std::string & __s, int *i*, interval & __b) const** `[inline, inherited]`

Definition at line 683 of file datamap-inline.h.

**6.69.2.114 bool datamap::retrieve_i (const std::string & __s, int *i*, double & __d, double __def) const** `[inline, inherited]`

Definition at line 674 of file datamap-inline.h.

**6.69.2.115 bool datamap::retrieve_i (const std::string & __s, int *i*, double & __d) const** `[inline, inherited]`

Definition at line 663 of file datamap-inline.h.

**6.69.2.116 bool datamap::retrieve_i (const std::string & __s, int *i*, unsigned int & __d, unsigned int __def) const** `[inline, inherited]`

Definition at line 654 of file datamap-inline.h.

**6.69.2.117 bool datamap::retrieve_i (const std::string & __s, int *i*, unsigned int & __d) const** `[inline, inherited]`

Definition at line 643 of file datamap-inline.h.

**6.69.2.118 bool datamap::retrieve_i (const std::string & __s, int *i*, int & __d, int __def) const** `[inline, inherited]`

Definition at line 634 of file datamap-inline.h.

**6.69.2.119 bool datamap::retrieve_i (const std::string & __s, int *i*, int & __d) const** `[inline, inherited]`

Definition at line 623 of file datamap-inline.h.

**6.69.2.120 bool datamap::retrieve_i (const std::string & __s, int *i*, bool & __b, bool __def) const** `[inline, inherited]`

Definition at line 614 of file datamap-inline.h.

**6.69.2.121 bool datamap::retrieve_i (const std::string & __s, int *i*, bool & __b) const** `[inline, inherited]`

Definition at line 603 of file datamap-inline.h.

**6.69.2.122 const additional_info_u & datamap::sfind (const char * __cp, int *i*) const** `[inline, inherited]`

Definition at line 101 of file datamap-inline.h.

**6.69.2.123 const additional_info_u & datamap::sfind (const std::string & __s, int *i*) const** `[inline, inherited]`

Definition at line 92 of file datamap-inline.h.

**6.69.2.124 const additional_info_u & datamap::sfind (const char * __cp) const** `[inline, inherited]`

Definition at line 87 of file datamap-inline.h.

**6.69.2.125 const additional_info_u & datamap::sfind (const std::string & __s) const** `[inline, inherited]`

Definition at line 78 of file datamap-inline.h.

**6.69.2.126 bool datamap::sinsert (const char * __cp, int *i*, const additional_info_u & __h, bool *replace*)** `[inline, inherited]`

Definition at line 47 of file datamap-inline.h.

**6.69.2.127 bool datamap::sinsert (const std::string & __s, int *i*, const additional_info_u & __h, bool *replace*)** `[inline, inherited]`

Definition at line 39 of file datamap-inline.h.

**6.69.2.128 bool datamap::sinsert (const char * __cp, const additional_info_u & __h, bool *replace*)** `[inline, inherited]`

Definition at line 33 of file datamap-inline.h.

**6.69.2.129 bool datamap::sinsert (const std::string & __s, const additional_info_u & __h, bool *replace*)** `[inherited]`

Definition at line 29 of file datamap.cc.

#### 6.69.2.130 bool datamap::which (const char * __cp, std::vector< int > & __idx) const [inline, inherited]

Definition at line 140 of file datamap-inline.h.

#### 6.69.2.131 bool datamap::which (const std::string & __s, std::vector< int > & __idx) const [inline, inherited]

Definition at line 126 of file datamap-inline.h.

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

- info_contents.h

## 6.70 interval Class Reference

`#include <interval.h>`

### Public Methods

- interval (const double &lo, const double &up)
- interval (const double &__d=0)
- interval (const int &__d)
- interval (const unsigned int &__d)
- interval (const long int &__d)
- interval (const unsigned long int &__d)
- interval (const __Base &__i)
- interval (const interval &__i)
- interval (const interval_st &__i)
- void set (double lo, double up)
- bool empty () const
- bool is_thin () const
- bool is_unbounded_below () const
- bool is_unbounded_above () const
- bool is_entire () const
- bool is_bounded () const
- double rel_width () const
- interval & intersect (const interval &__i)
- interval & hulldiff (const interval &__i)
- bool contains (const interval &__i) const
- void setpair (double &__l, double &__u) const
- interval & operator= (const double &__d)
- interval & operator= (const int &__d)
- interval & operator= (const unsigned int &__d)
- interval & operator= (const long int &__d)
- interval & operator= (const unsigned long int &__d)
- interval & ipow (int n)
- interval & intersect_div (const interval &__i, const interval &__j)
- interval & intersect_power (const interval &__i, const int &__n)
- interval & intersect_invsqr_wc (const interval &__i, const double &__l, const double &__d)
- interval & intersect_invpower_wc (const interval &__i, const double &__l, const int &__n)

- interval & intersect_invsin_wc (const interval &__i, const double &__l, const double &__d)
- interval & intersect_invcos_wc (const interval &__i, const double &__l, const double &__d)
- interval & intersect_invgauss_wc (const interval &__i, const double &__l, const double &__m, const double &__s)
- double project (const double &__d)

### 6.70.1 Constructor & Destructor Documentation

#### 6.70.1.1 interval::interval (const double & *lo*, const double & *up*) [inline]

Definition at line 59 of file interval.h.

#### 6.70.1.2 interval::interval (const double & __*d* = 0) [inline]

Definition at line 61 of file interval.h.

#### 6.70.1.3 interval::interval (const int & __*d*) [inline]

Definition at line 63 of file interval.h.

#### 6.70.1.4 interval::interval (const unsigned int & __*d*) [inline]

Definition at line 64 of file interval.h.

#### 6.70.1.5 interval::interval (const long int & __*d*) [inline]

Definition at line 65 of file interval.h.

#### 6.70.1.6 interval::interval (const unsigned long int & __*d*) [inline]

Definition at line 66 of file interval.h.

#### 6.70.1.7 interval::interval (const __Base & __*i*) [inline]

Definition at line 68 of file interval.h.

#### 6.70.1.8 interval::interval (const interval & __*i*) [inline]

Definition at line 70 of file interval.h.

#### 6.70.1.9 interval::interval (const interval_st & __*i*) [inline]

Definition at line 72 of file interval.h.

### 6.70.2 Member Function Documentation

#### 6.70.2.1 bool interval::contains (const interval & __*i*) const [inline]

Definition at line 111 of file interval.h.

#### 6.70.2.2 bool interval::empty () const [inline]

Definition at line 80 of file interval.h.

#### 6.70.2.3 interval& interval::hulldiff (const interval & __i) [inline]

Definition at line 97 of file interval.h.

#### 6.70.2.4 interval& interval::intersect (const interval & __i) [inline]

Definition at line 91 of file interval.h.

#### 6.70.2.5 interval & interval::intersect_div (const interval & __i, const interval & __j) [inline]

Definition at line 155 of file interval.h.

#### 6.70.2.6 interval & interval::intersect_invcos_wc (const interval & __i, const double & __l, const double & __d) [inline]

Definition at line 346 of file interval.h.

#### 6.70.2.7 interval & interval::intersect_invgauss_wc (const interval & __i, const double & __l, const double & __m, const double & __s) [inline]

Definition at line 355 of file interval.h.

#### 6.70.2.8 interval & interval::intersect_invpower_wc (const interval & __i, const double & __l, const int & __n) [inline]

Definition at line 226 of file interval.h.

#### 6.70.2.9 interval & interval::intersect_invsin_wc (const interval & __i, const double & __l, const double & __d) [inline]

Definition at line 337 of file interval.h.

#### 6.70.2.10 interval & interval::intersect_invsqr_wc (const interval & __i, const double & __l, const double & __d) [inline]

Definition at line 208 of file interval.h.

#### 6.70.2.11 interval & interval::intersect_power (const interval & __i, const int & __n) [inline]

Definition at line 179 of file interval.h.

#### 6.70.2.12 interval& interval::ipow (int n) [inline]

Definition at line 128 of file interval.h.

#### 6.70.2.13 bool interval::is_bounded () const [inline]

Definition at line 85 of file interval.h.

#### 6.70.2.14 bool interval::is_entire () const [inline]

Definition at line 84 of file interval.h.

#### 6.70.2.15 bool interval::is_thin () const [inline]

Definition at line 81 of file interval.h.

#### 6.70.2.16 bool interval::is_unbounded_above () const [inline]

Definition at line 83 of file interval.h.

#### 6.70.2.17 bool interval::is_unbounded_below () const [inline]

Definition at line 82 of file interval.h.

#### 6.70.2.18 interval& interval::operator= (const unsigned long int & __d) [inline]

Definition at line 125 of file interval.h.

#### 6.70.2.19 interval& interval::operator= (const long int & __d) [inline]

Definition at line 124 of file interval.h.

#### 6.70.2.20 interval& interval::operator= (const unsigned int & __d) [inline]

Definition at line 123 of file interval.h.

#### 6.70.2.21 interval& interval::operator= (const int & __d) [inline]

Definition at line 122 of file interval.h.

#### 6.70.2.22 interval& interval::operator= (const double & __d) [inline]

Definition at line 121 of file interval.h.

#### 6.70.2.23 double interval::project (const double & __d)

#### 6.70.2.24 double interval::rel_width () const [inline]

Definition at line 88 of file interval.h.

#### 6.70.2.25 void interval::set (double *lo*, double *up*) [inline]

Definition at line 75 of file interval.h.

#### 6.70.2.26 void interval::setpair (double & __l, double & __u) const [inline]

Definition at line 119 of file interval.h.

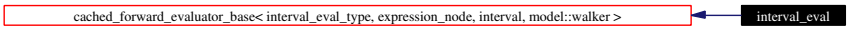
The documentation for this class was generated from the following file:

- interval.h

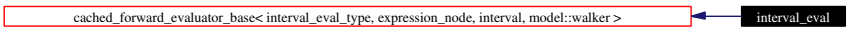
## 6.71 interval_eval Class Reference

`#include <int_evaluator.h>`

Inheritance diagram for interval_eval:




Collaboration diagram for interval_eval:




### Public Types

- typedef cached_evaluator_base< interval_eval_type, expression_node, interval, model::walker >::node_data_type node_data_type
- typedef cached_evaluator_base< interval_eval_type, expression_node, interval, model::walker >::return_value return_value
- typedef cached_evaluator_base< interval_eval_type, expression_node, interval, model::walker >::walker walker
- typedef interval_eval_type data_type

### Public Methods

- interval_eval (const std::vector< interval > &__x, const variable_indicator &__v, const model &__m, std::vector< interval > *__c, bool do_i=true)
- interval_eval (const interval_eval &__v)
- ~interval_eval ()
- model::walker short_cut_to (const expression_node &__data)
- void initialize ()
- int initialize (const expression_node &__data)
- void calculate (const expression_node &__data)
- void retrieve_from_cache (const expression_node &__data)
- int update (const interval &__rval)
- int update (const expression_node &__data, const interval &__rval)
- interval calculate_value (bool eval_all)
- int preorder (const node_data_type &__data)
- void postorder (const node_data_type &__data)
- int collect (const node_data_type &__data, const return_value &__rval)
- int vcollect (const return_value &__rval)
- return_value value ()
- return_value vvalue ()
- void vinit ()
- virtual int initialize (const node_data_type &__data)
- virtual void calculate (const node_data_type &__data)
- virtual void retrieve_from_cache (const node_data_type &__data)
- virtual void cleanup (const node_data_type &__data)
- virtual int update (const node_data_type &__data, const return_value &__rval)
- virtual int update (const return_value &__rval)
- virtual walker short_cut_to (const node_data_type &__data) PURE_VIRTUAL public

**Protected Methods**

- bool is_cached (const node_data_type &__data)

**Protected Attributes**

- const variable_indicator * v_ind
- interval_eval_type eval_data

### 6.71.1 Member Typedef Documentation

#### 6.71.1.1 typedef interval_eval_type _evaluator_base< interval_eval_type, expression_node, interval, model::walker >::data_type [inherited]

Definition at line 245 of file evaluator.h.

#### 6.71.1.2 typedef cached_evaluator_base<interval_eval_type, expression_node, interval,model::walker>::node_data_type cached_forward_evaluator_base< interval_eval_type, expression_node, interval, model::walker >::node_data_type [inherited]

Reimplemented from cached_evaluator_base< interval_eval_type, expression_node, interval, model::walker >.

Definition at line 396 of file evaluator.h.

#### 6.71.1.3 typedef cached_evaluator_base<interval_eval_type, expression_node, interval,model::walker>::return_value cached_forward_evaluator_base< interval_eval_type, expression_node, interval, model::walker >::return_value [inherited]

Reimplemented from cached_evaluator_base< interval_eval_type, expression_node, interval, model::walker >.

Definition at line 398 of file evaluator.h.

#### 6.71.1.4 typedef cached_evaluator_base<interval_eval_type, expression_node, interval,model::walker>::walker cached_forward_evaluator_base< interval_eval_type, expression_node, interval, model::walker >::walker [inherited]

Reimplemented from cached_evaluator_base< interval_eval_type, expression_node, interval, model::walker >.

Definition at line 400 of file evaluator.h.

### 6.71.2 Constructor & Destructor Documentation

#### 6.71.2.1 interval_eval::interval_eval (const std::vector< interval > & __x, const variable_indicator & __v, const model & __m, std::vector< interval > * __c, bool *do_i* = true) [inline]

Definition at line 109 of file int_evaluator.h.

#### 6.71.2.2 interval_eval::interval_eval (const interval_eval & __v) [inline]

Definition at line 122 of file int_evaluator.h.

#### 6.71.2.3 interval_eval::~interval_eval () [inline]

Definition at line 124 of file int_evaluator.h.

### 6.71.3 Member Function Documentation

#### 6.71.3.1 virtual void cached_forward_evaluator_base< interval_eval_type, expression_node, interval, model::walker >::calculate (const node_data_type & __data) [inline, virtual, inherited]

Definition at line 430 of file evaluator.h.

#### 6.71.3.2 void interval_eval::calculate (const expression_node & __data) [inline]

Definition at line 188 of file int_evaluator.h.

#### 6.71.3.3 interval interval_eval::calculate_value (bool *eval_all*) [inline, virtual]

Reimplemented from cached_forward_evaluator_base< interval_eval_type, expression_node, interval, model::walker >.

Definition at line 674 of file int_evaluator.h.

#### 6.71.3.4 virtual void cached_forward_evaluator_base< interval_eval_type, expression_node, interval, model::walker >::cleanup (const node_data_type & __data) [inline, virtual, inherited]

Definition at line 432 of file evaluator.h.

#### 6.71.3.5 int cached_forward_evaluator_base< interval_eval_type, expression_node, interval, model::walker >::collect (const node_data_type & __data, const return_value & __rval) [inline, virtual, inherited]

Reimplemented from _evaluator_base< interval_eval_type, expression_node, interval, model::walker >.

Definition at line 419 of file evaluator.h.

#### 6.71.3.6 virtual int cached_forward_evaluator_base< interval_eval_type, expression_node, interval, model::walker >::initialize (const node_data_type & __data) [inline, virtual, inherited]

Definition at line 429 of file evaluator.h.

#### 6.71.3.7 int interval_eval::initialize (const expression_node & __data) [inline]

Definition at line 131 of file int_evaluator.h.

#### 6.71.3.8 void interval_eval::initialize () [inline, virtual]

Reimplemented from cached_forward_evaluator_base< interval_eval_type, expression_node, interval, model::walker >.

Definition at line 129 of file int_evaluator.h.

#### 6.71.3.9 bool interval_eval::is_cached (const node_data_type & *__data*) [inline, protected, virtual]

Reimplemented from cached_forward_evaluator_base< interval_eval_type, expression_node, interval, model::walker >.

Definition at line 63 of file int_evaluator.h.

#### 6.71.3.10 void cached_forward_evaluator_base< interval_eval_type, expression_node, interval, model::walker >::postorder (const node_data_type & *__data*) [inline, virtual, inherited]

Reimplemented from _evaluator_base< interval_eval_type, expression_node, interval, model::walker >.

Definition at line 417 of file evaluator.h.

#### 6.71.3.11 int cached_forward_evaluator_base< interval_eval_type, expression_node, interval, model::walker >::preorder (const node_data_type & *__data*) [inline, virtual, inherited]

Reimplemented from cached_evaluator_base< interval_eval_type, expression_node, interval, model::walker >.

Definition at line 408 of file evaluator.h.

#### 6.71.3.12 virtual void cached_forward_evaluator_base< interval_eval_type, expression_node, interval, model::walker >::retrieve_from_cache (const node_data_type & *__data*) [inline, virtual, inherited]

Definition at line 431 of file evaluator.h.

#### 6.71.3.13 void interval_eval::retrieve_from_cache (const expression_node & *__data*) [inline]

Definition at line 199 of file int_evaluator.h.

#### 6.71.3.14 virtual walker cached_evaluator_base< interval_eval_type, expression_node, interval, model::walker >::short_cut_to (const node_data_type & *__data*) [inline, virtual, inherited]

Definition at line 303 of file evaluator.h.

#### 6.71.3.15 model::walker interval_eval::short_cut_to (const expression_node & *__data*) [inline]

Definition at line 126 of file int_evaluator.h.

#### 6.71.3.16 virtual int cached_forward_evaluator_base< interval_eval_type, expression_node, interval, model::walker >::update (const return_value & *__rval*) [inline, virtual, inherited]

Definition at line 435 of file evaluator.h.

#### 6.71.3.17 virtual int cached_forward_evaluator_base< interval_eval_type, expression_node, interval, model::walker >::update (const node_data_type & __data, const return_value & __rval) [inline, virtual, inherited]

Definition at line 433 of file evaluator.h.

#### 6.71.3.18 int interval_eval::update (const expression_node & __data, const interval & __rval) [inline]

Definition at line 224 of file int_evaluator.h.

#### 6.71.3.19 int interval_eval::update (const interval & __rval) [inline]

Definition at line 218 of file int_evaluator.h.

#### 6.71.3.20 return_value cached_forward_evaluator_base< interval_eval_type, expression_node, interval, model::walker >::value () [inline, virtual, inherited]

Reimplemented from _evaluator_base< interval_eval_type, expression_node, interval, model::walker >.

Definition at line 423 of file evaluator.h.

#### 6.71.3.21 int cached_forward_evaluator_base< interval_eval_type, expression_node, interval, model::walker >::vcollect (const return_value & __rval) [inline, virtual, inherited]

Reimplemented from _evaluator_base< interval_eval_type, expression_node, interval, model::walker >.

Definition at line 421 of file evaluator.h.

#### 6.71.3.22 void cached_forward_evaluator_base< interval_eval_type, expression_node, interval, model::walker >::vinit () [inline, inherited]

Definition at line 425 of file evaluator.h.

#### 6.71.3.23 return_value cached_forward_evaluator_base< interval_eval_type, expression_node, interval, model::walker >::vvalue () [inline, virtual, inherited]

Reimplemented from _evaluator_base< interval_eval_type, expression_node, interval, model::walker >.

Definition at line 424 of file evaluator.h.

### 6.71.4 Member Data Documentation

#### 6.71.4.1 interval_eval_type _evaluator_base< interval_eval_type, expression_node, interval, model::walker >::eval_data [protected, inherited]

Definition at line 252 of file evaluator.h.

#### 6.71.4.2 const variable_indicator* cached_evaluator_base< interval_eval_type, expression_node, interval, model::walker >::v_ind [protected, inherited]

Definition at line 295 of file evaluator.h.

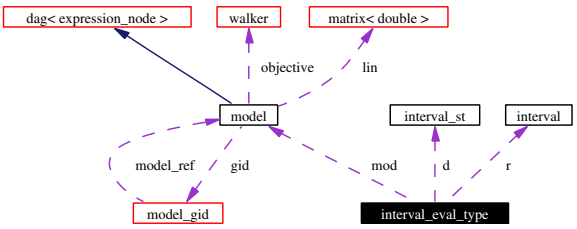
The documentation for this class was generated from the following file:

- int_evaluator.h

## 6.72 interval_eval_type Struct Reference

`#include <int_evaluator.h>`

Collaboration diagram for interval_eval_type:




### Public Attributes

- const std::vector< interval > * x
- std::vector< interval > * cache
- const model * mod
- union {
  void * p
  interval_st d
  int info
  } u
- interval r
- unsigned int n
- bool do_intersect

### 6.72.1 Member Data Documentation

#### 6.72.1.1 std::vector<interval>* interval_eval_type::cache

Definition at line 46 of file int_evaluator.h.

#### 6.72.1.2 interval_st interval_eval_type::d

Definition at line 48 of file int_evaluator.h.

#### 6.72.1.3 bool interval_eval_type::do_intersect

Definition at line 51 of file int_evaluator.h.

#### 6.72.1.4 int interval_eval_type::info

Definition at line 48 of file int_evaluator.h.

#### 6.72.1.5 const model* interval_eval_type::mod

Definition at line 47 of file int_evaluator.h.

#### 6.72.1.6 unsigned int interval_eval_type::n

Definition at line 50 of file int_evaluator.h.

#### 6.72.1.7 void* interval_eval_type::p

Definition at line 48 of file int_evaluator.h.

#### 6.72.1.8 interval interval_eval_type::r

Definition at line 49 of file int_evaluator.h.

#### 6.72.1.9 union { ... } interval_eval_type::u

#### 6.72.1.10 const std::vector<interval>* interval_eval_type::x

Definition at line 45 of file int_evaluator.h.

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

- int_evaluator.h

## 6.73 interval_st Struct Reference

`#include <interval.h>`

### Public Methods

- interval_st & operator= (const interval &__i)

### Public Attributes

- double l
- double u

### 6.73.1 Member Function Documentation

#### 6.73.1.1 interval_st & interval_st::operator= (const interval & __i) [inline]

Definition at line 147 of file interval.h.

### 6.73.2 Member Data Documentation

#### 6.73.2.1 double interval_st::l

Definition at line 40 of file interval.h.

#### 6.73.2.2 double interval_st::u

Definition at line 40 of file interval.h.

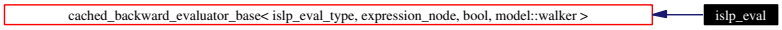
The documentation for this struct was generated from the following file:

- interval.h

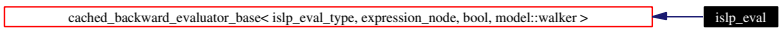
## 6.74 islp_eval Class Reference

`#include <islp_evaluator.h>`

Inheritance diagram for islp_eval:




Collaboration diagram for islp_eval:




### Public Types

- typedef cached_evaluator_base< islp_eval_type, expression_node, bool, model::walker >::node_data_type node_data_type
- typedef cached_evaluator_base< islp_eval_type, expression_node, bool, model::walker >::return_value return_value
- typedef cached_evaluator_base< islp_eval_type, expression_node, bool, model::walker >::walker walker
- typedef islp_eval_type data_type

### Public Methods

- islp_eval (std::vector< std::vector< interval > > &__der_data, variable_indicator &__v, const model &__m, std::vector< std::vector< interval > > *__d, std::vector< interval > &__islp)
- islp_eval (const islp_eval &__d)
- ~islp_eval ()
- void new_point (std::vector< std::vector< interval > > &__der_data, const variable_indicator &__v)
- void new_result (std::vector< interval > &__islp)
- void set_mult (double scal)
- model::walker short_cut_to (const expression_node &__data)
- void initialize ()
- int calculate (const expression_node &__data)
- void cleanup (const expression_node &__data)
- void retrieve_from_cache (const expression_node &__data)
- int update (const bool &__rval)
- int update (const expression_node &__data, const bool &__rval)
- bool calculate_value (bool eval_all)
- int preorder (const node_data_type &__data)
- void postorder (const node_data_type &__data)
- int collect (const node_data_type &__data, const return_value &__rval)

- int vcollect (const return_value &__rval)
- void vinit ()
- return_value value ()
- return_value vvalue ()
- virtual int calculate (const node_data_type &__data)
- virtual void cleanup (const node_data_type &__data)
- virtual void retrieve_from_cache (const node_data_type &__data)
- virtual int update (const node_data_type &__data, const return_value &__rval)
- virtual int update (const return_value &__rval)
- virtual walker short_cut_to (const node_data_type &__data) PURE_VIRTUAL public

**Protected Methods**

- bool is_cached (const node_data_type &__data)

**Protected Attributes**

- const variable_indicator * v_ind
- islp_eval_type eval_data

### 6.74.1 Member Typedef Documentation

#### 6.74.1.1 typedef islp_eval_type _evaluator_base< islp_eval_type, expression_node, bool, model::walker >::data_type [inherited]

Definition at line 245 of file evaluator.h.

#### 6.74.1.2 typedef cached_evaluator_base<islp_eval_type, expression_node, bool,model::walker>::node_data_type cached_backward_evaluator_base< islp_eval_type, expression_node, bool, model::walker >::node_data_type [inherited]

Reimplemented from cached_evaluator_base< islp_eval_type, expression_node, bool, model::walker >.

Definition at line 444 of file evaluator.h.

#### 6.74.1.3 typedef cached_evaluator_base<islp_eval_type, expression_node, bool,model::walker>::return_value cached_backward_evaluator_base< islp_eval_type, expression_node, bool, model::walker >::return_value [inherited]

Reimplemented from cached_evaluator_base< islp_eval_type, expression_node, bool, model::walker >.

Definition at line 446 of file evaluator.h.

#### 6.74.1.4 typedef cached_evaluator_base<islp_eval_type, expression_node, bool,model::walker>::walker cached_backward_evaluator_base< islp_eval_type, expression_node, bool, model::walker >::walker [inherited]

Reimplemented from cached_evaluator_base< islp_eval_type, expression_node, bool, model::walker >.

Definition at line 448 of file evaluator.h.

### 6.74.2 Constructor & Destructor Documentation

#### 6.74.2.1 islp_eval::islp_eval (std::vector< std::vector< interval > > & __der_data, variable_indicator & __v, const model & __m, std::vector< std::vector< interval > > * __d, std::vector< interval > & __islp) [inline]

Definition at line 1416 of file islp_evaluator.h.

#### 6.74.2.2 islp_eval::islp_eval (const islp_eval & __d) [inline]

Definition at line 1430 of file islp_evaluator.h.

#### 6.74.2.3 islp_eval::~islp_eval () [inline]

Definition at line 1432 of file islp_evaluator.h.

### 6.74.3 Member Function Documentation

#### 6.74.3.1 virtual int cached_backward_evaluator_base< islp_eval_type, expression_node, bool, model::walker >::calculate (const node_data_type & __data) [inline, virtual, inherited]

Definition at line 476 of file evaluator.h.

#### 6.74.3.2 int islp_eval::calculate (const expression_node & __data) [inline]

Definition at line 1462 of file islp_evaluator.h.

#### 6.74.3.3 bool islp_eval::calculate_value (bool eval_all) [inline, virtual]

Reimplemented from cached_backward_evaluator_base< islp_eval_type, expression_node, bool, model::walker >.

Definition at line 1561 of file islp_evaluator.h.

#### 6.74.3.4 virtual void cached_backward_evaluator_base< islp_eval_type, expression_node, bool, model::walker >::cleanup (const node_data_type & __data) [inline, virtual, inherited]

Definition at line 477 of file evaluator.h.

#### 6.74.3.5 void islp_eval::cleanup (const expression_node & __data) [inline]

Definition at line 1515 of file islp_evaluator.h.

#### 6.74.3.6 int cached_backward_evaluator_base< islp_eval_type, expression_node, bool, model::walker >::collect (const node_data_type & __data, const return_value & __rval) [inline, virtual, inherited]

Reimplemented from _evaluator_base< islp_eval_type, expression_node, bool, model::walker >.

Definition at line 466 of file evaluator.h.

#### 6.74.3.7 void islp_eval::initialize () [inline, virtual]

Reimplemented from cached_backward_evaluator_base< islp_eval_type, expression_node, bool, model::walker >.

Definition at line 1456 of file islp_evaluator.h.

#### 6.74.3.8 bool islp_eval::is_cached (const node_data_type & *__data*) [inline, protected, virtual]

Reimplemented from cached_backward_evaluator_base< islp_eval_type, expression_node, bool, model::walker >.

Definition at line 1403 of file islp_evaluator.h.

#### 6.74.3.9 void islp_eval::new_point (std::vector< std::vector< interval > > & *__der_data*, const variable_indicator & *__v*) [inline]

Definition at line 1434 of file islp_evaluator.h.

#### 6.74.3.10 void islp_eval::new_result (std::vector< interval > & *__islp*) [inline]

Definition at line 1441 of file islp_evaluator.h.

#### 6.74.3.11 void cached_backward_evaluator_base< islp_eval_type, expression_node, bool, model::walker >::postorder (const node_data_type & *__data*) [inline, virtual, inherited]

Reimplemented from _evaluator_base< islp_eval_type, expression_node, bool, model::walker >.

Definition at line 465 of file evaluator.h.

#### 6.74.3.12 int cached_backward_evaluator_base< islp_eval_type, expression_node, bool, model::walker >::preorder (const node_data_type & *__data*) [inline, virtual, inherited]

Reimplemented from cached_evaluator_base< islp_eval_type, expression_node, bool, model::walker >.

Definition at line 456 of file evaluator.h.

#### 6.74.3.13 virtual void cached_backward_evaluator_base< islp_eval_type, expression_node, bool, model::walker >::retrieve_from_cache (const node_data_type & *__data*) [inline, virtual, inherited]

Definition at line 478 of file evaluator.h.

#### 6.74.3.14 void islp_eval::retrieve_from_cache (const expression_node & *__data*) [inline]

Definition at line 1527 of file islp_evaluator.h.

#### 6.74.3.15 void islp_eval::set_mult (double *scal*) [inline]

Definition at line 1446 of file islp_evaluator.h.

#### 6.74.3.16 virtual walker cached_evaluator_base< islp_eval_type, expression_node, bool, model::walker >::short_cut_to (const node_data_type & __data) [inline, virtual, inherited]

Definition at line 303 of file evaluator.h.

#### 6.74.3.17 model::walker islp_eval::short_cut_to (const expression_node & __data) [inline]

Definition at line 1452 of file islp_evaluator.h.

#### 6.74.3.18 virtual int cached_backward_evaluator_base< islp_eval_type, expression_node, bool, model::walker >::update (const return_value & __rval) [inline, virtual, inherited]

Definition at line 481 of file evaluator.h.

#### 6.74.3.19 virtual int cached_backward_evaluator_base< islp_eval_type, expression_node, bool, model::walker >::update (const node_data_type & __data, const return_value & __rval) [inline, virtual, inherited]

Definition at line 479 of file evaluator.h.

#### 6.74.3.20 int islp_eval::update (const expression_node & __data, const bool & __rval) [inline]

Definition at line 1542 of file islp_evaluator.h.

#### 6.74.3.21 int islp_eval::update (const bool & __rval) [inline]

Definition at line 1535 of file islp_evaluator.h.

#### 6.74.3.22 return_value cached_backward_evaluator_base< islp_eval_type, expression_node, bool, model::walker >::value () [inline, virtual, inherited]

Reimplemented from _evaluator_base< islp_eval_type, expression_node, bool, model::walker >.

Definition at line 471 of file evaluator.h.

#### 6.74.3.23 int cached_backward_evaluator_base< islp_eval_type, expression_node, bool, model::walker >::vcollect (const return_value & __rval) [inline, virtual, inherited]

Reimplemented from _evaluator_base< islp_eval_type, expression_node, bool, model::walker >.

Definition at line 468 of file evaluator.h.

#### 6.74.3.24 void cached_backward_evaluator_base< islp_eval_type, expression_node, bool, model::walker >::vinit () [inline, inherited]

Definition at line 470 of file evaluator.h.

#### 6.74.3.25 return_value cached_backward_evaluator_base< islp_eval_type, expression_node, bool, model::walker >::vvalue () [inline, virtual, inherited]

Reimplemented from _evaluator_base< islp_eval_type, expression_node, bool, model::walker >.

Definition at line 472 of file evaluator.h.

### 6.74.4 Member Data Documentation

#### 6.74.4.1 islp_eval_type _evaluator_base< islp_eval_type, expression_node, bool, model::walker >::eval_data [protected, inherited]

Definition at line 252 of file evaluator.h.

#### 6.74.4.2 const variable_indicator* cached_evaluator_base< islp_eval_type, expression_node, bool, model::walker >::v_ind [protected, inherited]

Definition at line 295 of file evaluator.h.

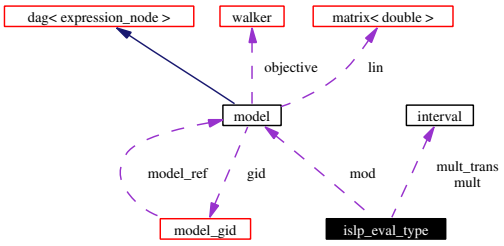
The documentation for this class was generated from the following file:

- islp_evaluator.h

## 6.75 islp_eval_type Struct Reference

`#include <islp_evaluator.h>`

Collaboration diagram for islp_eval_type:




### Public Attributes

- std::vector< std::vector< interval > > * islp_data
- std::vector< std::vector< interval > > * islp_cache
- std::vector< interval > * islp_vec
- const model * mod
- interval mult
- interval mult_trans
- unsigned int child_n

### 6.75.1 Member Data Documentation

#### 6.75.1.1 unsigned int islp_eval_type::child_n

Definition at line 1391 of file islp_evaluator.h.

#### 6.75.1.2 std::vector<std::vector<interval> >* islp_eval_type::islp_cache

Definition at line 1386 of file islp_evaluator.h.

#### 6.75.1.3 std::vector<std::vector<interval> >* islp_eval_type::islp_data

Definition at line 1385 of file islp_evaluator.h.

#### 6.75.1.4 std::vector<interval>* islp_eval_type::islp_vec

Definition at line 1387 of file islp_evaluator.h.

#### 6.75.1.5 const model* islp_eval_type::mod

Definition at line 1388 of file islp_evaluator.h.

#### 6.75.1.6 interval islp_eval_type::mult

Definition at line 1389 of file islp_evaluator.h.

#### 6.75.1.7 interval islp_eval_type::mult_trans

Definition at line 1390 of file islp_evaluator.h.

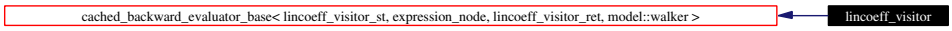
The documentation for this struct was generated from the following file:

- islp_evaluator.h

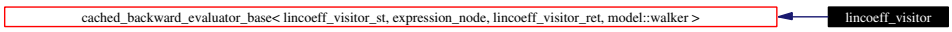
## 6.76 lincoeff_visitor Class Reference

`#include <model-inline.h>`

Inheritance diagram for lincoeff_visitor:




Collaboration diagram for lincoeff_visitor:




### Public Types

- typedef cached_evaluator_base< lincoeff_visitor_st, expression_node, lincoeff_visitor_ret, model::walker >::node_data_type node_data_type
- typedef cached_evaluator_base< lincoeff_visitor_st, expression_node, lincoeff_visitor_ret, model::walker >::return_value return_value
- typedef cached_evaluator_base< lincoeff_visitor_st, expression_node, lincoeff_visitor_ret, model::walker >::walker walker
- typedef lincoeff_visitor_st data_type

### Public Methods

- lincoeff_visitor (const std::vector< interval > *_rg, sparse_vector< double > *_cf, double *_cs, const model *_m)

- lincoeff_visitor (const lincoeff_visitor &__x)
- ~lincoeff_visitor ()
- bool is_cached (const expression_node &__data)
- void initialize ()
- void initialize (const expression_node &__data)
- void retrieve_from_cache (const expression_node &__data)
- int calculate (const expression_node &__data)
- int update (const expression_node &__data, const lincoeff_visitor_ret &__rval)
- int update (const lincoeff_visitor_ret &__rval)
- lincoeff_visitor_ret calculate_value (bool eval_all)
- virtual bool is_cached (const node_data_type &__data)
- int preorder (const node_data_type &__data)
- void postorder (const node_data_type &__data)
- int collect (const node_data_type &__data, const return_value &__rval)
- int vcollect (const return_value &__rval)
- void vinit ()
- return_value value ()
- return_value vvalue ()
- virtual int calculate (const node_data_type &__data)
- virtual void cleanup (const node_data_type &__data)
- virtual void retrieve_from_cache (const node_data_type &__data)
- virtual int update (const node_data_type &__data, const return_value &__rval)
- virtual int update (const return_value &__rval)
- virtual walker short_cut_to (const node_data_type &__data) PURE_VIRTUAL public

**Protected Attributes**

- const variable_indicator * v_ind
- lincoeff_visitor_st eval_data

### 6.76.1 Member Typedef Documentation

#### 6.76.1.1 typedef lincoeff_visitor_st _evaluator_base< lincoeff_visitor_st, expression_node, lincoeff_visitor_ret, model::walker >::data_type [inherited]

Definition at line 245 of file evaluator.h.

#### 6.76.1.2 typedef cached_evaluator_base<lincoeff_visitor_st, expression_node, lincoeff_visitor_ret,model::walker>::node_data_type cached_backward_evaluator_base< lincoeff_visitor_st, expression_node, lincoeff_visitor_ret, model::walker >::node_data_type [inherited]

Reimplemented from cached_evaluator_base< lincoeff_visitor_st, expression_node, lincoeff_visitor_ret, model::walker >.

Definition at line 444 of file evaluator.h.

#### 6.76.1.3 typedef cached_evaluator_base<lincoeff_visitor_st, expression_node, lincoeff_visitor_ret,model::walker>::return_value cached_backward_evaluator_base< lincoeff_visitor_st, expression_node, lincoeff_visitor_ret, model::walker >::return_value [inherited]

Reimplemented from cached_evaluator_base< lincoeff_visitor_st, expression_node, lincoeff_visitor_ret, model::walker >.

Definition at line 446 of file evaluator.h.

#### 6.76.1.4 typedef cached_evaluator_base<lincoeff_visitor_st, expression_node, lincoeff_visitor_ret,model::walker>::walker cached_backward_evaluator_base< lincoeff_visitor_st, expression_node, lincoeff_visitor_ret, model::walker >::walker [inherited]

Reimplemented from cached_evaluator_base< lincoeff_visitor_st, expression_node, lincoeff_visitor_ret, model::walker >.

Definition at line 448 of file evaluator.h.

### 6.76.2 Constructor & Destructor Documentation

#### 6.76.2.1 lincoeff_visitor::lincoeff_visitor (const std::vector< interval > * *_rg*, sparse_vector< double > * *_cf*, double * *_cs*, const model * *_m*) [inline]

Definition at line 1701 of file model-inline.h.

#### 6.76.2.2 lincoeff_visitor::lincoeff_visitor (const lincoeff_visitor & *__x*) [inline]

Definition at line 1720 of file model-inline.h.

#### 6.76.2.3 lincoeff_visitor::~lincoeff_visitor () [inline]

Definition at line 1722 of file model-inline.h.

### 6.76.3 Member Function Documentation

#### 6.76.3.1 virtual int cached_backward_evaluator_base< lincoeff_visitor_st, expression_node, lincoeff_visitor_ret, model::walker >::calculate (const node_data_type & *__data*) [inline, virtual, inherited]

Definition at line 476 of file evaluator.h.

#### 6.76.3.2 int lincoeff_visitor::calculate (const expression_node & *__data*) [inline]

Definition at line 1774 of file model-inline.h.

#### 6.76.3.3 lincoeff_visitor_ret lincoeff_visitor::calculate_value (bool *eval_all*) [inline, virtual]

Reimplemented from cached_backward_evaluator_base< lincoeff_visitor_st, expression_node, lincoeff_visitor_ret, model::walker >.

Definition at line 2362 of file model-inline.h.

#### 6.76.3.4 virtual void cached_backward_evaluator_base< lincoeff_visitor_st, expression_node, lincoeff_visitor_ret, model::walker >::cleanup (const node_data_type & *__data*) [inline, virtual, inherited]

Definition at line 477 of file evaluator.h.

#### 6.76.3.5 int cached_backward_evaluator_base< lincoeff_visitor_st, expression_node, lincoeff_visitor_ret, model::walker >::collect (const node_data_type & *__data*, const return_value & *__rval*) [inline, virtual, inherited]

Reimplemented from _evaluator_base< lincoeff_visitor_st, expression_node, lincoeff_visitor_ret, model::walker >.

Definition at line 466 of file evaluator.h.

#### 6.76.3.6 void lincoeff_visitor::initialize (const expression_node & *__data*) [inline]

Definition at line 1741 of file model-inline.h.

#### 6.76.3.7 void lincoeff_visitor::initialize () [inline, virtual]

Reimplemented from cached_backward_evaluator_base< lincoeff_visitor_st, expression_node, lincoeff_visitor_ret, model::walker >.

Definition at line 1730 of file model-inline.h.

#### 6.76.3.8 virtual bool cached_backward_evaluator_base< lincoeff_visitor_st, expression_node, lincoeff_visitor_ret, model::walker >::is_cached (const node_data_type & *__data*) [inline, virtual, inherited]

Definition at line 453 of file evaluator.h.

#### 6.76.3.9 bool lincoeff_visitor::is_cached (const expression_node & *__data*) [inline]

Definition at line 1724 of file model-inline.h.

#### 6.76.3.10 void cached_backward_evaluator_base< lincoeff_visitor_st, expression_node, lincoeff_visitor_ret, model::walker >::postorder (const node_data_type & *__data*) [inline, virtual, inherited]

Reimplemented from _evaluator_base< lincoeff_visitor_st, expression_node, lincoeff_visitor_ret, model::walker >.

Definition at line 465 of file evaluator.h.

#### 6.76.3.11 int cached_backward_evaluator_base< lincoeff_visitor_st, expression_node, lincoeff_visitor_ret, model::walker >::preorder (const node_data_type & *__data*) [inline, virtual, inherited]

Reimplemented from cached_evaluator_base< lincoeff_visitor_st, expression_node, lincoeff_visitor_ret, model::walker >.

Definition at line 456 of file evaluator.h.

#### 6.76.3.12 virtual void cached_backward_evaluator_base< lincoeff_visitor_st, expression_node, lincoeff_visitor_ret, model::walker >::retrieve_from_cache (const node_data_type & *__data*) [inline, virtual, inherited]

Definition at line 478 of file evaluator.h.

#### 6.76.3.13 void lincoeff_visitor::retrieve_from_cache (const expression_node & *__data*) [inline]

Definition at line 1752 of file model-inline.h.

**6.76.3.14 virtual walker cached_evaluator_base< lincoeff_visitor_st, expression_node, lincoeff_visitor_ret, model::walker >::short_cut_to (const node_data_type & *__data*)** `[inline, virtual, inherited]`

Definition at line 303 of file evaluator.h.

**6.76.3.15 virtual int cached_backward_evaluator_base< lincoeff_visitor_st, expression_node, lincoeff_visitor_ret, model::walker >::update (const return_value & *__rval*)** `[inline, virtual, inherited]`

Definition at line 481 of file evaluator.h.

**6.76.3.16 virtual int cached_backward_evaluator_base< lincoeff_visitor_st, expression_node, lincoeff_visitor_ret, model::walker >::update (const node_data_type & *__data*, const return_value & *__rval*)** `[inline, virtual, inherited]`

Definition at line 479 of file evaluator.h.

**6.76.3.17 int lincoeff_visitor::update (const lincoeff_visitor_ret & *__rval*)** `[inline]`

Definition at line 2354 of file model-inline.h.

**6.76.3.18 int lincoeff_visitor::update (const expression_node & *__data*, const lincoeff_visitor_ret & *__rval*)** `[inline]`

Definition at line 1881 of file model-inline.h.

**6.76.3.19 return_value cached_backward_evaluator_base< lincoeff_visitor_st, expression_node, lincoeff_visitor_ret, model::walker >::value ()** `[inline, virtual, inherited]`

Reimplemented from _evaluator_base< lincoeff_visitor_st, expression_node, lincoeff_visitor_ret, model::walker >.

Definition at line 471 of file evaluator.h.

**6.76.3.20 int cached_backward_evaluator_base< lincoeff_visitor_st, expression_node, lincoeff_visitor_ret, model::walker >::vcollect (const return_value & *__rval*)** `[inline, virtual, inherited]`

Reimplemented from _evaluator_base< lincoeff_visitor_st, expression_node, lincoeff_visitor_ret, model::walker >.

Definition at line 468 of file evaluator.h.

**6.76.3.21 void cached_backward_evaluator_base< lincoeff_visitor_st, expression_node, lincoeff_visitor_ret, model::walker >::vinit ()** `[inline, inherited]`

Definition at line 470 of file evaluator.h.

**6.76.3.22 return_value cached_backward_evaluator_base< lincoeff_visitor_st, expression_node, lincoeff_visitor_ret, model::walker >::vvalue ()** `[inline, virtual, inherited]`

Reimplemented from _evaluator_base< lincoeff_visitor_st, expression_node, lincoeff_visitor_ret, model::walker >.

Definition at line 472 of file evaluator.h.

### 6.76.4 Member Data Documentation

#### 6.76.4.1 lincoeff_visitor_st _evaluator_base< lincoeff_visitor_st, expression_node, lincoeff_visitor_ret, model::walker >::eval_data [protected, inherited]

Definition at line 252 of file evaluator.h.

#### 6.76.4.2 const variable_indicator* cached_evaluator_base< lincoeff_visitor_st, expression_node, lincoeff_visitor_ret, model::walker >::v_ind [protected, inherited]

Definition at line 295 of file evaluator.h.

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

- model-inline.h

## 6.77 lincoeff_visitor_ret Struct Reference

`#include <model-inline.h>`

### Public Attributes

- bool was_lin
- bool was_const
- int node_num
- double const_trans

### 6.77.1 Member Data Documentation

#### 6.77.1.1 double lincoeff_visitor_ret::const_trans

Definition at line 1676 of file model-inline.h.

#### 6.77.1.2 int lincoeff_visitor_ret::node_num

Definition at line 1675 of file model-inline.h.

#### 6.77.1.3 bool lincoeff_visitor_ret::was_const

Definition at line 1674 of file model-inline.h.

#### 6.77.1.4 bool lincoeff_visitor_ret::was_lin

Definition at line 1673 of file model-inline.h.

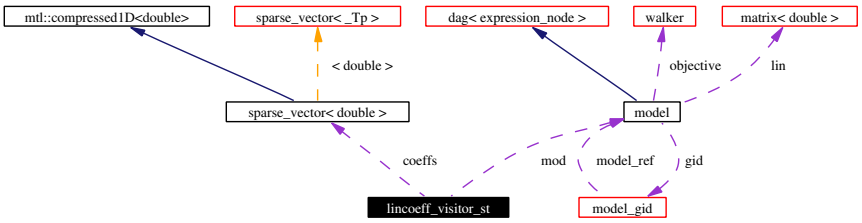
The documentation for this struct was generated from the following file:

- model-inline.h

## 6.78 lincoeff_visitor_st Struct Reference

`#include <model-inline.h>`

Collaboration diagram for lincoeff_visitor_st:




### Public Attributes

- const model * mod
- const std::vector< interval > * ranges
- sparse_vector< double > * coeffs
- double * constant
- double old_trans
- double trans
- double mm_help
- lincoeff_visitor_ret r
- int old_nlin
- int in_nlin
- unsigned int n

### 6.78.1 Member Data Documentation

#### 6.78.1.1 sparse_vector<double>* lincoeff_visitor_st::coeffs

Definition at line 1683 of file model-inline.h.

#### 6.78.1.2 double* lincoeff_visitor_st::constant

Definition at line 1685 of file model-inline.h.

#### 6.78.1.3 int lincoeff_visitor_st::in_nlin

Definition at line 1688 of file model-inline.h.

#### 6.78.1.4 double lincoeff_visitor_st::mm_help

Definition at line 1686 of file model-inline.h.

#### 6.78.1.5 const model* lincoeff_visitor_st::mod

Definition at line 1681 of file model-inline.h.

#### 6.78.1.6 unsigned int lincoeff_visitor_st::n

Definition at line 1689 of file model-inline.h.

#### 6.78.1.7 int lincoeff_visitor_st::old_nlin

Definition at line 1688 of file model-inline.h.

#### 6.78.1.8 double lincoeff_visitor_st::old_trans

Definition at line 1686 of file model-inline.h.

#### 6.78.1.9 lincoeff_visitor_ret lincoeff_visitor_st::r

Definition at line 1687 of file model-inline.h.

#### 6.78.1.10 const std::vector<interval>* lincoeff_visitor_st::ranges

Definition at line 1682 of file model-inline.h.

#### 6.78.1.11 double lincoeff_visitor_st::trans

Definition at line 1686 of file model-inline.h.

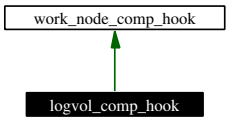
The documentation for this struct was generated from the following file:

- model-inline.h

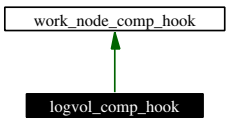
## 6.79 logvol_comp_hook Class Reference

`#include <logvol_hook.h>`

Inheritance diagram for logvol_comp_hook:




Collaboration diagram for logvol_comp_hook:




### Public Methods

- logvol_comp_hook ()

- void operator() (const work_node &wn, dbt_row &dbr)
- bool init_columns (vdbl::standard_table &stable)
- bool drop_columns (vdbl::standard_table &stable)

**Protected Methods**

- template<class _CI> bool _init_column (vdbl::standard_table &stable, const std::string &colname, const _CI &c)
- template<class _CI> bool _init_column (vdbl::standard_table &stable, const char *colname, const _CI &c)
- bool _drop_columns (vdbl::standard_table &stable)
- search_node_relation parent_relation (const work_node &wn) const
- search_node_id node_id (const work_node &wn) const
- const std::string & name ()

### 6.79.1 Constructor & Destructor Documentation

#### 6.79.1.1 logvol_comp_hook::logvol_comp_hook () [inline]

Definition at line 35 of file logvol_hook.h.

### 6.79.2 Member Function Documentation

#### 6.79.2.1 bool work_node_comp_hook::_drop_columns (vdbl::standard_table & *stable*) [protected, inherited]

Definition at line 42 of file comp_hook.cc.

#### 6.79.2.2 template<class _CI> bool work_node_comp_hook::_init_column (vdbl::standard_table & *stable*, const char * *colname*, const _CI & *c*) [inline, protected, inherited]

Definition at line 45 of file comp_hook.h.

#### 6.79.2.3 template<class _CI> bool work_node_comp_hook::_init_column (vdbl::standard_table & *stable*, const std::string & *colname*, const _CI & *c*) [protected, inherited]

Definition at line 31 of file comp_hook.cc.

#### 6.79.2.4 bool logvol_comp_hook::drop_columns (vdbl::standard_table & *stable*) [inline, virtual]

Reimplemented from work_node_comp_hook.

Definition at line 47 of file logvol_hook.h.

#### 6.79.2.5 bool logvol_comp_hook::init_columns (vdbl::standard_table & *stable*) [inline, virtual]

Reimplemented from work_node_comp_hook.

Definition at line 44 of file logvol_hook.h.

#### 6.79.2.6 const std::string& work_node_comp_hook::name () [inline, inherited]

Definition at line 68 of file comp_hook.h.

#### 6.79.2.7 search_node_id work_node_comp_hook::node_id (const work_node & *wn*) const [protected, inherited]

Definition at line 51 of file comp_hook.cc.

#### 6.79.2.8 void logvol_comp_hook::operator() (const work_node & *wn*, dbt_row & *dbr*) [inline, virtual]

Implements work_node_comp_hook.

Definition at line 39 of file logvol_hook.h.

#### 6.79.2.9 search_node_relation work_node_comp_hook::parent_relation (const work_node & *wn*) const [protected, inherited]

Definition at line 54 of file comp_hook.cc.

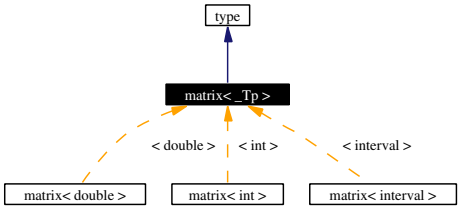
The documentation for this class was generated from the following file:

- logvol_hook.h

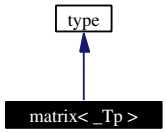
## 6.80 matrix< _Tp > Class Template Reference

`#include <linalg.h>`

Inheritance diagram for matrix< _Tp >:




Collaboration diagram for matrix< _Tp >:




### Public Types

- typedef _Base::OneD Row

**Public Methods**

- matrix ()
- matrix (const _Self &__m)
- matrix (const size_t &n, const size_t &m)

**template<class _Tp> class matrix< _Tp >**

### 6.80.1 Member Typedef Documentation

#### 6.80.1.1 template<class _Tp> typedef _Base::OneD matrix< _Tp >::Row

Definition at line 18 of file linalg.h.

### 6.80.2 Constructor & Destructor Documentation

#### 6.80.2.1 template<class _Tp> matrix< _Tp >::matrix () [inline]

Definition at line 20 of file linalg.h.

#### 6.80.2.2 template<class _Tp> matrix< _Tp >::matrix (const _Self & *__m*) [inline]

Definition at line 21 of file linalg.h.

#### 6.80.2.3 template<class _Tp> matrix< _Tp >::matrix (const size_t & *n*, const size_t & *m*) [inline]

Definition at line 22 of file linalg.h.

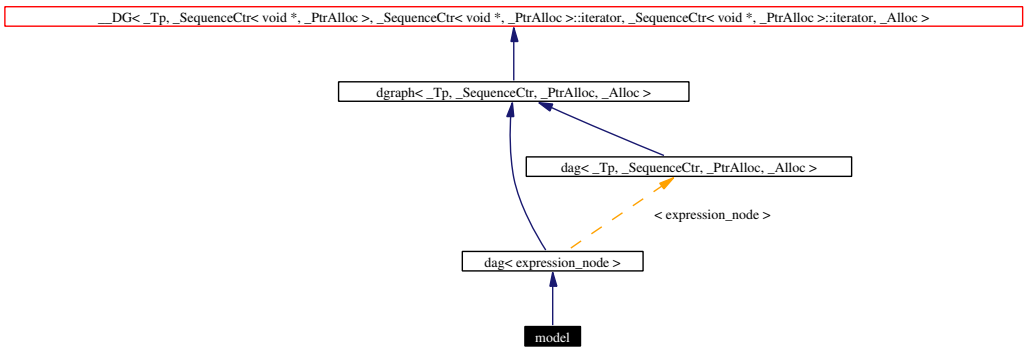
The documentation for this class was generated from the following file:

- linalg.h

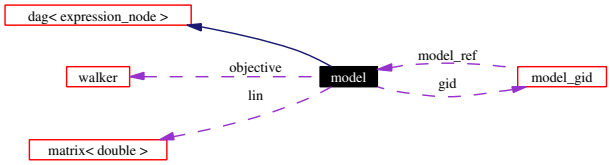
## 6.81 model Class Reference

`#include <model.h>`

Inheritance diagram for model:



Collaboration diagram for model:




### Public Types

- typedef **dag**< expression_node >::walker walker
- typedef **dag**< expression_node >::const_walker const_walker
- typedef std::vector< walker >::**iterator** ref_iterator
- typedef std::vector< walker >::**const_iterator** const_ref_iterator

### Public Methods

- model (model_gid *__id=NULL, bool clone=false)
- model (model_gid *__id, const **erased_part** &_ep, bool clone=false)
- model (int __num_of_vars)
- model (const model &__m)
- model (model_gid *__id, const model &__m)
- model (const char *name, bool do_simplify=true)
- ~model ()
- int next_num ()
- int next_variable_num ()
- int next_constraint_num ()
- unsigned int number_of_variables () const
- unsigned int number_of_nodes () const
- unsigned int number_of_constraints () const
- unsigned int number_of_managed_nodes () const
- unsigned int number_of_managed_variables () const
- unsigned int number_of_managed_constraints () const
- const walker & var (unsigned int i) const
- const walker & node (unsigned int i) const
- const walker & constraint (unsigned int i) const
- model_gid * gid_data () const
- void detach_gid ()
- void compress_numbers ()
- void renumber_variables ()
- void renumber_constraints ()
- bool basic_simplify ()
- void arrange_constraints ()
- void detect_0chain ()
- bool simplify_thin ()
- void set_counters ()
- void clr_sky_ground_link ()
- void write (std::ostream &__o=std::cout) const

- ref_iterator ghost_begin ()
- const_ref_iterator ghost_begin () const
- ref_iterator ghost_end ()
- const_ref_iterator ghost_end () const
- walker store_node (const walker &_w)
- walker store_variable (const walker &_w)
- walker store_ghost (const walker &_w)
- walker store_constraint (const walker &_w)
- void free_node_num (unsigned int _nnum)
- void remove_node (const walker &_w, unsigned int _nnum)
- void remove_node (const walker &_w)
- void remove_node (unsigned int __node_num)
- void new_variables (int _new_num_of_vars)
- walker ghost (unsigned int _nnum)
- walker constant (double _constant)
- walker constant (const std::vector< double > &_constant)
- walker variable (unsigned int _vnum)
- walker binary (const walker &_op1, const walker &_op2, int expr_type, double _coeff1=1.0, double _coeff2=1.0)
- walker binary (const walker &_op1, const walker &_op2, int expr_type, additional_info_u _params, double _coeff1=1.0, double _coeff2=1.0)
- walker unary (const walker &_op1, int expr_type, double _coeff=1.0)
- walker unary (const walker &_op1, int expr_type, additional_info_u _params, double _coeff=1.0)
- walker nary (const std::vector< walker > &_op, int expr_type, const std::vector< double > &_coeffs=std::vector< double >())
- walker nary (const std::vector< walker > &_op, int expr_type, additional_info_u _params, const std::vector< double > &_coeffs=std::vector< double >())
- walker vnary (int expr_type,...)
- bool is_empty (const walker &_w) const
- walker empty_reference () const
- const std::string var_name (unsigned int n) const
- const std::string const_name (unsigned int n) const
- const std::string obj_name () const
- double obj_adj () const
- double obj_mult () const
- size_t n_fixed_vars () const
- std::pair< const std::string, double > fixed_var (unsigned int n) const
- size_t n_unused_vars () const
- const std::string & unused_var (unsigned int n) const
- size_t n_unused_constrs () const
- const std::string & unused_constr (unsigned int n) const
- bool get_const_num (unsigned int node_num, unsigned int &const_num) const
- bool get_linear_coeffs (const walker &expr, sparse_vector< double > &coeffs, double &constant, const std::vector< interval > &_ranges)
- bool get_linear_coeffs (const walker &expr, sparse_vector< double > &coeffs, double &constant)
- bool check_acyclicity (const walker &__parent, const walker &__child)
- void clear ()
- walker between (const walker &__parent, const **children_iterator** &__cit, const walker &__child, const **parents_iterator** &__pit, const expression_node &__x)
- walker between (const __SequenceCtr1< walker, _Allocator1 > &__parents, const __SequenceCtr2< walker, _Allocator2 > &__children, const expression_node &__x)

- walker between (const walker &__parent, const **children_iterator** &__cit, const __SequenceCtr< walker, _Allocator > &__children, const expression_node &__x)
- walker between (const __SequenceCtr< walker, _Allocator > &__parents, const walker &__child, const **parents_iterator** &__pit, const expression_node &__x)
- walker **between** (const walker &__parent, const **children_iterator** &__cit, const walker &__child, const **parents_iterator** &__pit, const _Tp &__x)
- walker **between** (const __SequenceCtr1< walker, _Allocator1 > &__parents, const __SequenceCtr2< walker, _Allocator2 > &__children, const _Tp &__x)
- walker **between** (const walker &__parent, const **children_iterator** &__cit, const __SequenceCtr< walker, _Allocator > &__children, const _Tp &__x)
- walker **between** (const __SequenceCtr< walker, _Allocator > &__parents, const walker &__child, const **parents_iterator** &__pit, const _Tp &__x)
- walker split (const walker &__parent, const **children_iterator** &__ch_it, const walker &__child, const **parents_iterator** &__pa_it, const expression_node &__x)
- void split (const __SequenceCtr1< walker, _Allocator1 > &__parents, const __SequenceCtr2< walker, _Allocator2 > &__children, const expression_node &__x)
- walker split (const walker &__parent, const **children_iterator** &__ch_it, const __SequenceCtr< walker, _Allocator > &__children, const expression_node &__x)
- walker split (const __SequenceCtr< walker, _Allocator > &__parents, const walker &__child, const **parents_iterator** &__pr_it, const expression_node &__x)
- walker **split** (const walker &__parent, const **children_iterator** &__ch_it, const walker &__child, const **parents_iterator** &__pa_it, const _Tp &__x)
- void **split** (const __SequenceCtr1< walker, _Allocator1 > &__parents, const __SequenceCtr2< walker, _Allocator2 > &__children, const _Tp &__x)
- walker **split** (const walker &__parent, const **children_iterator** &__ch_it, const __SequenceCtr< walker, _Allocator > &__children, const _Tp &__x)
- walker **split** (const __SequenceCtr< walker, _Allocator > &__parents, const walker &__child, const **parents_iterator** &__pr_it, const _Tp &__x)
- walker between_back (const walker &__parent, const walker &__child, const expression_node &__x)
- walker between_back (const walker &__parent, const __SequenceCtr< walker, _Allocator > &__children, const expression_node &__x)
- walker between_back (const __SequenceCtr< walker, _Allocator > &__parents, const walker &__child, const expression_node &__x)
- walker **between_back** (const walker &__parent, const walker &__child, const _Tp &__x)
- walker **between_back** (const walker &__parent, const __SequenceCtr< walker, _Allocator > &__children, const _Tp &__x)
- walker **between_back** (const __SequenceCtr< walker, _Allocator > &__parents, const walker &__child, const _Tp &__x)
- walker split_back (const walker &__parent, const walker &__child, const expression_node &__x)
- walker split_back (const walker &__parent, const __SequenceCtr< walker, _Allocator > &__children, const expression_node &__x)
- walker split_back (const __SequenceCtr< walker, _Allocator > &__parents, const walker &__child, const expression_node &__x)
- walker **split_back** (const walker &__parent, const walker &__child, const _Tp &__x)
- walker **split_back** (const walker &__parent, const __SequenceCtr< walker, _Allocator > &__children, const _Tp &__x)
- walker **split_back** (const __SequenceCtr< walker, _Allocator > &__parents, const walker &__child, const _Tp &__x)
- walker between_front (const walker &__parent, const walker &__child, const expression_node &__x)
- walker between_front (const walker &__parent, const __SequenceCtr< walker, _Allocator > &__children, const expression_node &__x)

- walker between_front (const __SequenceCtr< walker, _Allocator > &__parents, const walker &__child, const expression_node &__x)
- walker **between_front** (const walker &__parent, const walker &__child, const _Tp &__x)
- walker **between_front** (const walker &__parent, const __SequenceCtr< walker, _Allocator > &__children, const _Tp &__x)
- walker **between_front** (const __SequenceCtr< walker, _Allocator > &__parents, const walker &__child, const _Tp &__x)
- walker split_front (const walker &__parent, const walker &__child, const expression_node &__x)
- walker split_front (const walker &__parent, const __SequenceCtr< walker, _Allocator > &__children, const expression_node &__x)
- walker split_front (const __SequenceCtr< walker, _Allocator > &__parents, const walker &__child, const expression_node &__x)
- walker **split_front** (const walker &__parent, const walker &__child, const _Tp &__x)
- walker **split_front** (const walker &__parent, const __SequenceCtr< walker, _Allocator > &__children, const _Tp &__x)
- walker **split_front** (const __SequenceCtr< walker, _Allocator > &__parents, const walker &__child, const _Tp &__x)
- void insert_subgraph (_Self &__subgraph, const walker &__parent, const **children_iterator** &__ch_it, const walker &__child, const **parents_iterator** &__pa_it)
- void insert_subgraph (_Self &__subgraph, const walker &__parent, const walker &__child, const container_insert_arg &__Itc, const container_insert_arg &__Itp)
- void insert_subgraph (_Self &__subgraph, const __SequenceCtr1< walker, _Allocator1 > &__parents, const __SequenceCtr2< walker, _Allocator2 > &__children)
- void insert_back_subgraph (_Self &__subgraph, const walker &__parent, const walker &__child)
- void insert_front_subgraph (_Self &__subgraph, const walker &__parent, const walker &__child)
- void add_edge (const walker &__parent, const **children_iterator** &__ch_it, const walker &__child, const **parents_iterator** &__pa_it)
- void add_edge (const **edge** &__edge, const container_insert_arg &__Itc, const container_insert_arg &__Itp)
- void add_edge (const walker &__parent, const walker &__child, const container_insert_arg &__Itc, const container_insert_arg &__Itp)
- void add_edge_back (const walker &__parent, const walker &__child)
- void add_edge_front (const walker &__parent, const walker &__child)
- **allocator_type** get_allocator () const
- walker ground ()
- const_walker ground () const
- walker sky ()
- const_walker sky () const
- **children_iterator** root_begin ()
- **children_iterator** root_end ()
- **parents_iterator** leaf_begin ()
- **parents_iterator** leaf_end ()
- bool empty () const
- **size_type** size () const
- **size_type** max_size () const
- void swap (_Self &__x)
- walker insert_node_in_graph (_Node *__n, const walker &__parent, const walker &__child, const container_insert_arg &__Itc, const container_insert_arg &__Itp)
- walker insert_node_in_graph (_Node *__node, const __SequenceCtr1< walker, _Allocator1 > &__parents, const __SequenceCtr2< walker, _Allocator2 > &__children)

- walker insert_node_in_graph (_Node *__node, const walker &__parent, const container_insert_arg &__pref, const __SequenceCtr< walker, _Allocator > &__children)
- walker insert_node_in_graph (_Node *__node, const __SequenceCtr< walker, _Allocator > &__parents, const walker &__child, const container_insert_arg &__cref)
- walker insert_in_graph (const expression_node &__x, const walker &__parent, const walker &__child, const container_insert_arg &__Itc, const container_insert_arg &__Itp)
- walker insert_in_graph (const walker &__parent, const walker &__child, const container_insert_arg &__Itc, const container_insert_arg &__Itp)
- walker insert_in_graph (const expression_node &__x, const __SequenceCtr1< walker, _Allocator1 > &__parents, const __SequenceCtr2< walker, _Allocator2 > &__children)
- walker insert_in_graph (const __SequenceCtr1< walker, _Allocator1 > &__parents, const __SequenceCtr2< walker, _Allocator2 > &__children)
- walker insert_in_graph (const expression_node &__x, const walker &__parent, const container_insert_arg &__pref, const __SequenceCtr< walker, _Allocator > &__children)
- walker insert_in_graph (const walker &__parent, const container_insert_arg &__pref, const __SequenceCtr< walker, _Allocator > &__children)
- walker insert_in_graph (const expression_node &__x, const __SequenceCtr< walker, _Allocator > &__parents, const walker &__child, const container_insert_arg &__cref)
- walker insert_in_graph (const __SequenceCtr< walker, _Allocator > &__parents, const walker &__child, const container_insert_arg &__cref)
- walker **insert_in_graph** (const _Tp &__x, const walker &__parent, const walker &__child, const container_insert_arg &__Itc, const container_insert_arg &__Itp)
- walker **insert_in_graph** (const _Tp &__x, const __SequenceCtr1< walker, _Allocator1 > &__parents, const __SequenceCtr2< walker, _Allocator2 > &__children)
- walker **insert_in_graph** (const _Tp &__x, const walker &__parent, const container_insert_arg &__pref, const __SequenceCtr< walker, _Allocator > &__children)
- walker **insert_in_graph** (const _Tp &__x, const __SequenceCtr< walker, _Allocator > &__parents, const walker &__child, const container_insert_arg &__cref)
- void replace_edge_to_child (const walker &__parent, const walker &__child_old, const walker &__child_new)
- void replace_edge_to_parent (const walker &__parent_old, const walker &__parent_new, const walker &__child)
- void remove_edge (const **edge** &__edge)
- void remove_edge (const walker &__parent, const walker &__child)
- void remove_edge_and_deattach (const walker &__parent, const walker &__child)
- void sort_child_edges (walker __position, **children_iterator** first, **children_iterator** last, Compare comp)
- void sort_child_edges (walker __position, Compare comp)
- void sort_parent_edges (walker __position, **parents_iterator** first, **parents_iterator** last, Compare comp)
- void sort_parent_edges (walker __position, Compare comp)
- walker insert_node (_Node *_node, const walker &__position, const container_insert_arg &__It)
- walker insert_node (const expression_node &__x, const walker &__position, const container_insert_arg &__It)
- walker insert_node (const walker &__position, const container_insert_arg &__It)
- walker **insert_node** (const _Tp &__x, const walker &__position, const container_insert_arg &__It)
- walker insert_node_before (_Node *_node, const walker &__position, const container_insert_arg &__It)
- void insert_node_before (const expression_node &__x, const walker &__position, const container_insert_arg &__It)
- void insert_node_before (const walker &__position, const container_insert_arg &__It)

- void **insert_node_before** (const _Tp &__x, const walker &__position, const container_insert_arg &__-It)
- void merge (const walker &__position, const walker &__second, bool merge_parent_edges=true, bool merge_child_edges=true)
- void erase (const walker &__position)
- void clear_erased_part (**erased_part** &_ep)
- **erased_part** erase_maximal_subgraph (const walker &__position)
- **erased_part** erase_maximal_subgraph (const __SequenceCtr< walker, _Allocator > &__positions)
- **erased_part** erase_minimal_subgraph (const walker &__position)
- **erased_part** erase_minimal_subgraph (const __SequenceCtr< walker, _Allocator > &__positions)
- **erased_part** erase_maximal_pregraph (const walker &__position)
- **erased_part** erase_maximal_pregraph (const __SequenceCtr< walker, _Allocator > &__positions)
- **erased_part** erase_minimal_pregraph (const walker &__position)
- **erased_part** erase_minimal_pregraph (const __SequenceCtr< walker, _Allocator > &__positions)
- bool erase_child (const walker &__position, const **children_iterator** &__It)
- bool erase_parent (const walker &__position, const **parents_iterator** &__It)
- void clear_children ()
- void clear_parents ()
- void add_all_children (_Output_Iterator fi, _**DG_node**< expression_node, _SequenceCtr< void *, _-PtrAlloc >, _SequenceCtr< void *, _PtrAlloc >::**iterator** > *_parent)
- void **add_all_children** (_Output_Iterator fi, _**DG_node**< _Tp, _SequenceCtr< void *, _PtrAlloc >, _SequenceCtr< void *, _PtrAlloc >::**iterator** > *_parent)
- void **add_all_children** (_Output_Iterator fi, _**DG_node**< _Tp, _Ctr, _Iterator > *_parent)
- void add_all_parents (_Output_Iterator fi, _**DG_node**< expression_node, _SequenceCtr< void *, _Ptr-Alloc >, _SequenceCtr< void *, _PtrAlloc >::**iterator** > *_child)
- void **add_all_parents** (_Output_Iterator fi, _**DG_node**< _Tp, _SequenceCtr< void *, _PtrAlloc >, _SequenceCtr< void *, _PtrAlloc >::**iterator** > *_child)
- void **add_all_parents** (_Output_Iterator fi, _**DG_node**< _Tp, _Ctr, _Iterator > *_child)
- _Node * _C_create_node (const expression_node &__x)
- _Node * _C_create_node ()
- _Node * _**C_create_node** (const _Tp &__x)
- void clear_graph (_**DG_node**< expression_node, _SequenceCtr< void *, _PtrAlloc >, _Sequence-Ctr< void *, _PtrAlloc >::**iterator** > *_node)
- void **clear_graph** (_**DG_node**< _Tp, _SequenceCtr< void *, _PtrAlloc >, _SequenceCtr< void *, _PtrAlloc >::**iterator** > *_node)
- void **clear_graph** (_**DG_node**< _Tp, _Ctr, _Iterator > *_node)
- _**DG_node**< expression_node, _SequenceCtr< void *, _PtrAlloc >, _SequenceCtr< void *, _PtrAlloc >::**iterator** > * _C_get_node ()
- void _C_put_node (_**DG_node**< expression_node, _SequenceCtr< void *, _PtrAlloc >, _Sequence-Ctr< void *, _PtrAlloc >::**iterator** > *__p)
- void _**C_put_node** (_**DG_node**< _Tp, _SequenceCtr< void *, _PtrAlloc >, _SequenceCtr< void *, _PtrAlloc >::**iterator** > *__p)
- void _**C_put_node** (_**DG_node**< _Tp, _Ctr, _Iterator > *__p)

### Public Attributes

- matrix< double > lin
- std::vector< matrix< double > > matd
- std::vector< matrix< interval > > mati
- int ocoeff
- walker objective

- std::vector< walker > constraints
- _Base::children_iterator children_iterator
- _Base::parents_iterator parents_iterator
- _Base::erased_part erased_part
- _SequenceCtr< void *, _PtrAlloc > container_type
- expression_node value_type
- **_DG_iterator**< expression_node, expression_node &, expression_node *, **container_type**, **children_iterator** > iterator
- **_DG_iterator**< expression_node, const expression_node &, const expression_node *, **container_type**, **children_iterator** > const_iterator
- std::reverse_iterator< **const_iterator** > const_reverse_iterator
- std::reverse_iterator< **iterator** > reverse_iterator
- std::pair< walker, walker > edge
- std::pair< **edge**, bool > enhanced_edge
- _Base::allocator_type allocator_type
- **_DG_node**< expression_node, _SequenceCtr< void *, _PtrAlloc >, _SequenceCtr< void *, _PtrAlloc >::**iterator** > * _C_ground
- **_DG_node**< expression_node, _SequenceCtr< void *, _PtrAlloc >, _SequenceCtr< void *, _PtrAlloc >::**iterator** > * _C_sky
- int _C_mark
- _Node node_type
- **value_type** * pointer
- const **value_type** * const_pointer
- **value_type** & reference
- const **value_type** & const_reference
- size_t size_type
- ptrdiff_t difference_type

**Friends**

- class dag_delta
- class dag_undelta
- bool operator==__VGTL_NULL_TMPL_ARGS (const __**DG** &__x, const __**DG** &__y)

### 6.81.1 Member Typedef Documentation

#### 6.81.1.1 typedef std::vector<walker>::const_iterator model::const_ref_iterator

Definition at line 48 of file model.h.

#### 6.81.1.2 typedef dag<expression_node>::const_walker model::const_walker

Reimplemented from **dag**< **expression_node** >.

Definition at line 45 of file model.h.

#### 6.81.1.3 typedef std::vector<walker>::iterator model::ref_iterator

Definition at line 47 of file model.h.

#### 6.81.1.4 typedef dag<expression_node>::walker model::walker

Reimplemented from **dag< expression_node >**.

Definition at line 44 of file model.h.

### 6.81.2 Constructor & Destructor Documentation

#### 6.81.2.1 model::model (model_gid * __id = NULL, bool clone = false) [inline]

Definition at line 420 of file model-inline.h.

#### 6.81.2.2 model::model (model_gid * __id, const erased_part & _ep, bool clone = false)

Definition at line 176 of file model.cc.

#### 6.81.2.3 model::model (int __num_of_vars) [inline]

Definition at line 414 of file model-inline.h.

#### 6.81.2.4 model::model (const model & __m) [inline]

Definition at line 441 of file model-inline.h.

#### 6.81.2.5 model::model (model_gid * __id, const model & __m) [inline]

Definition at line 455 of file model-inline.h.

#### 6.81.2.6 model::model (const char * name, bool do_simplify = true)

Definition at line 304 of file model.cc.

#### 6.81.2.7 model::~model () [inline]

Definition at line 470 of file model-inline.h.

### 6.81.3 Member Function Documentation

#### 6.81.3.1 _Node * dag< expression_node, _SequenceCtr, _PtrAlloc, _Alloc >::_C_create_node () [inherited]

Reimplemented from **dgraph< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >**.

#### 6.81.3.2 _Node * dag< expression_node, _SequenceCtr, _PtrAlloc, _Alloc >::_C_create_node (const expression_node & __x) [inherited]

#### 6.81.3.3 _DG_node< expression_node, _SequenceCtr< void *, _PtrAlloc >, _SequenceCtr< void *, _PtrAlloc >::iterator > * dag< expression_node, _SequenceCtr, _PtrAlloc, _Alloc >::_C_get_node () [inherited]

Reimplemented from **dgraph< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >**.

**6.81.3.4 void dag< expression_node, _SequenceCtr, _PtrAlloc, _Alloc >::_C_put_node (_DG_node< expression_node, _SequenceCtr< void *, _PtrAlloc >, _SequenceCtr< void *, _PtrAlloc >::iterator > * __p)** [inherited]

**6.81.3.5 void dag< expression_node, _SequenceCtr, _PtrAlloc, _Alloc >::add_all_children (_Output_Iterator *fi*, _DG_node< expression_node, _SequenceCtr< void *, _PtrAlloc >, _SequenceCtr< void *, _PtrAlloc >::iterator > * *_parent*)** [inherited]

**6.81.3.6 void dag< expression_node, _SequenceCtr, _PtrAlloc, _Alloc >::add_all_parents (_Output_Iterator *fi*, _DG_node< expression_node, _SequenceCtr< void *, _PtrAlloc >, _SequenceCtr< void *, _PtrAlloc >::iterator > * *_child*)** [inherited]

**6.81.3.7 void dag< expression_node, _SequenceCtr, _PtrAlloc, _Alloc >::add_edge (const walker & *__parent*, const walker & *__child*, const container_insert_arg & *__Itc*, const container_insert_arg & *__Itp*)** [inherited]

Reimplemented from **dgraph< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >**.

**6.81.3.8 void dag< expression_node, _SequenceCtr, _PtrAlloc, _Alloc >::add_edge (const edge & *__edge*, const container_insert_arg & *__Itc*, const container_insert_arg & *__Itp*)** [inherited]

Reimplemented from **dgraph< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >**.

**6.81.3.9 void dag< expression_node, _SequenceCtr, _PtrAlloc, _Alloc >::add_edge (const walker & *__parent*, const children_iterator & *__ch_it*, const walker & *__child*, const parents_iterator & *__pa_it*)** [inherited]

Reimplemented from **dgraph< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >**.

**6.81.3.10 void dag< expression_node, _SequenceCtr, _PtrAlloc, _Alloc >::add_edge_back (const walker & *__parent*, const walker & *__child*)** [inherited]

Reimplemented from **dgraph< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >**.

**6.81.3.11 void dag< expression_node, _SequenceCtr, _PtrAlloc, _Alloc >::add_edge_front (const walker & *__parent*, const walker & *__child*)** [inherited]

Reimplemented from **dgraph< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >**.

**6.81.3.12 void model::arrange_constraints ()** [inline]

Definition at line 1205 of file model-inline.h.

**6.81.3.13 bool model::basic_simplify ()**

Definition at line 2447 of file model.cc.

**6.81.3.14 walker dag< expression_node, _SequenceCtr, _PtrAlloc, _Alloc >::between (const __SequenceCtr< walker, _Allocator > & *__parents*, const walker & *__child*, const parents_iterator & *__pit*, const expression_node & *__x*)** [inherited]

**6.81.3.15 walker dag< expression_node, _SequenceCtr, _PtrAlloc, _Alloc >::between (const walker & __parent, const children_iterator & __cit, const __SequenceCtr< walker, _Allocator > & __children, const expression_node & __x)** [inherited]

**6.81.3.16 walker dag< expression_node, _SequenceCtr, _PtrAlloc, _Alloc >::between (const __SequenceCtr1< walker, _Allocator1 > & __parents, const __SequenceCtr2< walker, _Allocator2 > & __children, const expression_node & __x)** [inherited]

**6.81.3.17 walker dag< expression_node, _SequenceCtr, _PtrAlloc, _Alloc >::between (const walker & __parent, const children_iterator & __cit, const walker & __child, const parents_iterator & __pit, const expression_node & __x)** [inherited]

**6.81.3.18 walker dag< expression_node, _SequenceCtr, _PtrAlloc, _Alloc >::between_back (const __SequenceCtr< walker, _Allocator > & __parents, const walker & __child, const expression_node & __x)** [inherited]

**6.81.3.19 walker dag< expression_node, _SequenceCtr, _PtrAlloc, _Alloc >::between_back (const walker & __parent, const __SequenceCtr< walker, _Allocator > & __children, const expression_node & __x)** [inherited]

**6.81.3.20 walker dag< expression_node, _SequenceCtr, _PtrAlloc, _Alloc >::between_back (const walker & __parent, const walker & __child, const expression_node & __x)** [inherited]

**6.81.3.21 walker dag< expression_node, _SequenceCtr, _PtrAlloc, _Alloc >::between_front (const __SequenceCtr< walker, _Allocator > & __parents, const walker & __child, const expression_node & __x)** [inherited]

**6.81.3.22 walker dag< expression_node, _SequenceCtr, _PtrAlloc, _Alloc >::between_front (const walker & __parent, const __SequenceCtr< walker, _Allocator > & __children, const expression_node & __x)** [inherited]

**6.81.3.23 walker dag< expression_node, _SequenceCtr, _PtrAlloc, _Alloc >::between_front (const walker & __parent, const walker & __child, const expression_node & __x)** [inherited]

**6.81.3.24 model::walker model::binary (const walker & _op1, const walker & _op2, int expr_type, additional_info_u _params, double _coeff1 = 1.0, double _coeff2 = 1.0)** [inline]

Definition at line 1461 of file model-inline.h.

**6.81.3.25 model::walker model::binary (const walker & _op1, const walker & _op2, int expr_type, double _coeff1 = 1.0, double _coeff2 = 1.0)** [inline]

Definition at line 1441 of file model-inline.h.

**6.81.3.26 bool dag< expression_node, _SequenceCtr, _PtrAlloc, _Alloc >::check_acyclicity (const walker & __parent, const walker & __child)** [inherited]

#### 6.81.3.27 void dag< expression_node, _SequenceCtr, _PtrAlloc, _Alloc >::clear () [inherited]

Reimplemented from **dgraph**< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >.

#### 6.81.3.28 void dag< expression_node, _SequenceCtr, _PtrAlloc, _Alloc >::clear_children () [inherited]

Reimplemented from **dgraph**< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >.

#### 6.81.3.29 void dag< expression_node, _SequenceCtr, _PtrAlloc, _Alloc >::clear_erased_part (erased_part & *_ep*) [inherited]

Reimplemented from **dgraph**< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >.

#### 6.81.3.30 void dag< expression_node, _SequenceCtr, _PtrAlloc, _Alloc >::clear_graph (_DG_node< expression_node, _SequenceCtr< void *, _PtrAlloc >, _SequenceCtr< void *, _PtrAlloc >::iterator > * *_node*) [inherited]

#### 6.81.3.31 void dag< expression_node, _SequenceCtr, _PtrAlloc, _Alloc >::clear_parents () [inherited]

Reimplemented from **dgraph**< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >.

#### 6.81.3.32 void model::clr_sky_ground_link () [inline]

Definition at line 1539 of file model-inline.h.

#### 6.81.3.33 void model::compress_numbers () [inline]

Definition at line 1220 of file model-inline.h.

#### 6.81.3.34 const std::string model::const_name (unsigned int *n*) const [inline]

Definition at line 487 of file model-inline.h.

#### 6.81.3.35 model::walker model::constant (const std::vector< double > & *_constant*) [inline]

Definition at line 1410 of file model-inline.h.

#### 6.81.3.36 model::walker model::constant (double *_constant*) [inline]

Definition at line 1399 of file model-inline.h.

#### 6.81.3.37 const model::walker & model::constraint (unsigned int *i*) const [inline]

Definition at line 518 of file model-inline.h.

#### 6.81.3.38 void model::detach_gid () [inline]

Definition at line 412 of file model-inline.h.

#### 6.81.3.39 void model::detect_0chain ()

Definition at line 2618 of file model.cc.

#### 6.81.3.40 bool dag< expression_node, _SequenceCtr, _PtrAlloc, _Alloc >::empty () [inherited]

Reimplemented from **dgraph< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >**.

#### 6.81.3.41 model::walker model::empty_reference () [inline]

Definition at line 409 of file model-inline.h.

#### 6.81.3.42 void dag< expression_node, _SequenceCtr, _PtrAlloc, _Alloc >::erase (const walker & __position) [inherited]

Reimplemented from **dgraph< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >**.

#### 6.81.3.43 bool dag< expression_node, _SequenceCtr, _PtrAlloc, _Alloc >::erase_child (const walker & __position, const children_iterator & __It) [inherited]

Reimplemented from **dgraph< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >**.

#### 6.81.3.44 erased_part dag< expression_node, _SequenceCtr, _PtrAlloc, _Alloc >::erase_maximal_pregraph (const __SequenceCtr< walker, _Allocator > & __positions) [inherited]

Reimplemented from **dgraph< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >**.

#### 6.81.3.45 erased_part dag< expression_node, _SequenceCtr, _PtrAlloc, _Alloc >::erase_maximal_pregraph (const walker & __position) [inherited]

Reimplemented from **dgraph< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >**.

#### 6.81.3.46 erased_part dag< expression_node, _SequenceCtr, _PtrAlloc, _Alloc >::erase_maximal_subgraph (const __SequenceCtr< walker, _Allocator > & __positions) [inherited]

Reimplemented from **dgraph< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >**.

#### 6.81.3.47 erased_part dag< expression_node, _SequenceCtr, _PtrAlloc, _Alloc >::erase_maximal_subgraph (const walker & __position) [inherited]

Reimplemented from **dgraph< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >**.

#### 6.81.3.48 erased_part dag< expression_node, _SequenceCtr, _PtrAlloc, _Alloc >::erase_minimal_pregraph (const __SequenceCtr< walker, _Allocator > & __positions) [inherited]

Reimplemented from **dgraph< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >**.

#### 6.81.3.49 erased_part dag< expression_node, _SequenceCtr, _PtrAlloc, _Alloc >::erase_minimal_pregraph (const walker & __position) [inherited]

Reimplemented from **dgraph< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >**.

#### 6.81.3.50 erased_part dag< expression_node, _SequenceCtr, _PtrAlloc, _Alloc >::erase_minimal_subgraph (const __SequenceCtr< walker, _Allocator > & __positions) [inherited]

Reimplemented from **dgraph**< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >.

#### 6.81.3.51 erased_part dag< expression_node, _SequenceCtr, _PtrAlloc, _Alloc >::erase_minimal_subgraph (const walker & __position) [inherited]

Reimplemented from **dgraph**< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >.

#### 6.81.3.52 bool dag< expression_node, _SequenceCtr, _PtrAlloc, _Alloc >::erase_parent (const walker & __position, const parents_iterator & __It) [inherited]

Reimplemented from **dgraph**< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >.

#### 6.81.3.53 std::pair< const std::string, double > model::fixed_var (unsigned int *n*) const [inline]

Definition at line 493 of file model-inline.h.

#### 6.81.3.54 void model::free_node_num (unsigned int _nnum) [inline]

Definition at line 1305 of file model-inline.h.

#### 6.81.3.55 allocator_type dag< expression_node, _SequenceCtr, _PtrAlloc, _Alloc >::get_allocator () [inherited]

Reimplemented from **dgraph**< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >.

#### 6.81.3.56 bool model::get_const_num (unsigned int *node_num*, unsigned int & *const_num*) const [inline]

Definition at line 480 of file model-inline.h.

#### 6.81.3.57 bool model::get_linear_coeffs (const walker & *expr*, sparse_vector< double > & *coeffs*, double & *constant*) [inline]

Definition at line 2376 of file model-inline.h.

#### 6.81.3.58 bool model::get_linear_coeffs (const walker & *expr*, sparse_vector< double > & *coeffs*, double & *constant*, const std::vector< interval > & *_ranges*) [inline]

Definition at line 2366 of file model-inline.h.

#### 6.81.3.59 model::walker model::ghost (unsigned int _nnum) [inline]

Definition at line 1421 of file model-inline.h.

#### 6.81.3.60 const_ref_iterator model::ghost_begin () const [inline]

Definition at line 145 of file model.h.

#### 6.81.3.61 ref_iterator model::ghost_begin () [inline]

Definition at line 144 of file model.h.

#### 6.81.3.62 const_ref_iterator model::ghost_end () const [inline]

Definition at line 147 of file model.h.

#### 6.81.3.63 ref_iterator model::ghost_end () [inline]

Definition at line 146 of file model.h.

#### 6.81.3.64 model_gid* model::gid_data () const [inline]

Definition at line 92 of file model.h.

#### 6.81.3.65 const_walker dag< expression_node, _SequenceCtr, _PtrAlloc, _Alloc >::ground () [inherited]

Reimplemented from **dgraph< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >**.

#### 6.81.3.66 walker dag< expression_node, _SequenceCtr, _PtrAlloc, _Alloc >::ground () [inherited]

Reimplemented from **dgraph< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >**.

#### 6.81.3.67 void dag< expression_node, _SequenceCtr, _PtrAlloc, _Alloc >::insert_back_subgraph (_Self & __*subgraph*, const walker & __*parent*, const walker & __*child*) [inherited]

Reimplemented from **dgraph< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >**.

#### 6.81.3.68 void dag< expression_node, _SequenceCtr, _PtrAlloc, _Alloc >::insert_front_subgraph (_Self & __*subgraph*, const walker & __*parent*, const walker & __*child*) [inherited]

Reimplemented from **dgraph< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >**.

#### 6.81.3.69 walker dag< expression_node, _SequenceCtr, _PtrAlloc, _Alloc >::insert_in_graph (const __SequenceCtr< walker, _Allocator > & __*parents*, const walker & __*child*, const container_insert_arg & __*cref*) [inherited]

Reimplemented from **dgraph< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >**.

#### 6.81.3.70 walker dag< expression_node, _SequenceCtr, _PtrAlloc, _Alloc >::insert_in_graph (const expression_node & __*x*, const __SequenceCtr< walker, _Allocator > & __*parents*, const walker & __*child*, const container_insert_arg & __*cref*) [inherited]

#### 6.81.3.71 walker dag< expression_node, _SequenceCtr, _PtrAlloc, _Alloc >::insert_in_graph (const walker & __*parent*, const container_insert_arg & __*pref*, const __SequenceCtr< walker, _Allocator > & __*children*) [inherited]

Reimplemented from **dgraph< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >**.

**6.81.3.72 walker dag< expression_node, _SequenceCtr, _PtrAlloc, _Alloc >::insert_in_graph (const expression_node & __x, const walker & __parent, const container_insert_arg & __pref, const __SequenceCtr< walker, _Allocator > & __children)** [inherited]

**6.81.3.73 walker dag< expression_node, _SequenceCtr, _PtrAlloc, _Alloc >::insert_in_graph (const __SequenceCtr1< walker, _Allocator1 > & __parents, const __SequenceCtr2< walker, _Allocator2 > & __children)** [inherited]

Reimplemented from **dgraph< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >**.

**6.81.3.74 walker dag< expression_node, _SequenceCtr, _PtrAlloc, _Alloc >::insert_in_graph (const expression_node & __x, const __SequenceCtr1< walker, _Allocator1 > & __parents, const __SequenceCtr2< walker, _Allocator2 > & __children)** [inherited]

**6.81.3.75 walker dag< expression_node, _SequenceCtr, _PtrAlloc, _Alloc >::insert_in_graph (const walker & __parent, const walker & __child, const container_insert_arg & __Itc, const container_insert_arg & __Itp)** [inherited]

Reimplemented from **dgraph< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >**.

**6.81.3.76 walker dag< expression_node, _SequenceCtr, _PtrAlloc, _Alloc >::insert_in_graph (const expression_node & __x, const walker & __parent, const walker & __child, const container_insert_arg & __Itc, const container_insert_arg & __Itp)** [inherited]

**6.81.3.77 walker dag< expression_node, _SequenceCtr, _PtrAlloc, _Alloc >::insert_node (const walker & __position, const container_insert_arg & __It)** [inherited]

Reimplemented from **dgraph< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >**.

**6.81.3.78 walker dag< expression_node, _SequenceCtr, _PtrAlloc, _Alloc >::insert_node (const expression_node & __x, const walker & __position, const container_insert_arg & __It)** [inherited]

**6.81.3.79 walker dag< expression_node, _SequenceCtr, _PtrAlloc, _Alloc >::insert_node (_Node * __node, const walker & __position, const container_insert_arg & __It)** [inherited]

Reimplemented from **dgraph< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >**.

**6.81.3.80 void dag< expression_node, _SequenceCtr, _PtrAlloc, _Alloc >::insert_node_before (const walker & __position, const container_insert_arg & __It)** [inherited]

Reimplemented from **dgraph< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >**.

**6.81.3.81 void dag< expression_node, _SequenceCtr, _PtrAlloc, _Alloc >::insert_node_before (const expression_node & __x, const walker & __position, const container_insert_arg & __It)** [inherited]

**6.81.3.82 walker dag< expression_node, _SequenceCtr, _PtrAlloc, _Alloc >::insert_node_before (_Node * *_node*, const walker & *__position*, const container_insert_arg & *__It*)** [inherited]

Reimplemented from **dgraph< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >**.

**6.81.3.83 walker dag< expression_node, _SequenceCtr, _PtrAlloc, _Alloc >::insert_node_in_graph (_Node * *__node*, const __SequenceCtr< walker, _Allocator > & *__parents*, const walker & *__child*, const container_insert_arg & *__cref*)** [inherited]

Reimplemented from **dgraph< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >**.

**6.81.3.84 walker dag< expression_node, _SequenceCtr, _PtrAlloc, _Alloc >::insert_node_in_graph (_Node * *__node*, const walker & *__parent*, const container_insert_arg & *__pref*, const __SequenceCtr< walker, _Allocator > & *__children*)** [inherited]

Reimplemented from **dgraph< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >**.

**6.81.3.85 walker dag< expression_node, _SequenceCtr, _PtrAlloc, _Alloc >::insert_node_in_graph (_Node * *__node*, const __SequenceCtr1< walker, _Allocator1 > & *__parents*, const __SequenceCtr2< walker, _Allocator2 > & *__children*)** [inherited]

Reimplemented from **dgraph< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >**.

**6.81.3.86 walker dag< expression_node, _SequenceCtr, _PtrAlloc, _Alloc >::insert_node_in_graph (_Node * *__n*, const walker & *__parent*, const walker & *__child*, const container_insert_arg & *__Itc*, const container_insert_arg & *__Itp*)** [inherited]

Reimplemented from **dgraph< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >**.

**6.81.3.87 void dag< expression_node, _SequenceCtr, _PtrAlloc, _Alloc >::insert_subgraph (_Self & *__subgraph*, const __SequenceCtr1< walker, _Allocator1 > & *__parents*, const __SequenceCtr2< walker, _Allocator2 > & *__children*)** [inherited]

Reimplemented from **dgraph< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >**.

**6.81.3.88 void dag< expression_node, _SequenceCtr, _PtrAlloc, _Alloc >::insert_subgraph (_Self & *__subgraph*, const walker & *__parent*, const walker & *__child*, const container_insert_arg & *__Itc*, const container_insert_arg & *__Itp*)** [inherited]

Reimplemented from **dgraph< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >**.

**6.81.3.89 void dag< expression_node, _SequenceCtr, _PtrAlloc, _Alloc >::insert_subgraph (_Self & *__subgraph*, const walker & *__parent*, const children_iterator & *__ch_it*, const walker & *__child*, const parents_iterator & *__pa_it*)** [inherited]

Reimplemented from **dgraph< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >**.

**6.81.3.90 bool model::is_empty (const walker & *_w*) const** [inline]

Definition at line 408 of file model-inline.h.

#### 6.81.3.91 parents_iterator dag< expression_node, _SequenceCtr, _PtrAlloc, _Alloc >::leaf_begin () [inherited]

Reimplemented from **dgraph**< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >.

#### 6.81.3.92 parents_iterator dag< expression_node, _SequenceCtr, _PtrAlloc, _Alloc >::leaf_end () [inherited]

Reimplemented from **dgraph**< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >.

#### 6.81.3.93 size_type dag< expression_node, _SequenceCtr, _PtrAlloc, _Alloc >::max_size () [inherited]

Reimplemented from **dgraph**< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >.

#### 6.81.3.94 void dag< expression_node, _SequenceCtr, _PtrAlloc, _Alloc >::merge (const walker & *__position*, const walker & *__second*, bool *merge_parent_edges* = true, bool *merge_child_edges* = true) [inherited]

Reimplemented from **dgraph**< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >.

#### 6.81.3.95 size_t model::n_fixed_vars () [inline]

Definition at line 492 of file model-inline.h.

#### 6.81.3.96 size_t model::n_unused_constrs () [inline]

Definition at line 498 of file model-inline.h.

#### 6.81.3.97 size_t model::n_unused_vars () [inline]

Definition at line 495 of file model-inline.h.

#### 6.81.3.98 model::walker model::nary (const std::vector< walker > & *_op*, int *expr_type*, additional_info_u *_params*, const std::vector< double > & *_coeffs* = std::vector< double >()) [inline]

Definition at line 1522 of file model-inline.h.

#### 6.81.3.99 model::walker model::nary (const std::vector< walker > & *_op*, int *expr_type*, const std::vector< double > & *_coeffs* = std::vector< double >()) [inline]

Definition at line 1503 of file model-inline.h.

#### 6.81.3.100 void model::new_variables (int *_new_num_of_vars*) [inline]

Definition at line 1394 of file model-inline.h.

#### 6.81.3.101 int model::next_constraint_num () [inline]

Definition at line 1212 of file model-inline.h.

#### 6.81.3.102 int model::next_num () [inline]

Definition at line 1210 of file model-inline.h.

#### 6.81.3.103 int model::next_variable_num () [inline]

Definition at line 1211 of file model-inline.h.

#### 6.81.3.104 const model::walker & model::node (unsigned int *i*) const [inline]

Definition at line 512 of file model-inline.h.

#### 6.81.3.105 unsigned int model::number_of_constraints () [inline]

Definition at line 506 of file model-inline.h.

#### 6.81.3.106 unsigned int model::number_of_managed_constraints () const [inline]

Definition at line 86 of file model.h.

#### 6.81.3.107 unsigned int model::number_of_managed_nodes () const [inline]

Definition at line 82 of file model.h.

#### 6.81.3.108 unsigned int model::number_of_managed_variables () const [inline]

Definition at line 84 of file model.h.

#### 6.81.3.109 unsigned int model::number_of_nodes () [inline]

Definition at line 509 of file model-inline.h.

#### 6.81.3.110 unsigned int model::number_of_variables () [inline]

Definition at line 503 of file model-inline.h.

#### 6.81.3.111 double model::obj_adj () [inline]

Definition at line 490 of file model-inline.h.

#### 6.81.3.112 double model::obj_mult () [inline]

Definition at line 491 of file model-inline.h.

#### 6.81.3.113 const std::string model::obj_name () [inline]

Definition at line 489 of file model-inline.h.

#### 6.81.3.114 void dag< expression_node, _SequenceCtr, _PtrAlloc, _Alloc >::remove_edge (const walker & *__parent*, const walker & *__child*) [inherited]

Reimplemented from **dgraph< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >**.

#### 6.81.3.115 void dag< expression_node, _SequenceCtr, _PtrAlloc, _Alloc >::remove_edge (const edge & __edge) [inherited]

Reimplemented from **dgraph< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >**.

#### 6.81.3.116 void dag< expression_node, _SequenceCtr, _PtrAlloc, _Alloc >::remove_edge_and_deattach (const walker & __parent, const walker & __child) [inherited]

Reimplemented from **dgraph< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >**.

#### 6.81.3.117 void model::remove_node (unsigned int __node_num) [inline]

Definition at line 1389 of file model-inline.h.

#### 6.81.3.118 void model::remove_node (const walker & _w) [inline]

Definition at line 1384 of file model-inline.h.

#### 6.81.3.119 void model::remove_node (const walker & _w, unsigned int _nnum) [inline]

Definition at line 1317 of file model-inline.h.

#### 6.81.3.120 void model::renumber_constraints ()

#### 6.81.3.121 void model::renumber_variables () [inline]

Definition at line 1214 of file model-inline.h.

#### 6.81.3.122 void dag< expression_node, _SequenceCtr, _PtrAlloc, _Alloc >::replace_edge_to_child (const walker & __parent, const walker & __child_old, const walker & __child_new) [inherited]

Reimplemented from **dgraph< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >**.

#### 6.81.3.123 void dag< expression_node, _SequenceCtr, _PtrAlloc, _Alloc >::replace_edge_to_parent (const walker & __parent_old, const walker & __parent_new, const walker & __child) [inherited]

Reimplemented from **dgraph< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >**.

#### 6.81.3.124 children_iterator dag< expression_node, _SequenceCtr, _PtrAlloc, _Alloc >::root_begin () [inherited]

Reimplemented from **dgraph< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >**.

#### 6.81.3.125 children_iterator dag< expression_node, _SequenceCtr, _PtrAlloc, _Alloc >::root_end () [inherited]

Reimplemented from **dgraph< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >**.

#### 6.81.3.126 void model::set_counters () [inline]

Definition at line 1545 of file model-inline.h.

#### 6.81.3.127 bool model::simplify_thin () [inline]

Definition at line 1200 of file model-inline.h.

#### 6.81.3.128 size_type dag< expression_node, _SequenceCtr, _PtrAlloc, _Alloc >::size () [inherited]

Reimplemented from **dgraph< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >**.

#### 6.81.3.129 const_walker dag< expression_node, _SequenceCtr, _PtrAlloc, _Alloc >::sky () [inherited]

Reimplemented from **dgraph< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >**.

#### 6.81.3.130 walker dag< expression_node, _SequenceCtr, _PtrAlloc, _Alloc >::sky () [inherited]

Reimplemented from **dgraph< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >**.

#### 6.81.3.131 void dag< expression_node, _SequenceCtr, _PtrAlloc, _Alloc >::sort_child_edges (walker __*position*, Compare *comp*) [inherited]

Reimplemented from **dgraph< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >**.

#### 6.81.3.132 void dag< expression_node, _SequenceCtr, _PtrAlloc, _Alloc >::sort_child_edges (walker __*position*, children_iterator *first*, children_iterator *last*, Compare *comp*) [inherited]

Reimplemented from **dgraph< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >**.

#### 6.81.3.133 void dag< expression_node, _SequenceCtr, _PtrAlloc, _Alloc >::sort_parent_edges (walker __*position*, Compare *comp*) [inherited]

Reimplemented from **dgraph< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >**.

#### 6.81.3.134 void dag< expression_node, _SequenceCtr, _PtrAlloc, _Alloc >::sort_parent_edges (walker __*position*, parents_iterator *first*, parents_iterator *last*, Compare *comp*) [inherited]

Reimplemented from **dgraph< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >**.

#### 6.81.3.135 walker dag< expression_node, _SequenceCtr, _PtrAlloc, _Alloc >::split (const __SequenceCtr< walker, _Allocator > & __*parents*, const walker & __*child*, const parents_iterator & __*pr_it*, const expression_node & __*x*) [inherited]

#### 6.81.3.136 walker dag< expression_node, _SequenceCtr, _PtrAlloc, _Alloc >::split (const walker & __*parent*, const children_iterator & __*ch_it*, const __SequenceCtr< walker, _Allocator > & __*children*, const expression_node & __*x*) [inherited]

**6.81.3.137 void dag< expression_node, _SequenceCtr, _PtrAlloc, _Alloc >::split (const __Sequence-Ctr1< walker, _Allocator1 > & *__parents*, const __SequenceCtr2< walker, _Allocator2 > & *__children*, const expression_node & *__x*)** [inherited]

**6.81.3.138 walker dag< expression_node, _SequenceCtr, _PtrAlloc, _Alloc >::split (const walker & *__parent*, const children_iterator & *__ch_it*, const walker & *__child*, const parents_iterator & *__pa_it*, const expression_node & *__x*)** [inherited]

**6.81.3.139 walker dag< expression_node, _SequenceCtr, _PtrAlloc, _Alloc >::split_back (const __SequenceCtr< walker, _Allocator > & *__parents*, const walker & *__child*, const expression_node & *__x*)** [inherited]

**6.81.3.140 walker dag< expression_node, _SequenceCtr, _PtrAlloc, _Alloc >::split_back (const walker & *__parent*, const __SequenceCtr< walker, _Allocator > & *__children*, const expression_node & *__x*)** [inherited]

**6.81.3.141 walker dag< expression_node, _SequenceCtr, _PtrAlloc, _Alloc >::split_back (const walker & *__parent*, const walker & *__child*, const expression_node & *__x*)** [inherited]

**6.81.3.142 walker dag< expression_node, _SequenceCtr, _PtrAlloc, _Alloc >::split_front (const __SequenceCtr< walker, _Allocator > & *__parents*, const walker & *__child*, const expression_node & *__x*)** [inherited]

**6.81.3.143 walker dag< expression_node, _SequenceCtr, _PtrAlloc, _Alloc >::split_front (const walker & *__parent*, const __SequenceCtr< walker, _Allocator > & *__children*, const expression_node & *__x*)** [inherited]

**6.81.3.144 walker dag< expression_node, _SequenceCtr, _PtrAlloc, _Alloc >::split_front (const walker & *__parent*, const walker & *__child*, const expression_node & *__x*)** [inherited]

**6.81.3.145 model::walker model::store_constraint (const walker & *_w*)** [inline]

Definition at line 1278 of file model-inline.h.

**6.81.3.146 model::walker model::store_ghost (const walker & *_w*)** [inline]

Definition at line 1290 of file model-inline.h.

**6.81.3.147 model::walker model::store_node (const walker & *_w*)** [inline]

Definition at line 1246 of file model-inline.h.

**6.81.3.148 model::walker model::store_variable (const walker & *_w*)** [inline]

Definition at line 1262 of file model-inline.h.

#### 6.81.3.149 void dag< expression_node, _SequenceCtr, _PtrAlloc, _Alloc >::swap (_Self & __x) [inherited]

Reimplemented from **dgraph< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >**.

#### 6.81.3.150 model::walker model::unary (const walker & *_op1*, int *expr_type*, additional_info_u *_params*, double *_coeff* = 1.0) [inline]

Definition at line 1492 of file model-inline.h.

#### 6.81.3.151 model::walker model::unary (const walker & *_op1*, int *expr_type*, double *_coeff* = 1.0) [inline]

Definition at line 1478 of file model-inline.h.

#### 6.81.3.152 const std::string & model::unused_constr (unsigned int *n*) const [inline]

Definition at line 500 of file model-inline.h.

#### 6.81.3.153 const std::string & model::unused_var (unsigned int *n*) const [inline]

Definition at line 496 of file model-inline.h.

#### 6.81.3.154 const model::walker & model::var (unsigned int *i*) const [inline]

Definition at line 515 of file model-inline.h.

#### 6.81.3.155 const std::string model::var_name (unsigned int *n*) const [inline]

Definition at line 485 of file model-inline.h.

#### 6.81.3.156 model::walker model::variable (unsigned int *_vnum*)

Definition at line 2549 of file model.cc.

#### 6.81.3.157 model::walker model::vnary (int *expr_type*, ...)

Definition at line 2593 of file model.cc.

#### 6.81.3.158 void model::write (std::ostream & __o = std::cout) const

Definition at line 2480 of file model.cc.

### 6.81.4 Friends And Related Function Documentation

#### 6.81.4.1 friend class dag_delta [friend]

Definition at line 211 of file model.h.

#### 6.81.4.2 friend class dag_undelta [friend]

Definition at line 212 of file model.h.

#### 6.81.4.3 bool operator==__VGTL_NULL_TMPL_ARGS (const __DG & __x, const __DG & __y) [friend, inherited]

Reimplemented from **dgraph< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >**.

### 6.81.5 Member Data Documentation

#### 6.81.5.1 _DG_node< expression_node, _SequenceCtr< void *, _PtrAlloc >, _SequenceCtr< void *, _PtrAlloc >::iterator > * dag< expression_node, _SequenceCtr, _PtrAlloc, _Alloc >::_C_ground [inherited]

Reimplemented from **dgraph< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >**.

#### 6.81.5.2 int dag< expression_node, _SequenceCtr, _PtrAlloc, _Alloc >::_C_mark [inherited]

Reimplemented from **dgraph< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >**.

#### 6.81.5.3 _DG_node< expression_node, _SequenceCtr< void *, _PtrAlloc >, _SequenceCtr< void *, _PtrAlloc >::iterator > * dag< expression_node, _SequenceCtr, _PtrAlloc, _Alloc >::_C_sky [inherited]

Reimplemented from **dgraph< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >**.

#### 6.81.5.4 _Base::allocator_type dag< expression_node, _SequenceCtr, _PtrAlloc, _Alloc >::allocator_type [inherited]

Reimplemented from **dgraph< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >**.

#### 6.81.5.5 _Base::children_iterator dag< expression_node, _SequenceCtr, _PtrAlloc, _Alloc >::children_iterator [inherited]

Reimplemented from **dgraph< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >**.

#### 6.81.5.6 _DG_iterator< expression_node, const expression_node &, const expression_node *, container_type, children_iterator > dag< expression_node, _SequenceCtr, _PtrAlloc, _Alloc >::const_iterator [inherited]

Reimplemented from **dgraph< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >**.

#### 6.81.5.7 const value_type * __DG< _Tp, _SequenceCtr< void *, _PtrAlloc >, _SequenceCtr< void *, _PtrAlloc >::iterator, _SequenceCtr< void *, _PtrAlloc >::iterator, _Alloc >::const_pointer [inherited]

#### 6.81.5.8 const value_type & __DG< _Tp, _SequenceCtr< void *, _PtrAlloc >, _SequenceCtr< void *, _PtrAlloc >::iterator, _SequenceCtr< void *, _PtrAlloc >::iterator, _Alloc >::const_reference [inherited]

#### 6.81.5.9 std::reverse_iterator< const_iterator > dag< expression_node, _SequenceCtr, _PtrAlloc, _Alloc >::const_reverse_iterator [inherited]

Reimplemented from **dgraph< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >**.

#### 6.81.5.10 std::vector<walker> model::constraints

Definition at line 63 of file model.h.

#### 6.81.5.11 _SequenceCtr< void *, _PtrAlloc > dag< expression_node, _SequenceCtr, _PtrAlloc, _Alloc >::container_type [inherited]

Reimplemented from **dgraph**< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >.

#### 6.81.5.12 ptrdiff_t _DG< _Tp, _SequenceCtr< void *, _PtrAlloc >, _SequenceCtr< void *, _PtrAlloc >::iterator, _SequenceCtr< void *, _PtrAlloc >::iterator, _Alloc >::difference_type [inherited]

#### 6.81.5.13 std::pair< walker, walker > dag< expression_node, _SequenceCtr, _PtrAlloc, _Alloc >::edge [inherited]

Reimplemented from **dgraph**< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >.

#### 6.81.5.14 std::pair< edge, bool > dag< expression_node, _SequenceCtr, _PtrAlloc, _Alloc >::enhanced_edge [inherited]

Reimplemented from **dgraph**< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >.

#### 6.81.5.15 _Base::erased_part dag< expression_node, _SequenceCtr, _PtrAlloc, _Alloc >::erased_part [inherited]

Reimplemented from **dgraph**< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >.

#### 6.81.5.16 _DG_iterator< expression_node, expression_node &, expression_node *, container_type, children_iterator > dag< expression_node, _SequenceCtr, _PtrAlloc, _Alloc >::iterator [inherited]

Reimplemented from **dgraph**< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >.

#### 6.81.5.17 matrix<double> model::lin

Definition at line 58 of file model.h.

#### 6.81.5.18 std::vector<matrix<double> > model::matd

Definition at line 59 of file model.h.

#### 6.81.5.19 std::vector<matrix<interval> > model::mati

Definition at line 60 of file model.h.

#### 6.81.5.20 _Node _DG< _Tp, _SequenceCtr< void *, _PtrAlloc >, _SequenceCtr< void *, _PtrAlloc >::iterator, _SequenceCtr< void *, _PtrAlloc >::iterator, _Alloc >::node_type [inherited]

#### 6.81.5.21 walker model::objective

Definition at line 62 of file model.h.

#### 6.81.5.22 int model::ocoeff

Definition at line 61 of file model.h.

#### 6.81.5.23 _Base::parents_iterator dag< expression_node, _SequenceCtr, _PtrAlloc, _Alloc >::parents_iterator [inherited]

Reimplemented from **dgraph< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >**.

#### 6.81.5.24 value_type * __DG< _Tp, _SequenceCtr< void *, _PtrAlloc >, _SequenceCtr< void *, _PtrAlloc >::iterator, _SequenceCtr< void *, _PtrAlloc >::iterator, _Alloc >::pointer [inherited]

#### 6.81.5.25 value_type & __DG< _Tp, _SequenceCtr< void *, _PtrAlloc >, _SequenceCtr< void *, _PtrAlloc >::iterator, _SequenceCtr< void *, _PtrAlloc >::iterator, _Alloc >::reference [inherited]

#### 6.81.5.26 std::reverse_iterator< iterator > dag< expression_node, _SequenceCtr, _PtrAlloc, _Alloc >::reverse_iterator [inherited]

Reimplemented from **dgraph< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >**.

#### 6.81.5.27 size_t __DG< _Tp, _SequenceCtr< void *, _PtrAlloc >, _SequenceCtr< void *, _PtrAlloc >::iterator, _SequenceCtr< void *, _PtrAlloc >::iterator, _Alloc >::size_type [inherited]

#### 6.81.5.28 expression_node dag< expression_node, _SequenceCtr, _PtrAlloc, _Alloc >::value_type [inherited]

Reimplemented from **dgraph< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >**.

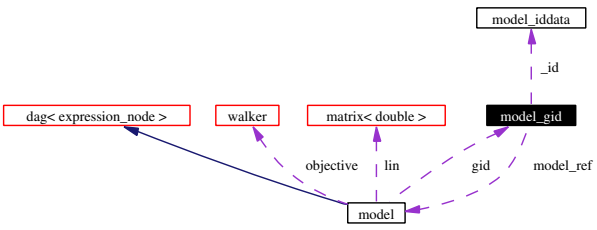
The documentation for this class was generated from the following files:

- model.h
- model-inline.h
- model.cc

## 6.82 model_gid Class Reference

`#include <model.h>`

Collaboration diagram for model_gid:



### Public Methods

- void remove_node_ref (unsigned int _n)
- void remove_var_ref (unsigned int _n)
- void remove_const_ref (unsigned int _n)
- model_gid (model &__m, model_iddata *__i=NULL)
- model_gid (model &__m, unsigned int n, model_iddata *__i=NULL)
- model_gid (model &__mr, const model_gid &__m)
- ~model_gid ()
- unsigned int number_of_nodes () const
- unsigned int number_of_variables () const
- unsigned int number_of_constraints () const
- void number_of_nodes (unsigned int _n)
- void number_of_variables (unsigned int _n)
- void number_of_constraints (unsigned int _n)
- void mk_globref (unsigned int n, const model::walker &__w)
- void mk_gvarref (unsigned int n, const model::walker &__w)
- void mk_gconstref (unsigned int n, const model::walker &__w)
- void make_const_back_ref (unsigned int node, unsigned int cnum)
- unsigned int get_node_id ()
- void remove_node_id (unsigned int n)
- unsigned int get_var_id ()
- void remove_var_id (unsigned int n)
- unsigned int get_const_id ()
- void remove_const_id (unsigned int n)
- void compress_numbers (bool renumber_vars=false, bool renumber_const=false)
- const model::walker & node (unsigned int i) const
- const model::walker & variable (unsigned int i) const
- const model::walker & constraint (unsigned int i) const
- bool empty (const model::walker &__x) const
- bool its_me (const model &__m) const
- model::walker empty_reference () const
- bool have_glob_ref (unsigned int _nnum) const
- bool have_gvar_ref (unsigned int _vnum) const
- bool have_gconst_ref (unsigned int _cnum) const
- const std::string var_name (unsigned int n) const
- void var_name (unsigned int n, const std::string &vn)
- void var_name (unsigned int n, const char *vn)
- const std::string const_name (unsigned int n) const
- void const_name (unsigned int n, const std::string &vn)
- void const_name (unsigned int n, const char *vn)
- bool get_const_num (unsigned int node_num, unsigned int &const_num)
- const std::string obj_name () const
- void obj_name (const std::string &vn)
- void obj_name (const char *vn)
- double obj_adj () const
- void obj_adj (double adj)
- double obj_mult () const
- void obj_mult (double mult)
- size_t n_fixed_vars () const

- std::pair< const std::string, double > fixed_var (unsigned int n) const
- void fixed_var (const std::string &vn, double val)
- void fixed_var (const char *vn, double val)
- size_t n_unused_vars () const
- const std::string & unused_var (unsigned int n) const
- void unused_var (const std::string &vn)
- void unused_var (const char *vn)
- size_t n_unused_constrs () const
- const std::string & unused_constr (unsigned int n) const
- void unused_constr (const std::string &vn)
- void unused_constr (const char *vn)

**Friends**

- class model_iddata

### 6.82.1 Constructor & Destructor Documentation

#### 6.82.1.1 model_gid::model_gid (model & *__m*, model_iddata * *__i* = NULL) [inline]

Definition at line 325 of file model.h.

#### 6.82.1.2 model_gid::model_gid (model & *__m*, unsigned int *n*, model_iddata * *__i* = NULL) [inline]

Definition at line 336 of file model.h.

#### 6.82.1.3 model_gid::model_gid (model & *__mr*, const model_gid & *__m*) [inline]

Definition at line 347 of file model.h.

#### 6.82.1.4 model_gid::~model_gid () [inline]

Definition at line 371 of file model.h.

### 6.82.2 Member Function Documentation

#### 6.82.2.1 void model_gid::compress_numbers (bool *renumber_vars* = false, bool *renumber_const* = false) [inline]

Definition at line 394 of file model.h.

#### 6.82.2.2 void model_gid::const_name (unsigned int *n*, const char * *vn*) [inline]

Definition at line 424 of file model.h.

#### 6.82.2.3 void model_gid::const_name (unsigned int *n*, const std::string & *vn*) [inline]

Definition at line 422 of file model.h.

#### 6.82.2.4 const std::string model_gid::const_name (unsigned int *n*) const [inline]

Definition at line 420 of file model.h.

#### 6.82.2.5 const model::walker& model_gid::constraint (unsigned int *i*) const [inline]

Definition at line 399 of file model.h.

#### 6.82.2.6 bool model_gid::empty (const model::walker & *__x*) const [inline]

Definition at line 402 of file model.h.

#### 6.82.2.7 model::walker model_gid::empty_reference () const [inline]

Definition at line 405 of file model.h.

#### 6.82.2.8 void model_gid::fixed_var (const char * *vn*, double *val*) [inline]

Definition at line 440 of file model.h.

#### 6.82.2.9 void model_gid::fixed_var (const std::string & *vn*, double *val*) [inline]

Definition at line 439 of file model.h.

#### 6.82.2.10 std::pair<const std::string, double> model_gid::fixed_var (unsigned int *n*) const [inline]

Definition at line 437 of file model.h.

#### 6.82.2.11 unsigned int model_gid::get_const_id () [inline]

Definition at line 392 of file model.h.

#### 6.82.2.12 bool model_gid::get_const_num (unsigned int *node_num*, unsigned int & *const_num*) [inline]

Definition at line 388 of file model-inline.h.

#### 6.82.2.13 unsigned int model_gid::get_node_id () [inline]

Definition at line 388 of file model.h.

#### 6.82.2.14 unsigned int model_gid::get_var_id () [inline]

Definition at line 390 of file model.h.

#### 6.82.2.15 bool model_gid::have_gconst_ref (unsigned int *_cnum*) const [inline]

Definition at line 411 of file model.h.

#### 6.82.2.16 bool model_gid::have_glob_ref (unsigned int *_nnum*) const [inline]

Definition at line 407 of file model.h.

#### 6.82.2.17 bool model_gid::have_gvar_ref (unsigned int _vnum) const [inline]

Definition at line 409 of file model.h.

#### 6.82.2.18 bool model_gid::its_me (const model & __m) const [inline]

Definition at line 404 of file model.h.

#### 6.82.2.19 void model_gid::make_const_back_ref (unsigned int *node*, unsigned int *cnum*) [inline]

Definition at line 399 of file model-inline.h.

#### 6.82.2.20 void model_gid::mk_gconstref (unsigned int *n*, const model::walker & __w) [inline]

Definition at line 348 of file model-inline.h.

#### 6.82.2.21 void model_gid::mk_globref (unsigned int *n*, const model::walker & __w) [inline]

Definition at line 322 of file model-inline.h.

#### 6.82.2.22 void model_gid::mk_gvarref (unsigned int *n*, const model::walker & __w) [inline]

Definition at line 335 of file model-inline.h.

#### 6.82.2.23 size_t model_gid::n_fixed_vars () const [inline]

Definition at line 436 of file model.h.

#### 6.82.2.24 size_t model_gid::n_unused_constrs () const [inline]

Definition at line 449 of file model.h.

#### 6.82.2.25 size_t model_gid::n_unused_vars () const [inline]

Definition at line 443 of file model.h.

#### 6.82.2.26 const model::walker& model_gid::node (unsigned int *i*) const [inline]

Definition at line 397 of file model.h.

#### 6.82.2.27 void model_gid::number_of_constraints (unsigned int _n) [inline]

Definition at line 380 of file model.h.

#### 6.82.2.28 unsigned int model_gid::number_of_constraints () const [inline]

Definition at line 376 of file model.h.

#### 6.82.2.29 void model_gid::number_of_nodes (unsigned int _n) [inline]

Definition at line 378 of file model.h.

#### 6.82.2.30 unsigned int model_gid::number_of_nodes () const [inline]

Definition at line 373 of file model.h.

#### 6.82.2.31 void model_gid::number_of_variables (unsigned int _n) [inline]

Definition at line 379 of file model.h.

#### 6.82.2.32 unsigned int model_gid::number_of_variables () const [inline]

Definition at line 374 of file model.h.

#### 6.82.2.33 void model_gid::obj_adj (double *adj*) [inline]

Definition at line 432 of file model.h.

#### 6.82.2.34 double model_gid::obj_adj () const [inline]

Definition at line 431 of file model.h.

#### 6.82.2.35 void model_gid::obj_mult (double *mult*) [inline]

Definition at line 434 of file model.h.

#### 6.82.2.36 double model_gid::obj_mult () const [inline]

Definition at line 433 of file model.h.

#### 6.82.2.37 void model_gid::obj_name (const char * *vn*) [inline]

Definition at line 430 of file model.h.

#### 6.82.2.38 void model_gid::obj_name (const std::string & *vn*) [inline]

Definition at line 429 of file model.h.

#### 6.82.2.39 const std::string model_gid::obj_name () const [inline]

Definition at line 428 of file model.h.

#### 6.82.2.40 void model_gid::remove_const_id (unsigned int *n*) [inline]

Definition at line 393 of file model.h.

#### 6.82.2.41 void model_gid::remove_const_ref (unsigned int _n) [inline]

Definition at line 366 of file model-inline.h.

#### 6.82.2.42 void model_gid::remove_node_id (unsigned int *n*) [inline]

Definition at line 389 of file model.h.

#### 6.82.2.43 void model_gid::remove_node_ref (unsigned int _n) [inline]

Definition at line 317 of file model.h.

#### 6.82.2.44 void model_gid::remove_var_id (unsigned int n) [inline]

Definition at line 391 of file model.h.

#### 6.82.2.45 void model_gid::remove_var_ref (unsigned int _n) [inline]

Definition at line 320 of file model.h.

#### 6.82.2.46 void model_gid::unused_constr (const char * vn) [inline]

Definition at line 453 of file model.h.

#### 6.82.2.47 void model_gid::unused_constr (const std::string & vn) [inline]

Definition at line 452 of file model.h.

#### 6.82.2.48 const std::string& model_gid::unused_constr (unsigned int n) const [inline]

Definition at line 450 of file model.h.

#### 6.82.2.49 void model_gid::unused_var (const char * vn) [inline]

Definition at line 447 of file model.h.

#### 6.82.2.50 void model_gid::unused_var (const std::string & vn) [inline]

Definition at line 446 of file model.h.

#### 6.82.2.51 const std::string& model_gid::unused_var (unsigned int n) const [inline]

Definition at line 444 of file model.h.

#### 6.82.2.52 void model_gid::var_name (unsigned int n, const char * vn) [inline]

Definition at line 417 of file model.h.

#### 6.82.2.53 void model_gid::var_name (unsigned int n, const std::string & vn) [inline]

Definition at line 416 of file model.h.

#### 6.82.2.54 const std::string model_gid::var_name (unsigned int n) const [inline]

Definition at line 415 of file model.h.

#### 6.82.2.55 const model::walker& model_gid::variable (unsigned int i) const [inline]

Definition at line 398 of file model.h.

### 6.82.3 Friends And Related Function Documentation

#### 6.82.3.1 friend class model_iddata [friend]

Definition at line 455 of file model.h.

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

- model.h
- model-inline.h
- model.cc

## 6.83 model_iddata Class Reference

`#include <model.h>`

### Public Methods

- model_iddata (unsigned int n=0)
- ~model_iddata ()
- void new_ref (model_gid &__m)
- bool delete_ref (model_gid &__m)
- unsigned int number_of_nodes () const
- unsigned int number_of_variables () const
- unsigned int number_of_constraints () const
- void number_of_nodes (unsigned int _n)
- void number_of_variables (unsigned int _n)
- void number_of_constraints (unsigned int _n)
- unsigned int get_node_id ()
- void remove_node_id (unsigned int n)
- unsigned int get_var_id ()
- void remove_var_id (unsigned int n)
- unsigned int get_const_id ()
- void remove_const_id (unsigned int n)
- void compress_numbers (bool renum_vars, bool renum_consts=false)
- const std::string var_name (unsigned int n) const
- void var_name (unsigned int n, const std::string &vn)
- const std::string const_name (unsigned int n) const
- void const_name (unsigned int n, const std::string &cn)
- const std::string obj_name () const
- void obj_name (const std::string &vn)
- double obj_adj () const
- void obj_adj (double adj)
- double obj_mult () const
- void obj_mult (double mult)
- size_t n_fixed_vars () const
- std::pair< const std::string, double > fixed_var (unsigned int n) const
- void fixed_var (const std::string &vn, double val)
- size_t n_unused_vars () const
- const std::string & unused_var (unsigned int n) const
- void unused_var (const std::string &vn)

- size_t n_unused_constrs () const
- const std::string & unused_constr (unsigned int n) const
- void unused_constr (const std::string &vn)

**Friends**

- class model_gid

### 6.83.1 Constructor & Destructor Documentation

#### 6.83.1.1 model_iddata::model_iddata (unsigned int *n* = 0) [inline]

Definition at line 237 of file model.h.

#### 6.83.1.2 model_iddata::~model_iddata () [inline]

Definition at line 246 of file model.h.

### 6.83.2 Member Function Documentation

#### 6.83.2.1 void model_iddata::compress_numbers (bool *renum_vars*, bool *renum_consts* = false)

Definition at line 31 of file model.cc.

#### 6.83.2.2 void model_iddata::const_name (unsigned int *n*, const std::string & *cn*) [inline]

Definition at line 182 of file model-inline.h.

#### 6.83.2.3 const std::string model_iddata::const_name (unsigned int *n*) const [inline]

Definition at line 170 of file model-inline.h.

#### 6.83.2.4 bool model_iddata::delete_ref (model_gid & *__m*) [inline]

Definition at line 33 of file model-inline.h.

#### 6.83.2.5 void model_iddata::fixed_var (const std::string & *vn*, double *val*) [inline]

Definition at line 219 of file model-inline.h.

#### 6.83.2.6 std::pair< const std::string, double > model_iddata::fixed_var (unsigned int *n*) const [inline]

Definition at line 208 of file model-inline.h.

#### 6.83.2.7 unsigned int model_iddata::get_const_id () [inline]

Definition at line 95 of file model-inline.h.

#### 6.83.2.8 unsigned int model_iddata::get_node_id () [inline]

Definition at line 47 of file model-inline.h.

#### 6.83.2.9 unsigned int model_iddata::get_var_id () [inline]

Definition at line 70 of file model-inline.h.

#### 6.83.2.10 size_t model_iddata::n_fixed_vars () const [inline]

Definition at line 282 of file model.h.

#### 6.83.2.11 size_t model_iddata::n_unused_constrs () const [inline]

Definition at line 290 of file model.h.

#### 6.83.2.12 size_t model_iddata::n_unused_vars () const [inline]

Definition at line 286 of file model.h.

#### 6.83.2.13 void model_iddata::new_ref (model_gid & *__m*) [inline]

Definition at line 248 of file model.h.

#### 6.83.2.14 void model_iddata::number_of_constraints (unsigned int *_n*) [inline]

Definition at line 137 of file model-inline.h.

#### 6.83.2.15 unsigned int model_iddata::number_of_constraints () const [inline]

Definition at line 254 of file model.h.

#### 6.83.2.16 void model_iddata::number_of_nodes (unsigned int *_n*) [inline]

Definition at line 120 of file model-inline.h.

#### 6.83.2.17 unsigned int model_iddata::number_of_nodes () const [inline]

Definition at line 252 of file model.h.

#### 6.83.2.18 void model_iddata::number_of_variables (unsigned int *_n*) [inline]

Definition at line 128 of file model-inline.h.

#### 6.83.2.19 unsigned int model_iddata::number_of_variables () const [inline]

Definition at line 253 of file model.h.

#### 6.83.2.20 void model_iddata::obj_adj (double *adj*) [inline]

Definition at line 204 of file model-inline.h.

#### 6.83.2.21 double model_iddata::obj_adj () [inline]

Definition at line 203 of file model-inline.h.

#### 6.83.2.22 void model_iddata::obj_mult (double *mult*) [inline]

Definition at line 206 of file model-inline.h.

#### 6.83.2.23 double model_iddata::obj_mult () [inline]

Definition at line 205 of file model-inline.h.

#### 6.83.2.24 void model_iddata::obj_name (const std::string & *vn*) [inline]

Definition at line 202 of file model-inline.h.

#### 6.83.2.25 const std::string model_iddata::obj_name () [inline]

Definition at line 194 of file model-inline.h.

#### 6.83.2.26 void model_iddata::remove_const_id (unsigned int *n*) [inline]

Definition at line 107 of file model-inline.h.

#### 6.83.2.27 void model_iddata::remove_node_id (unsigned int *n*) [inline]

Definition at line 59 of file model-inline.h.

#### 6.83.2.28 void model_iddata::remove_var_id (unsigned int *n*) [inline]

Definition at line 82 of file model-inline.h.

#### 6.83.2.29 void model_iddata::unused_constr (const std::string & *vn*) [inline]

Definition at line 249 of file model-inline.h.

#### 6.83.2.30 const std::string & model_iddata::unused_constr (unsigned int *n*) const [inline]

Definition at line 239 of file model-inline.h.

#### 6.83.2.31 void model_iddata::unused_var (const std::string & *vn*) [inline]

Definition at line 234 of file model-inline.h.

#### 6.83.2.32 const std::string & model_iddata::unused_var (unsigned int *n*) const [inline]

Definition at line 224 of file model-inline.h.

#### 6.83.2.33 void model_iddata::var_name (unsigned int *n*, const std::string & *vn*) [inline]

Definition at line 158 of file model-inline.h.

#### 6.83.2.34 const std::string model_iddata::var_name (unsigned int *n*) const [inline]

Definition at line 146 of file model-inline.h.

### 6.83.3 Friends And Related Function Documentation

#### 6.83.3.1 friend class model_gid [friend]

Definition at line 294 of file model.h.

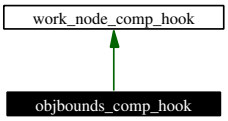
The documentation for this class was generated from the following files:

- model.h
- model-inline.h
- model.cc

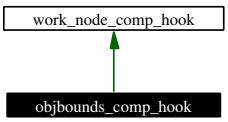
## 6.84 objbounds_comp_hook Class Reference

`#include <objbounds_hook.h>`

Inheritance diagram for objbounds_comp_hook:




Collaboration diagram for objbounds_comp_hook:




### Public Methods

- objbounds_comp_hook ()
- void operator() (const work_node &wn, dbt_row &dbr)
- bool init_columns (vdbl::standard_table &stable)
- bool drop_columns (vdbl::standard_table &stable)

### Protected Methods

- template<class _CI> bool _init_column (vdbl::standard_table &stable, const std::string &colname, const _CI &c)
- template<class _CI> bool _init_column (vdbl::standard_table &stable, const char *colname, const _CI &c)
- bool _drop_columns (vdbl::standard_table &stable)
- search_node_relation parent_relation (const work_node &wn) const
- search_node_id node_id (const work_node &wn) const
- const std::string & name ()

### 6.84.1 Constructor & Destructor Documentation

#### 6.84.1.1 objbounds_comp_hook::objbounds_comp_hook () [inline]

Definition at line 35 of file objbounds_hook.h.

### 6.84.2 Member Function Documentation

#### 6.84.2.1 bool work_node_comp_hook::_drop_columns (vdbl::standard_table & *stable*) [protected, inherited]

Definition at line 42 of file comp_hook.cc.

#### 6.84.2.2 template<class _CI> bool work_node_comp_hook::_init_column (vdbl::standard_table & *stable*, const char * *colname*, const _CI & *c*) [inline, protected, inherited]

Definition at line 45 of file comp_hook.h.

#### 6.84.2.3 template<class _CI> bool work_node_comp_hook::_init_column (vdbl::standard_table & *stable*, const std::string & *colname*, const _CI & *c*) [protected, inherited]

Definition at line 31 of file comp_hook.cc.

#### 6.84.2.4 bool objbounds_comp_hook::drop_columns (vdbl::standard_table & *stable*) [inline, virtual]

Reimplemented from work_node_comp_hook.

Definition at line 56 of file objbounds_hook.h.

#### 6.84.2.5 bool objbounds_comp_hook::init_columns (vdbl::standard_table & *stable*) [inline, virtual]

Reimplemented from work_node_comp_hook.

Definition at line 49 of file objbounds_hook.h.

#### 6.84.2.6 const std::string& work_node_comp_hook::name () [inline, inherited]

Definition at line 68 of file comp_hook.h.

#### 6.84.2.7 search_node_id work_node_comp_hook::node_id (const work_node & *wn*) const [protected, inherited]

Definition at line 51 of file comp_hook.cc.

#### 6.84.2.8 void objbounds_comp_hook::operator() (const work_node & *wn*, dbt_row & *dbr*) [inline, virtual]

Implements work_node_comp_hook.

Definition at line 39 of file objbounds_hook.h.

#### 6.84.2.9 search_node_relation work_node_comp_hook::parent_relation (const work_node & *wn*) const [protected, inherited]

Definition at line 54 of file comp_hook.cc.

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

- objbounds_hook.h

## 6.85 parents_compare Class Reference

`#include <expr-inline.h>`

### Public Methods

- bool operator() (const expression_node &__x, const expression_node &__y) const

### 6.85.1 Member Function Documentation

#### 6.85.1.1 bool parents_compare::operator() (const expression_node & *__x*, const expression_node & *__y*) const [inline]

Definition at line 34 of file expr-inline.h.

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

- expr-inline.h

## 6.86 parents_compare_eq Class Reference

`#include <expr-inline.h>`

### Public Methods

- bool operator() (const expression_node &__x, const expression_node &__y) const

### 6.86.1 Member Function Documentation

#### 6.86.1.1 bool parents_compare_eq::operator() (const expression_node & *__x*, const expression_node & *__y*) const [inline]

Definition at line 122 of file expr-inline.h.

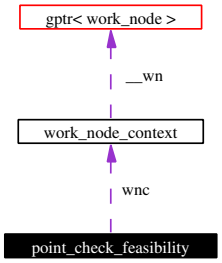
The documentation for this class was generated from the following file:

- expr-inline.h

## 6.87 point_check_feasibility Class Reference

`#include <dbtools.h>`

Collaboration diagram for point_check_feasibility:




### Public Types

- typedef work_node_context context
- typedef bool return_type

### Public Methods

- point_check_feasibility (**vdbl::colid** _x, **vdbl::colid** _L, **vdbl::colid** _f)
- point_check_feasibility (const point_check_feasibility &_i)
- virtual ~point_check_feasibility ()
- bool operator() () const
- bool def () const
- void setcontext (const context *c, const vdbl::row *r)

#### 6.87.1 Member Typedef Documentation

##### 6.87.1.1 typedef work_node_context point_check_feasibility::context

Definition at line 53 of file dbtools.h.

##### 6.87.1.2 typedef bool point_check_feasibility::return_type

Definition at line 62 of file dbtools.h.

#### 6.87.2 Constructor & Destructor Documentation

##### 6.87.2.1 point_check_feasibility::point_check_feasibility (vdbl::colid _x, vdbl::colid _L, vdbl::colid _f) [inline]

Definition at line 64 of file dbtools.h.

##### 6.87.2.2 point_check_feasibility::point_check_feasibility (const point_check_feasibility & _i) [inline]

Definition at line 66 of file dbtools.h.

##### 6.87.2.3 virtual point_check_feasibility::~point_check_feasibility () [inline, virtual]

Definition at line 69 of file dbtools.h.

### 6.87.3 Member Function Documentation

#### 6.87.3.1 bool point_check_feasibility::def () const [inline]

Definition at line 72 of file dbtools.h.

#### 6.87.3.2 bool point_check_feasibility::operator() ()

Definition at line 31 of file dbtools.cc.

#### 6.87.3.3 void point_check_feasibility::setcontext (const context * *c*, const vdbl::row * *r*) [inline]

Definition at line 73 of file dbtools.h.

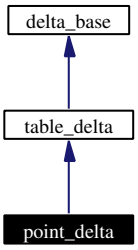
The documentation for this class was generated from the following files:

- dbtools.h
- dbtools.cc

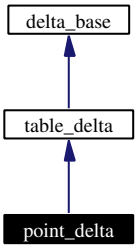
## 6.88 point_delta Class Reference

`#include <point_delta.h>`

Inheritance diagram for point_delta:




Collaboration diagram for point_delta:




### Public Types

- typedef std::pair< std::string, dbt_row > t_line
- typedef std::vector< t_line > t_ctr

### Public Methods

- point_delta ()
- point_delta (const dbt_row &__p)
- point_delta (const point_delta &__d)
- point_delta * new_copy () const
- void destroy_copy (point_delta *__d)
- void create_table (work_node &_x, vdbl::standard_table *&ptb, const std::string &__t) const
- bool apply (work_node &_x, undelta_base *&_u) const
- void add (const t_line &_tl)
- void add (const std::string &_tn, const dbt_row &_r)
- void add (const std::vector< t_line > &_tlv)
- void rm (const annotation &_tr)
- void rm (const std::vector< annotation > &_trv)
- virtual void destroy_copy (table_delta *__d)
- virtual void destroy_copy (delta_base *__d)
- void convert (work_node &_x, delta_base *&_u)
- delta make_delta (const std::string &a)
- const std::string & get_action () const
- virtual bool apply3 (work_node &_x, const work_node &_y, undelta_base *&_u) const

### Protected Attributes

- std::string _action

#### 6.88.1 Member Typedef Documentation

##### 6.88.1.1 typedef std::vector<t_line> table_delta::t_ctr [inherited]

Definition at line 36 of file table_delta.h.

##### 6.88.1.2 typedef std::pair<std::string,dbt_row> table_delta::t_line [inherited]

Definition at line 35 of file table_delta.h.

#### 6.88.2 Constructor & Destructor Documentation

##### 6.88.2.1 point_delta::point_delta () [inline]

Definition at line 39 of file point_delta.h.

##### 6.88.2.2 point_delta::point_delta (const dbt_row & *__p*) [inline]

Definition at line 41 of file point_delta.h.

##### 6.88.2.3 point_delta::point_delta (const point_delta & *__d*) [inline]

Definition at line 45 of file point_delta.h.

### 6.88.3 Member Function Documentation

#### 6.88.3.1 void table_delta::add (const std::vector< t_line > & *_tlv*) [inline, inherited]

Definition at line 73 of file table_delta.h.

#### 6.88.3.2 void table_delta::add (const std::string & *_tn*, const dbt_row & *_r*) [inline, inherited]

Definition at line 71 of file table_delta.h.

#### 6.88.3.3 void table_delta::add (const t_line & *_tl*) [inline, inherited]

Definition at line 70 of file table_delta.h.

#### 6.88.3.4 bool point_delta::apply (work_node & *_x*, undelta_base *& *_u*) const [virtual]

Reimplemented from table_delta.

Definition at line 30 of file point_delta.cc.

#### 6.88.3.5 bool delta_base::apply3 (work_node & *_x*, const work_node & *_y*, undelta_base *& *_u*) const [inline, virtual, inherited]

Definition at line 63 of file api_delta.h.

#### 6.88.3.6 void table_delta::convert (work_node & *_x*, delta_base *& *_u*) [virtual, inherited]

Reimplemented from delta_base.

Definition at line 37 of file table_delta.cc.

#### 6.88.3.7 void point_delta::create_table (work_node & *_x*, vdbl::standard_table *& *ptb*, const std::string & *__t*) const [virtual]

Reimplemented from table_delta.

Definition at line 38 of file point_delta.cc.

#### 6.88.3.8 virtual void delta_base::destroy_copy (delta_base * *__d*) [inline, virtual, inherited]

Definition at line 95 of file api_deltabase.h.

#### 6.88.3.9 virtual void table_delta::destroy_copy (table_delta * *__d*) [inline, virtual, inherited]

Definition at line 81 of file table_delta.h.

#### 6.88.3.10 void point_delta::destroy_copy (point_delta * *__d*) [inline]

Definition at line 54 of file point_delta.h.

#### 6.88.3.11 const std::string& delta_base::get_action () const [inline, inherited]

Definition at line 105 of file api_deltabase.h.

#### 6.88.3.12 delta delta_base::make_delta (const std::string & *a*) [inline, inherited]

Definition at line 99 of file api_deltabase.h.

#### 6.88.3.13 point_delta* point_delta::new_copy () const [inline, virtual]

Reimplemented from table_delta.

Definition at line 52 of file point_delta.h.

#### 6.88.3.14 void table_delta::rm (const std::vector< annotation > & *_trv*) [inline, inherited]

Definition at line 76 of file table_delta.h.

#### 6.88.3.15 void table_delta::rm (const annotation & *_tr*) [inline, inherited]

Definition at line 75 of file table_delta.h.

### 6.88.4 Member Data Documentation

#### 6.88.4.1 std::string delta_base::_action [protected, inherited]

Definition at line 86 of file api_deltabase.h.

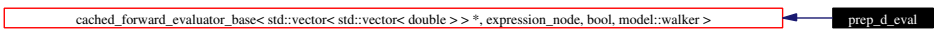
The documentation for this class was generated from the following files:

- point_delta.h
- point_delta.cc

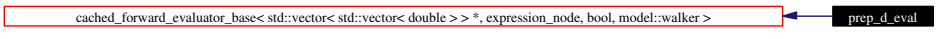
## 6.89 prep_d_eval Class Reference

`#include <der_evaluator.h>`

Inheritance diagram for prep_d_eval:




Collaboration diagram for prep_d_eval:




### Public Types

- typedef cached_evaluator_base< std::vector< std::vector< double > > *, expression_node, bool, model::walker >::node_data_type node_data_type

- typedef cached_evaluator_base< std::vector< std::vector< double > > *, expression_node, bool, model::walker >::return_value return_value
- typedef cached_evaluator_base< std::vector< std::vector< double > > *, expression_node, bool, model::walker >::walker walker
- typedef std::vector< std::vector< double > > * data_type

**Public Methods**

- prep_d_eval (std::vector< std::vector< double > > &__d, unsigned int _num_of_nodes)
- prep_d_eval (const prep_d_eval &__x)
- ~prep_d_eval ()
- void initialize ()
- bool is_cached (const expression_node &__data)
- void retrieve_from_cache (const expression_node &__data)
- int initialize (const expression_node &__data)
- void calculate (const expression_node &__data)
- int update (bool __rval)
- int update (const expression_node &__data, bool __rval)
- bool calculate_value (bool eval_all)
- virtual bool is_cached (const node_data_type &__data)
- int preorder (const node_data_type &__data)
- void postorder (const node_data_type &__data)
- int collect (const node_data_type &__data, const return_value &__rval)
- int vcollect (const return_value &__rval)
- return_value value ()
- return_value vvalue ()
- void vinit ()
- virtual int initialize (const node_data_type &__data)
- virtual void calculate (const node_data_type &__data)
- virtual void retrieve_from_cache (const node_data_type &__data)
- virtual void cleanup (const node_data_type &__data)
- virtual int update (const node_data_type &__data, const return_value &__rval)
- virtual int update (const return_value &__rval)
- virtual walker short_cut_to (const node_data_type &__data) PURE_VIRTUAL public

**Protected Attributes**

- const variable_indicator * v_ind
- std::vector< std::vector< double > > * eval_data

### 6.89.1 Member Typedef Documentation

#### 6.89.1.1 typedef std::vector< std::vector< double > > * _evaluator_base< std::vector< std::vector< double > > *, expression_node, bool, model::walker >::data_type [inherited]

Definition at line 245 of file evaluator.h.

#### 6.89.1.2 typedef cached_evaluator_base<std::vector< std::vector< double > > *, expression_node, bool,model::walker>::node_data_type cached_forward_evaluator_base< std::vector< std::vector< double > > *, expression_node, bool, model::walker >::node_data_type [inherited]

Reimplemented from cached_evaluator_base< std::vector< std::vector< double > > *, expression_node, bool, model::walker >.

Definition at line 396 of file evaluator.h.

#### 6.89.1.3 typedef cached_evaluator_base<std::vector< std::vector< double > > *, expression_node, bool,model::walker>::return_value cached_forward_evaluator_base< std::vector< std::vector< double > > *, expression_node, bool, model::walker >::return_value [inherited]

Reimplemented from cached_evaluator_base< std::vector< std::vector< double > > *, expression_node, bool, model::walker >.

Definition at line 398 of file evaluator.h.

#### 6.89.1.4 typedef cached_evaluator_base<std::vector< std::vector< double > > *, expression_node, bool,model::walker>::walker cached_forward_evaluator_base< std::vector< std::vector< double > > *, expression_node, bool, model::walker >::walker [inherited]

Reimplemented from cached_evaluator_base< std::vector< std::vector< double > > *, expression_node, bool, model::walker >.

Definition at line 400 of file evaluator.h.

### 6.89.2 Constructor & Destructor Documentation

#### 6.89.2.1 prep_d_eval::prep_d_eval (std::vector< std::vector< double > > & *__d*, unsigned int *_num_of_nodes*) [inline]

Definition at line 55 of file der_evaluator.h.

#### 6.89.2.2 prep_d_eval::prep_d_eval (const prep_d_eval & *__x*) [inline]

Definition at line 64 of file der_evaluator.h.

#### 6.89.2.3 prep_d_eval::~prep_d_eval () [inline]

Definition at line 66 of file der_evaluator.h.

### 6.89.3 Member Function Documentation

#### 6.89.3.1 virtual void cached_forward_evaluator_base< std::vector< std::vector< double > > *, expression_node, bool, model::walker >::calculate (const node_data_type & *__data*) [inline, virtual, inherited]

Definition at line 430 of file evaluator.h.

#### 6.89.3.2 void prep_d_eval::calculate (const expression_node & *__data*) [inline]

Definition at line 84 of file der_evaluator.h.

#### 6.89.3.3 bool prep_d_eval::calculate_value (bool *eval_all*) [inline, virtual]

Reimplemented from cached_forward_evaluator_base< std::vector< std::vector< double > > *, expression_node, bool, model::walker >.

Definition at line 91 of file der_evaluator.h.

#### 6.89.3.4 virtual void cached_forward_evaluator_base< std::vector< std::vector< double > > *, expression_node, bool, model::walker >::cleanup (const node_data_type & *__data*) [inline, virtual, inherited]

Definition at line 432 of file evaluator.h.

#### 6.89.3.5 int cached_forward_evaluator_base< std::vector< std::vector< double > > *, expression_node, bool, model::walker >::collect (const node_data_type & *__data*, const return_value & *__rval*) [inline, virtual, inherited]

Reimplemented from _evaluator_base< std::vector< std::vector< double > > *, expression_node, bool, model::walker >.

Definition at line 419 of file evaluator.h.

#### 6.89.3.6 virtual int cached_forward_evaluator_base< std::vector< std::vector< double > > *, expression_node, bool, model::walker >::initialize (const node_data_type & *__data*) [inline, virtual, inherited]

Definition at line 429 of file evaluator.h.

#### 6.89.3.7 int prep_d_eval::initialize (const expression_node & *__data*) [inline]

Definition at line 77 of file der_evaluator.h.

#### 6.89.3.8 void prep_d_eval::initialize () [inline, virtual]

Reimplemented from cached_forward_evaluator_base< std::vector< std::vector< double > > *, expression_node, bool, model::walker >.

Definition at line 68 of file der_evaluator.h.

#### 6.89.3.9 virtual bool cached_forward_evaluator_base< std::vector< std::vector< double > > *, expression_node, bool, model::walker >::is_cached (const node_data_type & *__data*) [inline, virtual, inherited]

Definition at line 405 of file evaluator.h.

#### 6.89.3.10 bool prep_d_eval::is_cached (const expression_node & *__data*) [inline]

Definition at line 70 of file der_evaluator.h.

#### 6.89.3.11 void cached_forward_evaluator_base< std::vector< std::vector< double > > *, expression_node, bool, model::walker >::postorder (const node_data_type & *__data*) [inline, virtual, inherited]

Reimplemented from _evaluator_base< std::vector< std::vector< double > > *, expression_node, bool, model::walker >.

Definition at line 417 of file evaluator.h.

#### 6.89.3.12 int cached_forward_evaluator_base< std::vector< std::vector< double > > *, expression_node, bool, model::walker >::preorder (const node_data_type & *__data*) [inline, virtual, inherited]

Reimplemented from cached_evaluator_base< std::vector< std::vector< double > > *, expression_node, bool, model::walker >.

Definition at line 408 of file evaluator.h.

#### 6.89.3.13 virtual void cached_forward_evaluator_base< std::vector< std::vector< double > > *, expression_node, bool, model::walker >::retrieve_from_cache (const node_data_type & *__data*) [inline, virtual, inherited]

Definition at line 431 of file evaluator.h.

#### 6.89.3.14 void prep_d_eval::retrieve_from_cache (const expression_node & *__data*) [inline]

Definition at line 75 of file der_evaluator.h.

#### 6.89.3.15 virtual walker cached_evaluator_base< std::vector< std::vector< double > > *, expression_node, bool, model::walker >::short_cut_to (const node_data_type & *__data*) [inline, virtual, inherited]

Definition at line 303 of file evaluator.h.

#### 6.89.3.16 virtual int cached_forward_evaluator_base< std::vector< std::vector< double > > *, expression_node, bool, model::walker >::update (const return_value & *__rval*) [inline, virtual, inherited]

Definition at line 435 of file evaluator.h.

#### 6.89.3.17 virtual int cached_forward_evaluator_base< std::vector< std::vector< double > > *, expression_node, bool, model::walker >::update (const node_data_type & *__data*, const return_value & *__rval*) [inline, virtual, inherited]

Definition at line 433 of file evaluator.h.

#### 6.89.3.18 int prep_d_eval::update (const expression_node & *__data*, bool *__rval*) [inline]

Definition at line 88 of file der_evaluator.h.

#### 6.89.3.19 int prep_d_eval::update (bool *__rval*) [inline]

Definition at line 86 of file der_evaluator.h.

#### 6.89.3.20 return_value cached_forward_evaluator_base< std::vector< std::vector< double > > *, expression_node, bool, model::walker >::value () [inline, virtual, inherited]

Reimplemented from _evaluator_base< std::vector< std::vector< double > > *, expression_node, bool, model::walker >.

Definition at line 423 of file evaluator.h.

#### 6.89.3.21 int cached_forward_evaluator_base< std::vector< std::vector< double > > *, expression_node, bool, model::walker >::vcollect (const return_value & *__rval*) [inline, virtual, inherited]

Reimplemented from _evaluator_base< std::vector< std::vector< double > > *, expression_node, bool, model::walker >.

Definition at line 421 of file evaluator.h.

#### 6.89.3.22 void cached_forward_evaluator_base< std::vector< std::vector< double > > *, expression_node, bool, model::walker >::vinit () [inline, inherited]

Definition at line 425 of file evaluator.h.

#### 6.89.3.23 return_value cached_forward_evaluator_base< std::vector< std::vector< double > > *, expression_node, bool, model::walker >::vvalue () [inline, virtual, inherited]

Reimplemented from _evaluator_base< std::vector< std::vector< double > > *, expression_node, bool, model::walker >.

Definition at line 424 of file evaluator.h.

### 6.89.4 Member Data Documentation

#### 6.89.4.1 std::vector< std::vector< double > > * _evaluator_base< std::vector< std::vector< double > > *, expression_node, bool, model::walker >::eval_data [protected, inherited]

Definition at line 252 of file evaluator.h.

#### 6.89.4.2 const variable_indicator* cached_evaluator_base< std::vector< std::vector< double > > *, expression_node, bool, model::walker >::v_ind [protected, inherited]

Definition at line 295 of file evaluator.h.

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

- der_evaluator.h

## 6.90 prep_h_eval_tp Struct Reference

`#include <hess_evaluator.h>`

### Public Attributes

- std::vector< std::vector< double > > * d
- std::vector< matrix< double > > * h

### 6.90.1 Member Data Documentation

#### 6.90.1.1 std::vector<std::vector<double> >* prep_h_eval_tp::d

Definition at line 51 of file hess_evaluator.h.

#### 6.90.1.2 std::vector<matrix<double> >* prep_h_eval_tp::h

Definition at line 52 of file hess_evaluator.h.

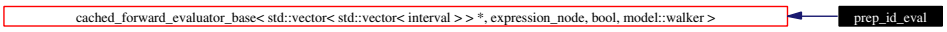
The documentation for this struct was generated from the following file:

- hess_evaluator.h

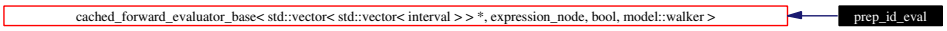
## 6.91 prep_id_eval Class Reference

`#include <ider_evaluator.h>`

Inheritance diagram for prep_id_eval:




Collaboration diagram for prep_id_eval:




### Public Types

- typedef cached_evaluator_base< std::vector< std::vector< interval > > *, expression_node, bool, model::walker >::node_data_type node_data_type
- typedef cached_evaluator_base< std::vector< std::vector< interval > > *, expression_node, bool, model::walker >::return_value return_value
- typedef cached_evaluator_base< std::vector< std::vector< interval > > *, expression_node, bool, model::walker >::walker walker
- typedef std::vector< std::vector< interval > > * data_type

### Public Methods

- prep_id_eval (std::vector< std::vector< interval > > &__d, unsigned int _num_of_nodes)
- prep_id_eval (const prep_id_eval &__x)
- ~prep_id_eval ()
- void initialize ()
- bool is_cached (const expression_node &__data)
- void retrieve_from_cache (const expression_node &__data)
- int initialize (const expression_node &__data)
- void calculate (const expression_node &__data)
- int update (bool __rval)
- int update (const expression_node &__data, bool __rval)
- bool calculate_value (bool eval_all)
- virtual bool is_cached (const node_data_type &__data)
- int preorder (const node_data_type &__data)
- void postorder (const node_data_type &__data)
- int collect (const node_data_type &__data, const return_value &__rval)
- int vcollect (const return_value &__rval)
- return_value value ()
- return_value vvalue ()

- void vinit ()
- virtual int initialize (const node_data_type &__data)
- virtual void calculate (const node_data_type &__data)
- virtual void retrieve_from_cache (const node_data_type &__data)
- virtual void cleanup (const node_data_type &__data)
- virtual int update (const node_data_type &__data, const return_value &__rval)
- virtual int update (const return_value &__rval)
- virtual walker short_cut_to (const node_data_type &__data) PURE_VIRTUAL public

**Protected Attributes**

- const variable_indicator * v_ind
- std::vector< std::vector< interval > > * eval_data

### 6.91.1 Member Typedef Documentation

**6.91.1.1 typedef std::vector< std::vector< interval > > * _evaluator_base< std::vector< std::vector< interval > > *, expression_node, bool, model::walker >::data_type** `[inherited]`

Definition at line 245 of file evaluator.h.

**6.91.1.2 typedef cached_evaluator_base<std::vector< std::vector< interval > > *, expression_node, bool,model::walker>::node_data_type cached_forward_evaluator_base< std::vector< std::vector< interval > > *, expression_node, bool, model::walker >::node_data_type** `[inherited]`

Reimplemented from cached_evaluator_base< std::vector< std::vector< interval > > *, expression_node, bool, model::walker >.

Definition at line 396 of file evaluator.h.

**6.91.1.3 typedef cached_evaluator_base<std::vector< std::vector< interval > > *, expression_node, bool,model::walker>::return_value cached_forward_evaluator_base< std::vector< std::vector< interval > > *, expression_node, bool, model::walker >::return_value** `[inherited]`

Reimplemented from cached_evaluator_base< std::vector< std::vector< interval > > *, expression_node, bool, model::walker >.

Definition at line 398 of file evaluator.h.

**6.91.1.4 typedef cached_evaluator_base<std::vector< std::vector< interval > > *, expression_node, bool,model::walker>::walker cached_forward_evaluator_base< std::vector< std::vector< interval > > *, expression_node, bool, model::walker >::walker** `[inherited]`

Reimplemented from cached_evaluator_base< std::vector< std::vector< interval > > *, expression_node, bool, model::walker >.

Definition at line 400 of file evaluator.h.

### 6.91.2 Constructor & Destructor Documentation

#### 6.91.2.1 prep_id_eval::prep_id_eval (std::vector< std::vector< interval > > & *__d*, unsigned int *_num_of_nodes*) [inline]

Definition at line 56 of file ider_evaluator.h.

#### 6.91.2.2 prep_id_eval::prep_id_eval (const prep_id_eval & *__x*) [inline]

Definition at line 65 of file ider_evaluator.h.

#### 6.91.2.3 prep_id_eval::~prep_id_eval () [inline]

Definition at line 67 of file ider_evaluator.h.

### 6.91.3 Member Function Documentation

#### 6.91.3.1 virtual void cached_forward_evaluator_base< std::vector< std::vector< interval > > *, expression_node, bool, model::walker >::calculate (const node_data_type & *__data*) [inline, virtual, inherited]

Definition at line 430 of file evaluator.h.

#### 6.91.3.2 void prep_id_eval::calculate (const expression_node & *__data*) [inline]

Definition at line 85 of file ider_evaluator.h.

#### 6.91.3.3 bool prep_id_eval::calculate_value (bool *eval_all*) [inline, virtual]

Reimplemented from cached_forward_evaluator_base< std::vector< std::vector< interval > > *, expression_node, bool, model::walker >.

Definition at line 92 of file ider_evaluator.h.

#### 6.91.3.4 virtual void cached_forward_evaluator_base< std::vector< std::vector< interval > > *, expression_node, bool, model::walker >::cleanup (const node_data_type & *__data*) [inline, virtual, inherited]

Definition at line 432 of file evaluator.h.

#### 6.91.3.5 int cached_forward_evaluator_base< std::vector< std::vector< interval > > *, expression_node, bool, model::walker >::collect (const node_data_type & *__data*, const return_value & *__rval*) [inline, virtual, inherited]

Reimplemented from _evaluator_base< std::vector< std::vector< interval > > *, expression_node, bool, model::walker >.

Definition at line 419 of file evaluator.h.

#### 6.91.3.6 virtual int cached_forward_evaluator_base< std::vector< std::vector< interval > > *, expression_node, bool, model::walker >::initialize (const node_data_type & *__data*) [inline, virtual, inherited]

Definition at line 429 of file evaluator.h.

#### 6.91.3.7 int prep_id_eval::initialize (const expression_node & *__data*) [inline]

Definition at line 78 of file ider_evaluator.h.

#### 6.91.3.8 void prep_id_eval::initialize () [inline, virtual]

Reimplemented from cached_forward_evaluator_base< std::vector< std::vector< interval > > *, expression_node, bool, model::walker >.

Definition at line 69 of file ider_evaluator.h.

#### 6.91.3.9 virtual bool cached_forward_evaluator_base< std::vector< std::vector< interval > > *, expression_node, bool, model::walker >::is_cached (const node_data_type & *__data*) [inline, virtual, inherited]

Definition at line 405 of file evaluator.h.

#### 6.91.3.10 bool prep_id_eval::is_cached (const expression_node & *__data*) [inline]

Definition at line 71 of file ider_evaluator.h.

#### 6.91.3.11 void cached_forward_evaluator_base< std::vector< std::vector< interval > > *, expression_node, bool, model::walker >::postorder (const node_data_type & *__data*) [inline, virtual, inherited]

Reimplemented from _evaluator_base< std::vector< std::vector< interval > > *, expression_node, bool, model::walker >.

Definition at line 417 of file evaluator.h.

#### 6.91.3.12 int cached_forward_evaluator_base< std::vector< std::vector< interval > > *, expression_node, bool, model::walker >::preorder (const node_data_type & *__data*) [inline, virtual, inherited]

Reimplemented from cached_evaluator_base< std::vector< std::vector< interval > > *, expression_node, bool, model::walker >.

Definition at line 408 of file evaluator.h.

#### 6.91.3.13 virtual void cached_forward_evaluator_base< std::vector< std::vector< interval > > *, expression_node, bool, model::walker >::retrieve_from_cache (const node_data_type & *__data*) [inline, virtual, inherited]

Definition at line 431 of file evaluator.h.

#### 6.91.3.14 void prep_id_eval::retrieve_from_cache (const expression_node & *__data*) [inline]

Definition at line 76 of file ider_evaluator.h.

#### 6.91.3.15 virtual walker cached_evaluator_base< std::vector< std::vector< interval > > *, expression_node, bool, model::walker >::short_cut_to (const node_data_type & *__data*) [inline, virtual, inherited]

Definition at line 303 of file evaluator.h.

#### 6.91.3.16 virtual int cached_forward_evaluator_base< std::vector< std::vector< interval > > *, expression_node, bool, model::walker >::update (const return_value & __rval) [inline, virtual, inherited]

Definition at line 435 of file evaluator.h.

#### 6.91.3.17 virtual int cached_forward_evaluator_base< std::vector< std::vector< interval > > *, expression_node, bool, model::walker >::update (const node_data_type & __data, const return_value & __rval) [inline, virtual, inherited]

Definition at line 433 of file evaluator.h.

#### 6.91.3.18 int prep_id_eval::update (const expression_node & __data, bool __rval) [inline]

Definition at line 89 of file ider_evaluator.h.

#### 6.91.3.19 int prep_id_eval::update (bool __rval) [inline]

Definition at line 87 of file ider_evaluator.h.

#### 6.91.3.20 return_value cached_forward_evaluator_base< std::vector< std::vector< interval > > *, expression_node, bool, model::walker >::value () [inline, virtual, inherited]

Reimplemented from _evaluator_base< std::vector< std::vector< interval > > *, expression_node, bool, model::walker >.

Definition at line 423 of file evaluator.h.

#### 6.91.3.21 int cached_forward_evaluator_base< std::vector< std::vector< interval > > *, expression_node, bool, model::walker >::vcollect (const return_value & __rval) [inline, virtual, inherited]

Reimplemented from _evaluator_base< std::vector< std::vector< interval > > *, expression_node, bool, model::walker >.

Definition at line 421 of file evaluator.h.

#### 6.91.3.22 void cached_forward_evaluator_base< std::vector< std::vector< interval > > *, expression_node, bool, model::walker >::vinit () [inline, inherited]

Definition at line 425 of file evaluator.h.

#### 6.91.3.23 return_value cached_forward_evaluator_base< std::vector< std::vector< interval > > *, expression_node, bool, model::walker >::vvalue () [inline, virtual, inherited]

Reimplemented from _evaluator_base< std::vector< std::vector< interval > > *, expression_node, bool, model::walker >.

Definition at line 424 of file evaluator.h.

### 6.91.4 Member Data Documentation

#### 6.91.4.1 std::vector< std::vector< interval > > * _evaluator_base< std::vector< std::vector< interval > > *, expression_node, bool, model::walker >::eval_data [protected, inherited]

Definition at line 252 of file evaluator.h.

#### 6.91.4.2 const variable_indicator* cached_evaluator_base< std::vector< std::vector< interval > > *, expression_node, bool, model::walker >::v_ind [protected, inherited]

Definition at line 295 of file evaluator.h.

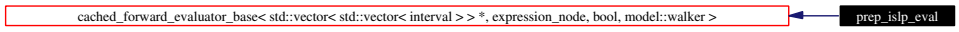
The documentation for this class was generated from the following file:

- ider_evaluator.h

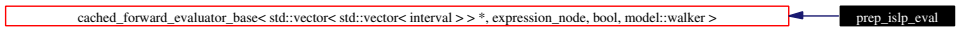
## 6.92 prep_islp_eval Class Reference

`#include <islp_evaluator.h>`

Inheritance diagram for prep_islp_eval:




Collaboration diagram for prep_islp_eval:




### Public Types

- typedef cached_evaluator_base< std::vector< std::vector< interval > > *, expression_node, bool, model::walker >::node_data_type node_data_type
- typedef cached_evaluator_base< std::vector< std::vector< interval > > *, expression_node, bool, model::walker >::return_value return_value
- typedef cached_evaluator_base< std::vector< std::vector< interval > > *, expression_node, bool, model::walker >::walker walker
- typedef std::vector< std::vector< interval > > * data_type

### Public Methods

- prep_islp_eval (std::vector< std::vector< interval > > &__d, unsigned int _num_of_nodes)
- prep_islp_eval (const prep_islp_eval &__x)
- ~prep_islp_eval ()
- void initialize ()
- bool is_cached (const expression_node &__data)
- void retrieve_from_cache (const expression_node &__data)
- int initialize (const expression_node &__data)
- void calculate (const expression_node &__data)
- int update (bool __rval)
- int update (const expression_node &__data, bool __rval)
- bool calculate_value (bool eval_all)
- virtual bool is_cached (const node_data_type &__data)
- int preorder (const node_data_type &__data)
- void postorder (const node_data_type &__data)

- int collect (const node_data_type &__data, const return_value &__rval)
- int vcollect (const return_value &__rval)
- return_value value ()
- return_value vvalue ()
- void vinit ()
- virtual int initialize (const node_data_type &__data)
- virtual void calculate (const node_data_type &__data)
- virtual void retrieve_from_cache (const node_data_type &__data)
- virtual void cleanup (const node_data_type &__data)
- virtual int update (const node_data_type &__data, const return_value &__rval)
- virtual int update (const return_value &__rval)
- virtual walker short_cut_to (const node_data_type &__data) PURE_VIRTUAL public

**Protected Attributes**

- const variable_indicator * v_ind
- std::vector< std::vector< interval > > * eval_data

### 6.92.1 Member Typedef Documentation

**6.92.1.1 typedef std::vector< std::vector< interval > > * _evaluator_base< std::vector< std::vector< interval > > *, expression_node, bool, model::walker >::data_type** `[inherited]`

Definition at line 245 of file evaluator.h.

**6.92.1.2 typedef cached_evaluator_base<std::vector< std::vector< interval > > *, expression_node, bool,model::walker>::node_data_type cached_forward_evaluator_base< std::vector< std::vector< interval > > *, expression_node, bool, model::walker >::node_data_type** `[inherited]`

Reimplemented from cached_evaluator_base< std::vector< std::vector< interval > > *, expression_node, bool, model::walker >.

Definition at line 396 of file evaluator.h.

**6.92.1.3 typedef cached_evaluator_base<std::vector< std::vector< interval > > *, expression_node, bool,model::walker>::return_value cached_forward_evaluator_base< std::vector< std::vector< interval > > *, expression_node, bool, model::walker >::return_value** `[inherited]`

Reimplemented from cached_evaluator_base< std::vector< std::vector< interval > > *, expression_node, bool, model::walker >.

Definition at line 398 of file evaluator.h.

**6.92.1.4 typedef cached_evaluator_base<std::vector< std::vector< interval > > *, expression_node, bool,model::walker>::walker cached_forward_evaluator_base< std::vector< std::vector< interval > > *, expression_node, bool, model::walker >::walker** `[inherited]`

Reimplemented from cached_evaluator_base< std::vector< std::vector< interval > > *, expression_node, bool, model::walker >.

Definition at line 400 of file evaluator.h.

### 6.92.2 Constructor & Destructor Documentation

#### 6.92.2.1 prep_islp_eval::prep_islp_eval (std::vector< std::vector< interval > > & *__d*, unsigned int *_num_of_nodes*) [inline]

Definition at line 63 of file islp_evaluator.h.

#### 6.92.2.2 prep_islp_eval::prep_islp_eval (const prep_islp_eval & *__x*) [inline]

Definition at line 72 of file islp_evaluator.h.

#### 6.92.2.3 prep_islp_eval::~prep_islp_eval () [inline]

Definition at line 74 of file islp_evaluator.h.

### 6.92.3 Member Function Documentation

#### 6.92.3.1 virtual void cached_forward_evaluator_base< std::vector< std::vector< interval > > *, expression_node, bool, model::walker >::calculate (const node_data_type & *__data*) [inline, virtual, inherited]

Definition at line 430 of file evaluator.h.

#### 6.92.3.2 void prep_islp_eval::calculate (const expression_node & *__data*) [inline]

Definition at line 92 of file islp_evaluator.h.

#### 6.92.3.3 bool prep_islp_eval::calculate_value (bool *eval_all*) [inline, virtual]

Reimplemented from cached_forward_evaluator_base< std::vector< std::vector< interval > > *, expression_node, bool, model::walker >.

Definition at line 99 of file islp_evaluator.h.

#### 6.92.3.4 virtual void cached_forward_evaluator_base< std::vector< std::vector< interval > > *, expression_node, bool, model::walker >::cleanup (const node_data_type & *__data*) [inline, virtual, inherited]

Definition at line 432 of file evaluator.h.

#### 6.92.3.5 int cached_forward_evaluator_base< std::vector< std::vector< interval > > *, expression_node, bool, model::walker >::collect (const node_data_type & *__data*, const return_value & *_rval*) [inline, virtual, inherited]

Reimplemented from _evaluator_base< std::vector< std::vector< interval > > *, expression_node, bool, model::walker >.

Definition at line 419 of file evaluator.h.

#### 6.92.3.6 virtual int cached_forward_evaluator_base< std::vector< std::vector< interval > > *, expression_node, bool, model::walker >::initialize (const node_data_type & *__data*) [inline, virtual, inherited]

Definition at line 429 of file evaluator.h.

#### 6.92.3.7 int prep_islp_eval::initialize (const expression_node & *__data*) [inline]

Definition at line 85 of file islp_evaluator.h.

#### 6.92.3.8 void prep_islp_eval::initialize () [inline, virtual]

Reimplemented from cached_forward_evaluator_base< std::vector< std::vector< interval > > *, expression_node, bool, model::walker >.

Definition at line 76 of file islp_evaluator.h.

#### 6.92.3.9 virtual bool cached_forward_evaluator_base< std::vector< std::vector< interval > > *, expression_node, bool, model::walker >::is_cached (const node_data_type & *__data*) [inline, virtual, inherited]

Definition at line 405 of file evaluator.h.

#### 6.92.3.10 bool prep_islp_eval::is_cached (const expression_node & *__data*) [inline]

Definition at line 78 of file islp_evaluator.h.

#### 6.92.3.11 void cached_forward_evaluator_base< std::vector< std::vector< interval > > *, expression_node, bool, model::walker >::postorder (const node_data_type & *__data*) [inline, virtual, inherited]

Reimplemented from _evaluator_base< std::vector< std::vector< interval > > *, expression_node, bool, model::walker >.

Definition at line 417 of file evaluator.h.

#### 6.92.3.12 int cached_forward_evaluator_base< std::vector< std::vector< interval > > *, expression_node, bool, model::walker >::preorder (const node_data_type & *__data*) [inline, virtual, inherited]

Reimplemented from cached_evaluator_base< std::vector< std::vector< interval > > *, expression_node, bool, model::walker >.

Definition at line 408 of file evaluator.h.

#### 6.92.3.13 virtual void cached_forward_evaluator_base< std::vector< std::vector< interval > > *, expression_node, bool, model::walker >::retrieve_from_cache (const node_data_type & *__data*) [inline, virtual, inherited]

Definition at line 431 of file evaluator.h.

#### 6.92.3.14 void prep_islp_eval::retrieve_from_cache (const expression_node & *__data*) [inline]

Definition at line 83 of file islp_evaluator.h.

#### 6.92.3.15 virtual walker cached_evaluator_base< std::vector< std::vector< interval > > *, expression_node, bool, model::walker >::short_cut_to (const node_data_type & *__data*) [inline, virtual, inherited]

Definition at line 303 of file evaluator.h.

#### 6.92.3.16 virtual int cached_forward_evaluator_base< std::vector< std::vector< interval > > *, expression_node, bool, model::walker >::update (const return_value & __rval) [inline, virtual, inherited]

Definition at line 435 of file evaluator.h.

#### 6.92.3.17 virtual int cached_forward_evaluator_base< std::vector< std::vector< interval > > *, expression_node, bool, model::walker >::update (const node_data_type & __data, const return_value & __rval) [inline, virtual, inherited]

Definition at line 433 of file evaluator.h.

#### 6.92.3.18 int prep_islp_eval::update (const expression_node & __data, bool __rval) [inline]

Definition at line 96 of file islp_evaluator.h.

#### 6.92.3.19 int prep_islp_eval::update (bool __rval) [inline]

Definition at line 94 of file islp_evaluator.h.

#### 6.92.3.20 return_value cached_forward_evaluator_base< std::vector< std::vector< interval > > *, expression_node, bool, model::walker >::value () [inline, virtual, inherited]

Reimplemented from _evaluator_base< std::vector< std::vector< interval > > *, expression_node, bool, model::walker >.

Definition at line 423 of file evaluator.h.

#### 6.92.3.21 int cached_forward_evaluator_base< std::vector< std::vector< interval > > *, expression_node, bool, model::walker >::vcollect (const return_value & __rval) [inline, virtual, inherited]

Reimplemented from _evaluator_base< std::vector< std::vector< interval > > *, expression_node, bool, model::walker >.

Definition at line 421 of file evaluator.h.

#### 6.92.3.22 void cached_forward_evaluator_base< std::vector< std::vector< interval > > *, expression_node, bool, model::walker >::vinit () [inline, inherited]

Definition at line 425 of file evaluator.h.

#### 6.92.3.23 return_value cached_forward_evaluator_base< std::vector< std::vector< interval > > *, expression_node, bool, model::walker >::vvalue () [inline, virtual, inherited]

Reimplemented from _evaluator_base< std::vector< std::vector< interval > > *, expression_node, bool, model::walker >.

Definition at line 424 of file evaluator.h.

### 6.92.4 Member Data Documentation

#### 6.92.4.1 std::vector< std::vector< interval > > * _evaluator_base< std::vector< std::vector< interval > > *, expression_node, bool, model::walker >::eval_data [protected, inherited]

Definition at line 252 of file evaluator.h.

#### 6.92.4.2 const variable_indicator* cached_evaluator_base< std::vector< std::vector< interval > > *, expression_node, bool, model::walker >::v_ind [protected, inherited]

Definition at line 295 of file evaluator.h.

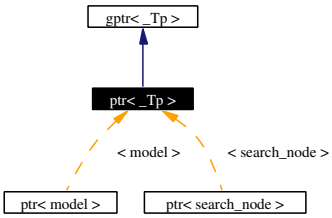
The documentation for this class was generated from the following file:

- islp_evaluator.h

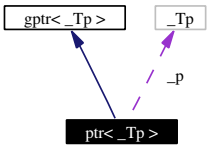
## 6.93 ptr< _Tp > Class Template Reference

`#include <gptr.h>`

Inheritance diagram for ptr< _Tp >:




Collaboration diagram for ptr< _Tp >:




### Public Methods

- ptr (_Tp &__p)
- ptr (_Tp *__p)
- ptr (const _Self &__p)
- ~ptr ()
- reference operator * ()
- const_reference operator * () const
- pointer operator → ()
- const_pointer operator → () const
- pointer get_local_copy ()
- const_pointer get_local_copy () const
- _Self & operator= (const _Self &__p)
- _Self & operator= (_Tp &__p)

**template<class _Tp> class ptr< _Tp >**

### 6.93.1 Constructor & Destructor Documentation

#### 6.93.1.1 template<class _Tp> ptr< _Tp >::ptr (_Tp & __p) [inline]

Definition at line 39 of file gptr.h.

#### 6.93.1.2 template<class _Tp> ptr< _Tp >::ptr (_Tp * __p) [inline]

Definition at line 40 of file gptr.h.

#### 6.93.1.3 template<class _Tp> ptr< _Tp >::ptr (const _Self & __p) [inline]

Definition at line 41 of file gptr.h.

#### 6.93.1.4 template<class _Tp> ptr< _Tp >::~ptr () [inline]

Definition at line 43 of file gptr.h.

### 6.93.2 Member Function Documentation

#### 6.93.2.1 template<class _Tp> const_pointer ptr< _Tp >::get_local_copy () const [inline]

Definition at line 52 of file gptr.h.

#### 6.93.2.2 template<class _Tp> pointer ptr< _Tp >::get_local_copy () [inline]

Definition at line 51 of file gptr.h.

#### 6.93.2.3 template<class _Tp> const_reference ptr< _Tp >::operator * () const [inline]

Definition at line 46 of file gptr.h.

#### 6.93.2.4 template<class _Tp> reference ptr< _Tp >::operator * () [inline]

Definition at line 45 of file gptr.h.

#### 6.93.2.5 template<class _Tp> const_pointer ptr< _Tp >::operator -> () const [inline]

Definition at line 49 of file gptr.h.

#### 6.93.2.6 template<class _Tp> pointer ptr< _Tp >::operator -> () [inline]

Definition at line 48 of file gptr.h.

#### 6.93.2.7 template<class _Tp> _Self& ptr< _Tp >::operator= (_Tp & __p) [inline]

Definition at line 55 of file gptr.h.

#### 6.93.2.8 template<class _Tp> _Self& ptr< _Tp >::operator= (const _Self & __p) [inline]

Definition at line 54 of file gptr.h.

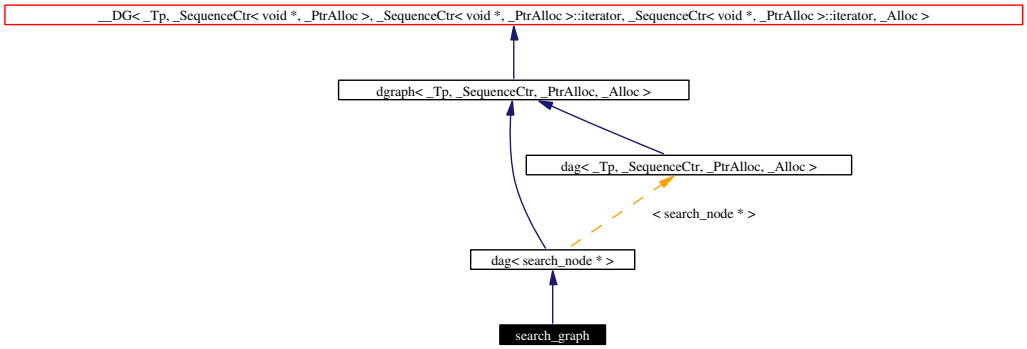
The documentation for this class was generated from the following file:

- gptr.h

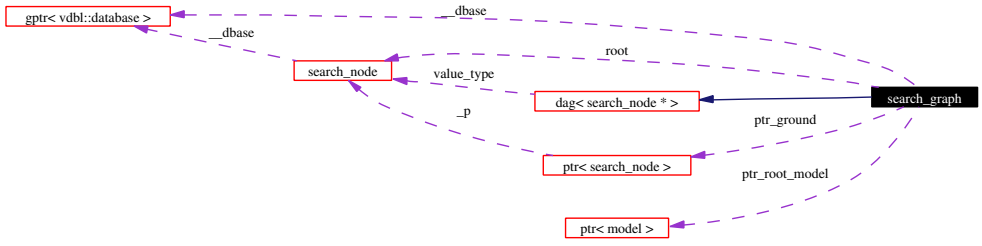
## 6.94 search_graph Class Reference

`#include <search_graph.h>`

Inheritance diagram for search_graph:




Collaboration diagram for search_graph:




### Public Methods

- search_graph (model &root_model, gptr< vdbl::database > &_db)
- search_graph (model &root_model, gptr< vdbl::database > &_db, const std::vector< annotation > &_a)
- ~search_graph ()
- search_node_id get_node_id ()
- search_focus & new_focus (const search_inspector &_si)
- void destroy_focus (const search_focus &_sf)
- search_focus & set_focus (search_focus &_fc, const search_inspector &_si)
- search_inspector & new_inspector (const search_inspector &inspector_to_add)
- search_inspector & new_inspector ()

- void destroy_inspector (const search_inspector &inspector_to_destroy)
- work_node extract (const search_inspector &where)
- work_node extract (const search_focus &where)
- search_inspector & insert (const search_focus &where, const search_node &what)
- search_inspector & replace (search_focus &where, const search_node &what)
- void remove (search_focus &_sf)
- void remove_upwards (search_focus &_sf, bool relaxation_kills)
- search_focus & promote (search_focus &_sf)
- search_inspector child (search_inspector &n, unsigned int i)
- search_inspector parent (search_inspector &n, unsigned int i)
- bool check_acyclicity (const **walker** &__parent, const **walker** &__child)
- void clear ()
- **walker** between (const **walker** &__parent, const **children_iterator** &__cit, const **walker** &__child, const **parents_iterator** &__pit, const search_node *&__x)
- **walker** between (const __SequenceCtr1< **walker**, _Allocator1 > &__parents, const __SequenceCtr2< **walker**, _Allocator2 > &__children, const search_node *&__x)
- **walker** between (const **walker** &__parent, const **children_iterator** &__cit, const __SequenceCtr< **walker**, _Allocator > &__children, const search_node *&__x)
- **walker** between (const __SequenceCtr< **walker**, _Allocator > &__parents, const **walker** &__child, const **parents_iterator** &__pit, const search_node *&__x)
- **walker between** (const **walker** &__parent, const **children_iterator** &__cit, const **walker** &__child, const **parents_iterator** &__pit, const _Tp &__x)
- **walker between** (const __SequenceCtr1< **walker**, _Allocator1 > &__parents, const __SequenceCtr2< **walker**, _Allocator2 > &__children, const _Tp &__x)
- **walker between** (const **walker** &__parent, const **children_iterator** &__cit, const __SequenceCtr< **walker**, _Allocator > &__children, const _Tp &__x)
- **walker between** (const __SequenceCtr< **walker**, _Allocator > &__parents, const **walker** &__child, const **parents_iterator** &__pit, const _Tp &__x)
- **walker** split (const **walker** &__parent, const **children_iterator** &__ch_it, const **walker** &__child, const **parents_iterator** &__pa_it, const search_node *&__x)
- void split (const __SequenceCtr1< **walker**, _Allocator1 > &__parents, const __SequenceCtr2< **walker**, _Allocator2 > &__children, const search_node *&__x)
- **walker** split (const **walker** &__parent, const **children_iterator** &__ch_it, const __SequenceCtr< **walker**, _Allocator > &__children, const search_node *&__x)
- **walker** split (const __SequenceCtr< **walker**, _Allocator > &__parents, const **walker** &__child, const **parents_iterator** &__pr_it, const search_node *&__x)
- **walker split** (const **walker** &__parent, const **children_iterator** &__ch_it, const **walker** &__child, const **parents_iterator** &__pa_it, const _Tp &__x)
- void **split** (const __SequenceCtr1< **walker**, _Allocator1 > &__parents, const __SequenceCtr2< **walker**, _Allocator2 > &__children, const _Tp &__x)
- **walker split** (const **walker** &__parent, const **children_iterator** &__ch_it, const __SequenceCtr< **walker**, _Allocator > &__children, const _Tp &__x)
- **walker split** (const __SequenceCtr< **walker**, _Allocator > &__parents, const **walker** &__child, const **parents_iterator** &__pr_it, const _Tp &__x)
- **walker** between_back (const **walker** &__parent, const **walker** &__child, const search_node *&__x)
- **walker** between_back (const **walker** &__parent, const __SequenceCtr< **walker**, _Allocator > &__children, const search_node *&__x)
- **walker** between_back (const __SequenceCtr< **walker**, _Allocator > &__parents, const **walker** &__child, const search_node *&__x)
- **walker between_back** (const **walker** &__parent, const **walker** &__child, const _Tp &__x)
- **walker between_back** (const **walker** &__parent, const __SequenceCtr< **walker**, _Allocator > &__children, const _Tp &__x)

- **walker between_back** (const __SequenceCtr< **walker**, _Allocator > &__parents, const **walker** &__child, const _Tp &__x)
- **walker** split_back (const **walker** &__parent, const **walker** &__child, const search_node *&__x)
- **walker** split_back (const **walker** &__parent, const __SequenceCtr< **walker**, _Allocator > &__children, const search_node *&__x)
- **walker** split_back (const __SequenceCtr< **walker**, _Allocator > &__parents, const **walker** &__child, const search_node *&__x)
- **walker split_back** (const **walker** &__parent, const **walker** &__child, const _Tp &__x)
- **walker split_back** (const **walker** &__parent, const __SequenceCtr< **walker**, _Allocator > &__children, const _Tp &__x)
- **walker split_back** (const __SequenceCtr< **walker**, _Allocator > &__parents, const **walker** &__child, const _Tp &__x)
- **walker** between_front (const **walker** &__parent, const **walker** &__child, const search_node *&__x)
- **walker** between_front (const **walker** &__parent, const __SequenceCtr< **walker**, _Allocator > &__children, const search_node *&__x)
- **walker** between_front (const __SequenceCtr< **walker**, _Allocator > &__parents, const **walker** &__child, const search_node *&__x)
- **walker between_front** (const **walker** &__parent, const **walker** &__child, const _Tp &__x)
- **walker between_front** (const **walker** &__parent, const __SequenceCtr< **walker**, _Allocator > &__children, const _Tp &__x)
- **walker between_front** (const __SequenceCtr< **walker**, _Allocator > &__parents, const **walker** &__child, const _Tp &__x)
- **walker** split_front (const **walker** &__parent, const **walker** &__child, const search_node *&__x)
- **walker** split_front (const **walker** &__parent, const __SequenceCtr< **walker**, _Allocator > &__children, const search_node *&__x)
- **walker** split_front (const __SequenceCtr< **walker**, _Allocator > &__parents, const **walker** &__child, const search_node *&__x)
- **walker split_front** (const **walker** &__parent, const **walker** &__child, const _Tp &__x)
- **walker split_front** (const **walker** &__parent, const __SequenceCtr< **walker**, _Allocator > &__children, const _Tp &__x)
- **walker split_front** (const __SequenceCtr< **walker**, _Allocator > &__parents, const **walker** &__child, const _Tp &__x)
- void insert_subgraph (_Self &__subgraph, const **walker** &__parent, const **children_iterator** &__ch_it, const **walker** &__child, const **parents_iterator** &__pa_it)
- void insert_subgraph (_Self &__subgraph, const **walker** &__parent, const **walker** &__child, const container_insert_arg &__Itc, const container_insert_arg &__Itp)
- void insert_subgraph (_Self &__subgraph, const __SequenceCtr1< **walker**, _Allocator1 > &__parents, const __SequenceCtr2< **walker**, _Allocator2 > &__children)
- void insert_back_subgraph (_Self &__subgraph, const **walker** &__parent, const **walker** &__child)
- void insert_front_subgraph (_Self &__subgraph, const **walker** &__parent, const **walker** &__child)
- void add_edge (const **walker** &__parent, const **children_iterator** &__ch_it, const **walker** &__child, const **parents_iterator** &__pa_it)
- void add_edge (const **edge** &__edge, const container_insert_arg &__Itc, const container_insert_arg &__Itp)
- void add_edge (const **walker** &__parent, const **walker** &__child, const container_insert_arg &__Itc, const container_insert_arg &__Itp)
- void add_edge_back (const **walker** &__parent, const **walker** &__child)
- void add_edge_front (const **walker** &__parent, const **walker** &__child)
- **allocator_type** get_allocator () const
- **walker** ground ()
- **const_walker** ground () const

- **walker** sky ()
- **const_walker** sky () const
- **children_iterator** root_begin ()
- **children_iterator** root_end ()
- **parents_iterator** leaf_begin ()
- **parents_iterator** leaf_end ()
- bool empty () const
- **size_type** size () const
- **size_type** max_size () const
- void swap (_Self &__x)
- **walker** insert_node_in_graph (_Node *__n, const **walker** &__parent, const **walker** &__child, const container_insert_arg &__Itc, const container_insert_arg &__Itp)
- **walker** insert_node_in_graph (_Node *__node, const __SequenceCtr1< **walker**, _Allocator1 > &__parents, const __SequenceCtr2< **walker**, _Allocator2 > &__children)
- **walker** insert_node_in_graph (_Node *__node, const **walker** &__parent, const container_insert_arg &__pref, const __SequenceCtr< **walker**, _Allocator > &__children)
- **walker** insert_node_in_graph (_Node *__node, const __SequenceCtr< **walker**, _Allocator > &__parents, const **walker** &__child, const container_insert_arg &__cref)
- **walker** insert_in_graph (const search_node *&__x, const **walker** &__parent, const **walker** &__child, const container_insert_arg &__Itc, const container_insert_arg &__Itp)
- **walker** insert_in_graph (const **walker** &__parent, const **walker** &__child, const container_insert_arg &__Itc, const container_insert_arg &__Itp)
- **walker** insert_in_graph (const search_node *&__x, const __SequenceCtr1< **walker**, _Allocator1 > &__parents, const __SequenceCtr2< **walker**, _Allocator2 > &__children)
- **walker** insert_in_graph (const __SequenceCtr1< **walker**, _Allocator1 > &__parents, const __SequenceCtr2< **walker**, _Allocator2 > &__children)
- **walker** insert_in_graph (const search_node *&__x, const **walker** &__parent, const container_insert_arg &__pref, const __SequenceCtr< **walker**, _Allocator > &__children)
- **walker** insert_in_graph (const **walker** &__parent, const container_insert_arg &__pref, const __SequenceCtr< **walker**, _Allocator > &__children)
- **walker** insert_in_graph (const search_node *&__x, const __SequenceCtr< **walker**, _Allocator > &__parents, const **walker** &__child, const container_insert_arg &__cref)
- **walker** insert_in_graph (const __SequenceCtr< **walker**, _Allocator > &__parents, const **walker** &__child, const container_insert_arg &__cref)
- **walker insert_in_graph** (const _Tp &__x, const **walker** &__parent, const **walker** &__child, const container_insert_arg &__Itc, const container_insert_arg &__Itp)
- **walker insert_in_graph** (const _Tp &__x, const __SequenceCtr1< **walker**, _Allocator1 > &__parents, const __SequenceCtr2< **walker**, _Allocator2 > &__children)
- **walker insert_in_graph** (const _Tp &__x, const **walker** &__parent, const container_insert_arg &__pref, const __SequenceCtr< **walker**, _Allocator > &__children)
- **walker insert_in_graph** (const _Tp &__x, const __SequenceCtr< **walker**, _Allocator > &__parents, const **walker** &__child, const container_insert_arg &__cref)
- void replace_edge_to_child (const **walker** &__parent, const **walker** &__child_old, const **walker** &__child_new)
- void replace_edge_to_parent (const **walker** &__parent_old, const **walker** &__parent_new, const **walker** &__child)
- void remove_edge (const **edge** &__edge)
- void remove_edge (const **walker** &__parent, const **walker** &__child)
- void remove_edge_and_deattach (const **walker** &__parent, const **walker** &__child)
- void sort_child_edges (**walker** __position, **children_iterator** first, **children_iterator** last, Compare comp)

- void sort_child_edges (**walker** __position, Compare comp)
- void sort_parent_edges (**walker** __position, **parents_iterator** first, **parents_iterator** last, Compare comp)
- void sort_parent_edges (**walker** __position, Compare comp)
- **walker** insert_node (_Node *_node, const **walker** &__position, const container_insert_arg &__It)
- **walker** insert_node (const search_node *&__x, const **walker** &__position, const container_insert_arg &__It)
- **walker** insert_node (const **walker** &__position, const container_insert_arg &__It)
- **walker** **insert_node** (const _Tp &__x, const **walker** &__position, const container_insert_arg &__It)
- **walker** insert_node_before (_Node *_node, const **walker** &__position, const container_insert_arg &__It)
- void insert_node_before (const search_node *&__x, const **walker** &__position, const container_insert_arg &__It)
- void insert_node_before (const **walker** &__position, const container_insert_arg &__It)
- void **insert_node_before** (const _Tp &__x, const **walker** &__position, const container_insert_arg &__It)
- void merge (const **walker** &__position, const **walker** &__second, bool merge_parent_edges=true, bool merge_child_edges=true)
- void erase (const **walker** &__position)
- void clear_erased_part (**erased_part** &_ep)
- **erased_part** erase_maximal_subgraph (const **walker** &__position)
- **erased_part** erase_maximal_subgraph (const __SequenceCtr< **walker**, _Allocator > &__positions)
- **erased_part** erase_minimal_subgraph (const **walker** &__position)
- **erased_part** erase_minimal_subgraph (const __SequenceCtr< **walker**, _Allocator > &__positions)
- **erased_part** erase_maximal_pregraph (const **walker** &__position)
- **erased_part** erase_maximal_pregraph (const __SequenceCtr< **walker**, _Allocator > &__positions)
- **erased_part** erase_minimal_pregraph (const **walker** &__position)
- **erased_part** erase_minimal_pregraph (const __SequenceCtr< **walker**, _Allocator > &__positions)
- bool erase_child (const **walker** &__position, const **children_iterator** &__It)
- bool erase_parent (const **walker** &__position, const **parents_iterator** &__It)
- void clear_children ()
- void clear_parents ()
- void add_all_children (_Output_Iterator fi, **_DG_node**< search_node *, _SequenceCtr< void *, _PtrAlloc >, _SequenceCtr< void *, _PtrAlloc >::**iterator** > *_parent)
- void **add_all_children** (_Output_Iterator fi, **_DG_node**< _Tp, _SequenceCtr< void *, _PtrAlloc >, _SequenceCtr< void *, _PtrAlloc >::**iterator** > *_parent)
- void **add_all_children** (_Output_Iterator fi, **_DG_node**< _Tp, _Ctr, _Iterator > *_parent)
- void add_all_parents (_Output_Iterator fi, **_DG_node**< search_node *, _SequenceCtr< void *, _PtrAlloc >, _SequenceCtr< void *, _PtrAlloc >::**iterator** > *_child)
- void **add_all_parents** (_Output_Iterator fi, **_DG_node**< _Tp, _SequenceCtr< void *, _PtrAlloc >, _SequenceCtr< void *, _PtrAlloc >::**iterator** > *_child)
- void **add_all_parents** (_Output_Iterator fi, **_DG_node**< _Tp, _Ctr, _Iterator > *_child)
- _Node * _C_create_node (const search_node *&__x)
- _Node * _C_create_node ()
- _Node * **_C_create_node** (const _Tp &__x)
- void clear_graph (**_DG_node**< search_node *, _SequenceCtr< void *, _PtrAlloc >, _SequenceCtr< void *, _PtrAlloc >::**iterator** > *_node)
- void **clear_graph** (**_DG_node**< _Tp, _SequenceCtr< void *, _PtrAlloc >, _SequenceCtr< void *, _PtrAlloc >::**iterator** > *_node)
- void **clear_graph** (**_DG_node**< _Tp, _Ctr, _Iterator > *_node)

- _DG_node< search_node *, _SequenceCtr< void *, _PtrAlloc >, _SequenceCtr< void *, _PtrAlloc >::**iterator** > * _C_get_node ()
- void _C_put_node (_DG_node< search_node *, _SequenceCtr< void *, _PtrAlloc >, _SequenceCtr< void *, _PtrAlloc >::**iterator** > *__p)
- void **_C_put_node** (_DG_node< _Tp, _SequenceCtr< void *, _PtrAlloc >, _SequenceCtr< void *, _PtrAlloc >::**iterator** > *__p)
- void **_C_put_node** (_DG_node< _Tp, _Ctr, _Iterator > *__p)

**Public Attributes**

- std::list< **walker** > focus
- std::list< **const_walker** > inspector
- search_node * root
- search_inspector inspector_for_root
- ptr< search_node > ptr_ground
- ptr< model > ptr_root_model
- list< search_node * > search_nodes_to_deallocate
- list< delta_id > db_deltas_to_remove
- gptr< vdbl::database > * __dbase
- **vdbl::userid** __dbuser
- _Base::walker walker
- _Base::const_walker const_walker
- _Base::children_iterator children_iterator
- _Base::parents_iterator parents_iterator
- _Base::erased_part erased_part
- _SequenceCtr< void *, _PtrAlloc > container_type
- search_node * value_type
- **_DG_iterator**< search_node *, search_node *&, search_node **, **container_type**, **children_iterator** > iterator
- **_DG_iterator**< search_node *, const search_node *&, const search_node **, **container_type**, **children_iterator** > const_iterator
- std::reverse_iterator< **const_iterator** > const_reverse_iterator
- std::reverse_iterator< **iterator** > reverse_iterator
- std::pair< **walker**, **walker** > edge
- std::pair< **edge**, bool > enhanced_edge
- _Base::allocator_type allocator_type
- _DG_node< search_node *, _SequenceCtr< void *, _PtrAlloc >, _SequenceCtr< void *, _PtrAlloc >::**iterator** > * _C_ground
- _DG_node< search_node *, _SequenceCtr< void *, _PtrAlloc >, _SequenceCtr< void *, _PtrAlloc >::**iterator** > * _C_sky
- int _C_mark
- _Node node_type
- **value_type** * pointer
- const **value_type** * const_pointer
- **value_type** & reference
- const **value_type** & const_reference
- size_t size_type
- ptrdiff_t difference_type

**Friends**

- bool operator==__VGTL_NULL_TMPL_ARGS (const __**DG** &__x, const __**DG** &__y)

### 6.94.1 Constructor & Destructor Documentation

#### 6.94.1.1 search_graph::search_graph (model & *root_model*, gptr< vdbl::database > & *_db*) [inline]

Definition at line 90 of file search_graph.h.

#### 6.94.1.2 search_graph::search_graph (model & *root_model*, gptr< vdbl::database > & *_db*, const std::vector< annotation > & *_a*) [inline]

Definition at line 104 of file search_graph.h.

#### 6.94.1.3 search_graph::~search_graph () [inline]

Definition at line 119 of file search_graph.h.

### 6.94.2 Member Function Documentation

#### 6.94.2.1 _Node * dag< search_node *, _SequenceCtr, _PtrAlloc, _Alloc >::_C_create_node () [inherited]

Reimplemented from **dgraph**< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >.

#### 6.94.2.2 _Node * dag< search_node *, _SequenceCtr, _PtrAlloc, _Alloc >::_C_create_node (const search_node * & __*x*) [inherited]

#### 6.94.2.3 _DG_node< search_node *, _SequenceCtr< void *, _PtrAlloc >, _SequenceCtr< void *, _PtrAlloc >::iterator > * dag< search_node *, _SequenceCtr, _PtrAlloc, _Alloc >::_C_get_node () [inherited]

Reimplemented from **dgraph**< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >.

#### 6.94.2.4 void dag< search_node *, _SequenceCtr, _PtrAlloc, _Alloc >::_C_put_node (_DG_node< search_node *, _SequenceCtr< void *, _PtrAlloc >, _SequenceCtr< void *, _PtrAlloc >::iterator > * __*p*) [inherited]

#### 6.94.2.5 void dag< search_node *, _SequenceCtr, _PtrAlloc, _Alloc >::add_all_children (_Output_Iterator *fi*, _DG_node< search_node *, _SequenceCtr< void *, _PtrAlloc >, _SequenceCtr< void *, _PtrAlloc >::iterator > * *_parent*) [inherited]

#### 6.94.2.6 void dag< search_node *, _SequenceCtr, _PtrAlloc, _Alloc >::add_all_parents (_Output_Iterator *fi*, _DG_node< search_node *, _SequenceCtr< void *, _PtrAlloc >, _SequenceCtr< void *, _PtrAlloc >::iterator > * *_child*) [inherited]

**6.94.2.7 void dag< search_node *, _SequenceCtr, _PtrAlloc, _Alloc >::add_edge (const walker & __parent, const walker & __child, const container_insert_arg & __Itc, const container_insert_arg & __Itp)** [inherited]

Reimplemented from **dgraph< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >**.

**6.94.2.8 void dag< search_node *, _SequenceCtr, _PtrAlloc, _Alloc >::add_edge (const edge & __edge, const container_insert_arg & __Itc, const container_insert_arg & __Itp)** [inherited]

Reimplemented from **dgraph< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >**.

**6.94.2.9 void dag< search_node *, _SequenceCtr, _PtrAlloc, _Alloc >::add_edge (const walker & __parent, const children_iterator & __ch_it, const walker & __child, const parents_iterator & __pa_it)** [inherited]

Reimplemented from **dgraph< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >**.

**6.94.2.10 void dag< search_node *, _SequenceCtr, _PtrAlloc, _Alloc >::add_edge_back (const walker & __parent, const walker & __child)** [inherited]

Reimplemented from **dgraph< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >**.

**6.94.2.11 void dag< search_node *, _SequenceCtr, _PtrAlloc, _Alloc >::add_edge_front (const walker & __parent, const walker & __child)** [inherited]

Reimplemented from **dgraph< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >**.

**6.94.2.12 walker dag< search_node *, _SequenceCtr, _PtrAlloc, _Alloc >::between (const __SequenceCtr< walker, _Allocator > & __parents, const walker & __child, const parents_iterator & __pit, const search_node * & __x)** [inherited]

**6.94.2.13 walker dag< search_node *, _SequenceCtr, _PtrAlloc, _Alloc >::between (const walker & __parent, const children_iterator & __cit, const __SequenceCtr< walker, _Allocator > & __children, const search_node * & __x)** [inherited]

**6.94.2.14 walker dag< search_node *, _SequenceCtr, _PtrAlloc, _Alloc >::between (const __SequenceCtr1< walker, _Allocator1 > & __parents, const __SequenceCtr2< walker, _Allocator2 > & __children, const search_node * & __x)** [inherited]

**6.94.2.15 walker dag< search_node *, _SequenceCtr, _PtrAlloc, _Alloc >::between (const walker & __parent, const children_iterator & __cit, const walker & __child, const parents_iterator & __pit, const search_node * & __x)** [inherited]

**6.94.2.16 walker dag< search_node *, _SequenceCtr, _PtrAlloc, _Alloc >::between_back (const __SequenceCtr< walker, _Allocator > & __parents, const walker & __child, const search_node * & __x)** [inherited]

**6.94.2.17 walker dag< search_node *, _SequenceCtr, _PtrAlloc, _Alloc >::between_back (const walker & __parent, const __SequenceCtr< walker, _Allocator > & __children, const search_node * & __x)** [inherited]

**6.94.2.18 walker dag< search_node *, _SequenceCtr, _PtrAlloc, _Alloc >::between_back (const walker & __parent, const walker & __child, const search_node * & __x)** [inherited]

**6.94.2.19 walker dag< search_node *, _SequenceCtr, _PtrAlloc, _Alloc >::between_front (const __SequenceCtr< walker, _Allocator > & __parents, const walker & __child, const search_node * & __x)** [inherited]

**6.94.2.20 walker dag< search_node *, _SequenceCtr, _PtrAlloc, _Alloc >::between_front (const walker & __parent, const __SequenceCtr< walker, _Allocator > & __children, const search_node * & __x)** [inherited]

**6.94.2.21 walker dag< search_node *, _SequenceCtr, _PtrAlloc, _Alloc >::between_front (const walker & __parent, const walker & __child, const search_node * & __x)** [inherited]

**6.94.2.22 bool dag< search_node *, _SequenceCtr, _PtrAlloc, _Alloc >::check_acyclicity (const walker & __parent, const walker & __child)** [inherited]

**6.94.2.23 search_inspector search_graph::child (search_inspector & *n*, unsigned int *i*)** [inline]

Definition at line 157 of file search_graph.h.

**6.94.2.24 void dag< search_node *, _SequenceCtr, _PtrAlloc, _Alloc >::clear ()** [inherited]

Reimplemented from **dgraph< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >**.

**6.94.2.25 void dag< search_node *, _SequenceCtr, _PtrAlloc, _Alloc >::clear_children ()** [inherited]

Reimplemented from **dgraph< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >**.

**6.94.2.26 void dag< search_node *, _SequenceCtr, _PtrAlloc, _Alloc >::clear_erased_part (erased_part & __ep)** [inherited]

Reimplemented from **dgraph< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >**.

**6.94.2.27 void dag< search_node *, _SequenceCtr, _PtrAlloc, _Alloc >::clear_graph (_DG_node< search_node *, _SequenceCtr< void *, _PtrAlloc >, _SequenceCtr< void *, _PtrAlloc >::iterator > * __node)** [inherited]

**6.94.2.28 void dag< search_node *, _SequenceCtr, _PtrAlloc, _Alloc >::clear_parents ()** [inherited]

Reimplemented from **dgraph< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >**.

**6.94.2.29 void search_graph::destroy_focus (const search_focus & __sf)**

Definition at line 53 of file search_graph.cc.

#### 6.94.2.30 void search_graph::destroy_inspector (const search_inspector & *inspector_to_destroy*)

Definition at line 87 of file search_graph.cc.

#### 6.94.2.31 bool dag< search_node *, _SequenceCtr, _PtrAlloc, _Alloc >::empty () [inherited]

Reimplemented from **dgraph**< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >.

#### 6.94.2.32 void dag< search_node *, _SequenceCtr, _PtrAlloc, _Alloc >::erase (const walker & __*position*) [inherited]

Reimplemented from **dgraph**< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >.

#### 6.94.2.33 bool dag< search_node *, _SequenceCtr, _PtrAlloc, _Alloc >::erase_child (const walker & __*position*, const children_iterator & __*It*) [inherited]

Reimplemented from **dgraph**< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >.

#### 6.94.2.34 erased_part dag< search_node *, _SequenceCtr, _PtrAlloc, _Alloc >::erase_maximal_pregraph (const __SequenceCtr< walker, _Allocator > & __*positions*) [inherited]

Reimplemented from **dgraph**< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >.

#### 6.94.2.35 erased_part dag< search_node *, _SequenceCtr, _PtrAlloc, _Alloc >::erase_maximal_pregraph (const walker & __*position*) [inherited]

Reimplemented from **dgraph**< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >.

#### 6.94.2.36 erased_part dag< search_node *, _SequenceCtr, _PtrAlloc, _Alloc >::erase_maximal_subgraph (const __SequenceCtr< walker, _Allocator > & __*positions*) [inherited]

Reimplemented from **dgraph**< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >.

#### 6.94.2.37 erased_part dag< search_node *, _SequenceCtr, _PtrAlloc, _Alloc >::erase_maximal_subgraph (const walker & __*position*) [inherited]

Reimplemented from **dgraph**< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >.

#### 6.94.2.38 erased_part dag< search_node *, _SequenceCtr, _PtrAlloc, _Alloc >::erase_minimal_pregraph (const __SequenceCtr< walker, _Allocator > & __*positions*) [inherited]

Reimplemented from **dgraph**< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >.

#### 6.94.2.39 erased_part dag< search_node *, _SequenceCtr, _PtrAlloc, _Alloc >::erase_minimal_pregraph (const walker & __*position*) [inherited]

Reimplemented from **dgraph**< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >.

#### 6.94.2.40 erased_part dag< search_node *, _SequenceCtr, _PtrAlloc, _Alloc >::erase_minimal_subgraph (const __SequenceCtr< walker, _Allocator > & __*positions*) [inherited]

Reimplemented from **dgraph**< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >.

**6.94.2.41 erased_part dag< search_node *, _SequenceCtr, _PtrAlloc, _Alloc >::erase_minimal_subgraph (const walker & __position)** [inherited]

Reimplemented from **dgraph< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >**.

**6.94.2.42 bool dag< search_node *, _SequenceCtr, _PtrAlloc, _Alloc >::erase_parent (const walker & __position, const parents_iterator & __It)** [inherited]

Reimplemented from **dgraph< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >**.

**6.94.2.43 work_node search_graph::extract (const search_focus & *where*)**

Definition at line 113 of file search_graph.cc.

**6.94.2.44 work_node search_graph::extract (const search_inspector & *where*)**

Definition at line 98 of file search_graph.cc.

**6.94.2.45 allocator_type dag< search_node *, _SequenceCtr, _PtrAlloc, _Alloc >::get_allocator ()** [inherited]

Reimplemented from **dgraph< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >**.

**6.94.2.46 search_node_id search_graph::get_node_id ()** [inline]

Definition at line 134 of file search_graph.h.

**6.94.2.47 const_walker dag< search_node *, _SequenceCtr, _PtrAlloc, _Alloc >::ground ()** [inherited]

Reimplemented from **dgraph< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >**.

**6.94.2.48 walker dag< search_node *, _SequenceCtr, _PtrAlloc, _Alloc >::ground ()** [inherited]

Reimplemented from **dgraph< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >**.

**6.94.2.49 search_inspector & search_graph::insert (const search_focus & *where*, const search_node & *what*)**

Definition at line 128 of file search_graph.cc.

**6.94.2.50 void dag< search_node *, _SequenceCtr, _PtrAlloc, _Alloc >::insert_back_subgraph (_Self & __subgraph, const walker & __parent, const walker & __child)** [inherited]

Reimplemented from **dgraph< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >**.

**6.94.2.51 void dag< search_node *, _SequenceCtr, _PtrAlloc, _Alloc >::insert_front_subgraph (_Self & __subgraph, const walker & __parent, const walker & __child)** [inherited]

Reimplemented from **dgraph< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >**.

**6.94.2.52 walker dag< search_node *, _SequenceCtr, _PtrAlloc, _Alloc >::insert_in_graph (const __SequenceCtr< walker, _Allocator > & __*parents*, const walker & __*child*, const container_insert_arg & __*cref*)** [inherited]

Reimplemented from **dgraph< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >**.

**6.94.2.53 walker dag< search_node *, _SequenceCtr, _PtrAlloc, _Alloc >::insert_in_graph (const search_node * & __*x*, const __SequenceCtr< walker, _Allocator > & __*parents*, const walker & __*child*, const container_insert_arg & __*cref*)** [inherited]

**6.94.2.54 walker dag< search_node *, _SequenceCtr, _PtrAlloc, _Alloc >::insert_in_graph (const walker & __*parent*, const container_insert_arg & __*pref*, const __SequenceCtr< walker, _Allocator > & __*children*)** [inherited]

Reimplemented from **dgraph< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >**.

**6.94.2.55 walker dag< search_node *, _SequenceCtr, _PtrAlloc, _Alloc >::insert_in_graph (const search_node * & __*x*, const walker & __*parent*, const container_insert_arg & __*pref*, const __SequenceCtr< walker, _Allocator > & __*children*)** [inherited]

**6.94.2.56 walker dag< search_node *, _SequenceCtr, _PtrAlloc, _Alloc >::insert_in_graph (const __SequenceCtr1< walker, _Allocator1 > & __*parents*, const __SequenceCtr2< walker, _Allocator2 > & __*children*)** [inherited]

Reimplemented from **dgraph< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >**.

**6.94.2.57 walker dag< search_node *, _SequenceCtr, _PtrAlloc, _Alloc >::insert_in_graph (const search_node * & __*x*, const __SequenceCtr1< walker, _Allocator1 > & __*parents*, const __SequenceCtr2< walker, _Allocator2 > & __*children*)** [inherited]

**6.94.2.58 walker dag< search_node *, _SequenceCtr, _PtrAlloc, _Alloc >::insert_in_graph (const walker & __*parent*, const walker & __*child*, const container_insert_arg & __*Itc*, const container_insert_arg & __*Itp*)** [inherited]

Reimplemented from **dgraph< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >**.

**6.94.2.59 walker dag< search_node *, _SequenceCtr, _PtrAlloc, _Alloc >::insert_in_graph (const search_node * & __*x*, const walker & __*parent*, const walker & __*child*, const container_insert_arg & __*Itc*, const container_insert_arg & __*Itp*)** [inherited]

**6.94.2.60 walker dag< search_node *, _SequenceCtr, _PtrAlloc, _Alloc >::insert_node (const walker & __*position*, const container_insert_arg & __*It*)** [inherited]

Reimplemented from **dgraph< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >**.

**6.94.2.61 walker dag< search_node *, _SequenceCtr, _PtrAlloc, _Alloc >::insert_node (const search_node * & __*x*, const walker & __*position*, const container_insert_arg & __*It*)** [inherited]

**6.94.2.62 walker dag< search_node *, _SequenceCtr, _PtrAlloc, _Alloc >::insert_node (_Node * __node, const walker & __position, const container_insert_arg & __It)** [inherited]

Reimplemented from **dgraph< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >**.

**6.94.2.63 void dag< search_node *, _SequenceCtr, _PtrAlloc, _Alloc >::insert_node_before (const walker & __position, const container_insert_arg & __It)** [inherited]

Reimplemented from **dgraph< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >**.

**6.94.2.64 void dag< search_node *, _SequenceCtr, _PtrAlloc, _Alloc >::insert_node_before (const search_node * & __x, const walker & __position, const container_insert_arg & __It)** [inherited]

**6.94.2.65 walker dag< search_node *, _SequenceCtr, _PtrAlloc, _Alloc >::insert_node_before (_Node * __node, const walker & __position, const container_insert_arg & __It)** [inherited]

Reimplemented from **dgraph< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >**.

**6.94.2.66 walker dag< search_node *, _SequenceCtr, _PtrAlloc, _Alloc >::insert_node_in_graph (_Node * __node, const __SequenceCtr< walker, _Allocator > & __parents, const walker & __child, const container_insert_arg & __cref)** [inherited]

Reimplemented from **dgraph< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >**.

**6.94.2.67 walker dag< search_node *, _SequenceCtr, _PtrAlloc, _Alloc >::insert_node_in_graph (_Node * __node, const walker & __parent, const container_insert_arg & __pref, const __SequenceCtr< walker, _Allocator > & __children)** [inherited]

Reimplemented from **dgraph< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >**.

**6.94.2.68 walker dag< search_node *, _SequenceCtr, _PtrAlloc, _Alloc >::insert_node_in_graph (_Node * __node, const __SequenceCtr1< walker, _Allocator1 > & __parents, const __SequenceCtr2< walker, _Allocator2 > & __children)** [inherited]

Reimplemented from **dgraph< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >**.

**6.94.2.69 walker dag< search_node *, _SequenceCtr, _PtrAlloc, _Alloc >::insert_node_in_graph (_Node * __n, const walker & __parent, const walker & __child, const container_insert_arg & __Itc, const container_insert_arg & __Itp)** [inherited]

Reimplemented from **dgraph< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >**.

**6.94.2.70 void dag< search_node *, _SequenceCtr, _PtrAlloc, _Alloc >::insert_subgraph (_Self & __subgraph, const __SequenceCtr1< walker, _Allocator1 > & __parents, const __SequenceCtr2< walker, _Allocator2 > & __children)** [inherited]

Reimplemented from **dgraph< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >**.

**6.94.2.71 void dag< search_node *, _SequenceCtr, _PtrAlloc, _Alloc >::insert_subgraph (_Self & __subgraph, const walker & __parent, const walker & __child, const container_insert_arg & __Itc, const container_insert_arg & __Itp)** [inherited]

Reimplemented from **dgraph**< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >.

**6.94.2.72 void dag< search_node *, _SequenceCtr, _PtrAlloc, _Alloc >::insert_subgraph (_Self & __subgraph, const walker & __parent, const children_iterator & __ch_it, const walker & __child, const parents_iterator & __pa_it)** `[inherited]`

Reimplemented from **dgraph**< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >.

**6.94.2.73 parents_iterator dag< search_node *, _SequenceCtr, _PtrAlloc, _Alloc >::leaf_begin ()** `[inherited]`

Reimplemented from **dgraph**< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >.

**6.94.2.74 parents_iterator dag< search_node *, _SequenceCtr, _PtrAlloc, _Alloc >::leaf_end ()** `[inherited]`

Reimplemented from **dgraph**< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >.

**6.94.2.75 size_type dag< search_node *, _SequenceCtr, _PtrAlloc, _Alloc >::max_size ()** `[inherited]`

Reimplemented from **dgraph**< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >.

**6.94.2.76 void dag< search_node *, _SequenceCtr, _PtrAlloc, _Alloc >::merge (const walker & __position, const walker & __second, bool *merge_parent_edges* = true, bool *merge_child_edges* = true)** `[inherited]`

Reimplemented from **dgraph**< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >.

**6.94.2.77 search_focus & search_graph::new_focus (const search_inspector & _si)**

Definition at line 41 of file search_graph.cc.

**6.94.2.78 search_inspector& search_graph::new_inspector ()** `[inline]`

Definition at line 142 of file search_graph.h.

**6.94.2.79 search_inspector & search_graph::new_inspector (const search_inspector & *inspector_to_add*)**

Definition at line 76 of file search_graph.cc.

**6.94.2.80 search_inspector search_graph::parent (search_inspector & *n*, unsigned int *i*)** `[inline]`

Definition at line 165 of file search_graph.h.

**6.94.2.81 search_focus & search_graph::promote (search_focus & _sf)**

Definition at line 279 of file search_graph.cc.

#### 6.94.2.82 void search_graph::remove (search_focus & *_sf*)

Definition at line 267 of file search_graph.cc.

#### 6.94.2.83 void dag< search_node *, _SequenceCtr, _PtrAlloc, _Alloc >::remove_edge (const walker & *__parent*, const walker & *__child*) [inherited]

Reimplemented from **dgraph**< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >.

#### 6.94.2.84 void dag< search_node *, _SequenceCtr, _PtrAlloc, _Alloc >::remove_edge (const edge & *__edge*) [inherited]

Reimplemented from **dgraph**< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >.

#### 6.94.2.85 void dag< search_node *, _SequenceCtr, _PtrAlloc, _Alloc >::remove_edge_and_deattach (const walker & *__parent*, const walker & *__child*) [inherited]

Reimplemented from **dgraph**< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >.

#### 6.94.2.86 void search_graph::remove_upwards (search_focus & *_sf*, bool *relaxation_kills*)

Definition at line 370 of file search_graph.cc.

#### 6.94.2.87 search_inspector & search_graph::replace (search_focus & *where*, const search_node & *what*)

Definition at line 142 of file search_graph.cc.

#### 6.94.2.88 void dag< search_node *, _SequenceCtr, _PtrAlloc, _Alloc >::replace_edge_to_child (const walker & *__parent*, const walker & *__child_old*, const walker & *__child_new*) [inherited]

Reimplemented from **dgraph**< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >.

#### 6.94.2.89 void dag< search_node *, _SequenceCtr, _PtrAlloc, _Alloc >::replace_edge_to_parent (const walker & *__parent_old*, const walker & *__parent_new*, const walker & *__child*) [inherited]

Reimplemented from **dgraph**< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >.

#### 6.94.2.90 children_iterator dag< search_node *, _SequenceCtr, _PtrAlloc, _Alloc >::root_begin () [inherited]

Reimplemented from **dgraph**< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >.

#### 6.94.2.91 children_iterator dag< search_node *, _SequenceCtr, _PtrAlloc, _Alloc >::root_end () [inherited]

Reimplemented from **dgraph**< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >.

#### 6.94.2.92 search_focus & search_graph::set_focus (search_focus & *_fc*, const search_inspector & *_si*)

Definition at line 62 of file search_graph.cc.

**6.94.2.93 size_type dag< search_node *, _SequenceCtr, _PtrAlloc, _Alloc >::size () [inherited]**

Reimplemented from **dgraph< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >**.

**6.94.2.94 const_walker dag< search_node *, _SequenceCtr, _PtrAlloc, _Alloc >::sky () [inherited]**

Reimplemented from **dgraph< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >**.

**6.94.2.95 walker dag< search_node *, _SequenceCtr, _PtrAlloc, _Alloc >::sky () [inherited]**

Reimplemented from **dgraph< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >**.

**6.94.2.96 void dag< search_node *, _SequenceCtr, _PtrAlloc, _Alloc >::sort_child_edges (walker __position, Compare *comp*) [inherited]**

Reimplemented from **dgraph< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >**.

**6.94.2.97 void dag< search_node *, _SequenceCtr, _PtrAlloc, _Alloc >::sort_child_edges (walker __position, children_iterator *first*, children_iterator *last*, Compare *comp*) [inherited]**

Reimplemented from **dgraph< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >**.

**6.94.2.98 void dag< search_node *, _SequenceCtr, _PtrAlloc, _Alloc >::sort_parent_edges (walker __position, Compare *comp*) [inherited]**

Reimplemented from **dgraph< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >**.

**6.94.2.99 void dag< search_node *, _SequenceCtr, _PtrAlloc, _Alloc >::sort_parent_edges (walker __position, parents_iterator *first*, parents_iterator *last*, Compare *comp*) [inherited]**

Reimplemented from **dgraph< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >**.

**6.94.2.100 walker dag< search_node *, _SequenceCtr, _PtrAlloc, _Alloc >::split (const __SequenceCtr< walker, _Allocator > & __parents, const walker & __child, const parents_iterator & __pr_it, const search_node * & __x) [inherited]**

**6.94.2.101 walker dag< search_node *, _SequenceCtr, _PtrAlloc, _Alloc >::split (const walker & __parent, const children_iterator & __ch_it, const __SequenceCtr< walker, _Allocator > & __children, const search_node * & __x) [inherited]**

**6.94.2.102 void dag< search_node *, _SequenceCtr, _PtrAlloc, _Alloc >::split (const __SequenceCtr1< walker, _Allocator1 > & __parents, const __SequenceCtr2< walker, _Allocator2 > & __children, const search_node * & __x) [inherited]**

**6.94.2.103 walker dag< search_node *, _SequenceCtr, _PtrAlloc, _Alloc >::split (const walker & __parent, const children_iterator & __ch_it, const walker & __child, const parents_iterator & __pa_it, const search_node * & __x) [inherited]**

**6.94.2.104 walker dag< search_node *, _SequenceCtr, _PtrAlloc, _Alloc >::split_back (const __SequenceCtr< walker, _Allocator > & __*parents*, const walker & __*child*, const search_node * & __*x*)** [inherited]

**6.94.2.105 walker dag< search_node *, _SequenceCtr, _PtrAlloc, _Alloc >::split_back (const walker & __*parent*, const __SequenceCtr< walker, _Allocator > & __*children*, const search_node * & __*x*)** [inherited]

**6.94.2.106 walker dag< search_node *, _SequenceCtr, _PtrAlloc, _Alloc >::split_back (const walker & __*parent*, const walker & __*child*, const search_node * & __*x*)** [inherited]

**6.94.2.107 walker dag< search_node *, _SequenceCtr, _PtrAlloc, _Alloc >::split_front (const __SequenceCtr< walker, _Allocator > & __*parents*, const walker & __*child*, const search_node * & __*x*)** [inherited]

**6.94.2.108 walker dag< search_node *, _SequenceCtr, _PtrAlloc, _Alloc >::split_front (const walker & __*parent*, const __SequenceCtr< walker, _Allocator > & __*children*, const search_node * & __*x*)** [inherited]

**6.94.2.109 walker dag< search_node *, _SequenceCtr, _PtrAlloc, _Alloc >::split_front (const walker & __*parent*, const walker & __*child*, const search_node * & __*x*)** [inherited]

**6.94.2.110 void dag< search_node *, _SequenceCtr, _PtrAlloc, _Alloc >::swap (_Self & __*x*)** [inherited]

Reimplemented from **dgraph< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >**.

### 6.94.3 Friends And Related Function Documentation

**6.94.3.1 bool operator==__VGTL_NULL_TMPL_ARGS (const __DG & __*x*, const __DG & __*y*)** [friend, inherited]

Reimplemented from **dgraph< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >**.

### 6.94.4 Member Data Documentation

**6.94.4.1 gptr<vdbl::database>* search_graph::__dbase**

Definition at line 81 of file search_graph.h.

**6.94.4.2 vdbl::userid search_graph::__dbuser**

Definition at line 82 of file search_graph.h.

**6.94.4.3 _DG_node< search_node *, _SequenceCtr< void *, _PtrAlloc >, _SequenceCtr< void *, _PtrAlloc >::iterator > * dag< search_node *, _SequenceCtr, _PtrAlloc, _Alloc >::_C_ground** [inherited]

Reimplemented from **dgraph< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >**.

#### 6.94.4.4 int dag< search_node *, _SequenceCtr, _PtrAlloc, _Alloc >::_C_mark [inherited]

Reimplemented from **dgraph< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >**.

#### 6.94.4.5 _DG_node< search_node *, _SequenceCtr< void *, _PtrAlloc >, _SequenceCtr< void *, _PtrAlloc >::iterator > * dag< search_node *, _SequenceCtr, _PtrAlloc, _Alloc >::_C_sky [inherited]

Reimplemented from **dgraph< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >**.

#### 6.94.4.6 _Base::allocator_type dag< search_node *, _SequenceCtr, _PtrAlloc, _Alloc >::allocator_type [inherited]

Reimplemented from **dgraph< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >**.

#### 6.94.4.7 _Base::children_iterator dag< search_node *, _SequenceCtr, _PtrAlloc, _Alloc >::children_iterator [inherited]

Reimplemented from **dgraph< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >**.

#### 6.94.4.8 _DG_iterator< search_node *, const search_node * &, const search_node * *, container_type, children_iterator > dag< search_node *, _SequenceCtr, _PtrAlloc, _Alloc >::const_iterator [inherited]

Reimplemented from **dgraph< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >**.

#### 6.94.4.9 const value_type * __DG< _Tp, _SequenceCtr< void *, _PtrAlloc >, _SequenceCtr< void *, _PtrAlloc >::iterator, _SequenceCtr< void *, _PtrAlloc >::iterator, _Alloc >::const_pointer [inherited]

#### 6.94.4.10 const value_type & __DG< _Tp, _SequenceCtr< void *, _PtrAlloc >, _SequenceCtr< void *, _PtrAlloc >::iterator, _SequenceCtr< void *, _PtrAlloc >::iterator, _Alloc >::const_reference [inherited]

#### 6.94.4.11 std::reverse_iterator< const_iterator > dag< search_node *, _SequenceCtr, _PtrAlloc, _Alloc >::const_reverse_iterator [inherited]

Reimplemented from **dgraph< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >**.

#### 6.94.4.12 _Base::const_walker dag< search_node *, _SequenceCtr, _PtrAlloc, _Alloc >::const_walker [inherited]

Reimplemented from **dgraph< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >**.

#### 6.94.4.13 _SequenceCtr< void *, _PtrAlloc > dag< search_node *, _SequenceCtr, _PtrAlloc, _Alloc >::container_type [inherited]

Reimplemented from **dgraph< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >**.

#### 6.94.4.14 list<delta_id> search_graph::db_deltas_to_remove

Definition at line 80 of file search_graph.h.

**6.94.4.15 ptrdiff_t __DG< _Tp, _SequenceCtr< void *, _PtrAlloc >, _SequenceCtr< void *, _PtrAlloc >::iterator, _SequenceCtr< void *, _PtrAlloc >::iterator, _Alloc >::difference_type** `[inherited]`

**6.94.4.16 std::pair< walker, walker > dag< search_node *, _SequenceCtr, _PtrAlloc, _Alloc >::edge** `[inherited]`

Reimplemented from **dgraph< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >**.

**6.94.4.17 std::pair< edge, bool > dag< search_node *, _SequenceCtr, _PtrAlloc, _Alloc >::enhanced_edge** `[inherited]`

Reimplemented from **dgraph< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >**.

**6.94.4.18 _Base::erased_part dag< search_node *, _SequenceCtr, _PtrAlloc, _Alloc >::erased_part** `[inherited]`

Reimplemented from **dgraph< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >**.

**6.94.4.19 std::list<walker> search_graph::focus**

Definition at line 72 of file search_graph.h.

**6.94.4.20 std::list<const_walker> search_graph::inspector**

Definition at line 73 of file search_graph.h.

**6.94.4.21 search_inspector search_graph::inspector_for_root**

Definition at line 76 of file search_graph.h.

**6.94.4.22 _DG_iterator< search_node *, search_node * &, search_node * *, container_type, children_iterator > dag< search_node *, _SequenceCtr, _PtrAlloc, _Alloc >::iterator** `[inherited]`

Reimplemented from **dgraph< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >**.

**6.94.4.23 _Node __DG< _Tp, _SequenceCtr< void *, _PtrAlloc >, _SequenceCtr< void *, _PtrAlloc >::iterator, _SequenceCtr< void *, _PtrAlloc >::iterator, _Alloc >::node_type** `[inherited]`

**6.94.4.24 _Base::parents_iterator dag< search_node *, _SequenceCtr, _PtrAlloc, _Alloc >::parents_iterator** `[inherited]`

Reimplemented from **dgraph< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >**.

**6.94.4.25 value_type * __DG< _Tp, _SequenceCtr< void *, _PtrAlloc >, _SequenceCtr< void *, _PtrAlloc >::iterator, _SequenceCtr< void *, _PtrAlloc >::iterator, _Alloc >::pointer** `[inherited]`

#### 6.94.4.26 ptr<search_node> search_graph::ptr_ground

Definition at line 77 of file search_graph.h.

#### 6.94.4.27 ptr<model> search_graph::ptr_root_model

Definition at line 78 of file search_graph.h.

#### 6.94.4.28 value_type & __DG< _Tp, _SequenceCtr< void *, _PtrAlloc >, _SequenceCtr< void *, _PtrAlloc >::iterator, _SequenceCtr< void *, _PtrAlloc >::iterator, _Alloc >::reference [inherited]

#### 6.94.4.29 std::reverse_iterator< iterator > dag< search_node *, _SequenceCtr, _PtrAlloc, _Alloc >::reverse_iterator [inherited]

Reimplemented from **dgraph< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >**.

#### 6.94.4.30 search_node* search_graph::root

Definition at line 75 of file search_graph.h.

#### 6.94.4.31 list<search_node*> search_graph::search_nodes_to_deallocate

Definition at line 79 of file search_graph.h.

#### 6.94.4.32 size_t __DG< _Tp, _SequenceCtr< void *, _PtrAlloc >, _SequenceCtr< void *, _PtrAlloc >::iterator, _SequenceCtr< void *, _PtrAlloc >::iterator, _Alloc >::size_type [inherited]

#### 6.94.4.33 search_node * dag< search_node *, _SequenceCtr, _PtrAlloc, _Alloc >::value_type [inherited]

Reimplemented from **dgraph< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >**.

#### 6.94.4.34 _Base::walker dag< search_node *, _SequenceCtr, _PtrAlloc, _Alloc >::walker [inherited]

Reimplemented from **dgraph< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >**.

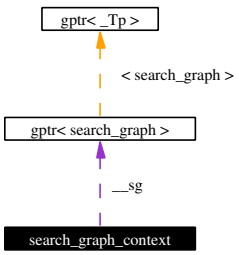
The documentation for this class was generated from the following files:

- search_graph.h
- search_graph.cc

## 6.95 search_graph_context Class Reference

`#include <sgraphctx.h>`

Collaboration diagram for search_graph_context:




### Public Methods

- search_graph_context ()
- search_graph_context (const gptr< search_graph > *_i)
- search_graph_context (const search_graph_context &_s)
- virtual ~search_graph_context ()
- search_graph_context & operator= (const search_graph_context &_s)
- const gptr< search_graph > * sg () const

#### 6.95.1 Constructor & Destructor Documentation

##### 6.95.1.1 search_graph_context::search_graph_context () [inline]

Definition at line 39 of file sgraphctx.h.

##### 6.95.1.2 search_graph_context::search_graph_context (const gptr< search_graph > * *_i*) [inline]

Definition at line 40 of file sgraphctx.h.

##### 6.95.1.3 search_graph_context::search_graph_context (const search_graph_context & *_s*) [inline]

Definition at line 41 of file sgraphctx.h.

##### 6.95.1.4 virtual search_graph_context::~search_graph_context () [inline, virtual]

Definition at line 42 of file sgraphctx.h.

#### 6.95.2 Member Function Documentation

##### 6.95.2.1 search_graph_context& search_graph_context::operator= (const search_graph_context & *_s*) [inline]

Definition at line 44 of file sgraphctx.h.

##### 6.95.2.2 const gptr<search_graph>* search_graph_context::sg () const [inline]

Definition at line 47 of file sgraphctx.h.

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

- sgraphctx.h

## 6.96 search_graph_exception Class Reference

`#include <search_graph.h>`

### Public Methods

- search_graph_exception (char const *m)
- virtual char const * what () const throw ()

### Public Attributes

- char const *const message

### 6.96.1 Constructor & Destructor Documentation

#### 6.96.1.1 search_graph_exception::search_graph_exception (char const * *m*) [inline]

Definition at line 54 of file search_graph.h.

### 6.96.2 Member Function Documentation

#### 6.96.2.1 virtual char const* search_graph_exception::what () const throw () [inline, virtual]

Definition at line 56 of file search_graph.h.

### 6.96.3 Member Data Documentation

#### 6.96.3.1 char const* const search_graph_exception::message

Definition at line 52 of file search_graph.h.

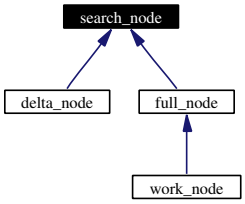
The documentation for this class was generated from the following file:

- search_graph.h

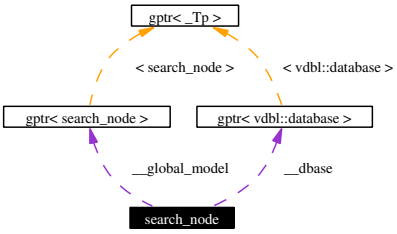
## 6.97 search_node Class Reference

`#include <search_node.h>`

Inheritance diagram for search_node:



Collaboration diagram for search_node:




### Public Methods

- virtual bool is_delta () const
- search_node (const search_node_id &_i, const **vdbl::userid** &_dui, gptr< search_node > &_gm, gptr< vdbl::database > &_db, search_node_relation __snr=snr_reduction)
- search_node (const search_node_id &_i, const **vdbl::userid** &_dui, gptr< search_node > *_gm, gptr< vdbl::database > &_db, search_node_relation __snr=snr_reduction)
- search_node (const search_node &__sn)
- search_node (const search_node_id &_i, const **vdbl::userid** &_dui, gptr< vdbl::database > &_db, search_node_relation __snr=snr_root)
- virtual ~search_node ()
- **vdbl::userid** get_dbuserid () const
- gptr< search_node > * global_model () const
- gptr< vdbl::database > * database () const
- search_node_id get_id () const
- void keep (const annotation &_an)
- void keep (const std::vector< annotation > &_anv)
- void unkeep (const annotation &_an)
- void unkeep (const std::vector< annotation > &_anv)

### Protected Methods

- search_node_id node_id () const
- search_node_relation parent_relation () const

### Protected Attributes

- gptr< search_node > * __global_model
- gptr< vdbl::database > * __dbase
- **vdbl::userid** _dbuser
- search_node_relation _snr
- search_node_id _id
- std::vector< annotation > _keep

### Friends

- class search_graph

### 6.97.1 Constructor & Destructor Documentation

#### 6.97.1.1 search_node::search_node (const search_node_id & _i, const vdbl::userid & _dui, gptr< search_node > & _gm, gptr< vdbl::database > & _db, search_node_relation __snr = snr_reduction) [inline]

Definition at line 102 of file search_node.h.

#### 6.97.1.2 search_node::search_node (const search_node_id & _i, const vdbl::userid & _dui, gptr< search_node > * _gm, gptr< vdbl::database > & _db, search_node_relation __snr = snr_reduction) [inline]

Definition at line 109 of file search_node.h.

#### 6.97.1.3 search_node::search_node (const search_node & __sn) [inline]

Definition at line 115 of file search_node.h.

#### 6.97.1.4 search_node::search_node (const search_node_id & _i, const vdbl::userid & _dui, gptr< vdbl::database > & _db, search_node_relation __snr = snr_root) [inline]

Definition at line 119 of file search_node.h.

#### 6.97.1.5 virtual search_node::~search_node () [inline, virtual]

Definition at line 131 of file search_node.h.

### 6.97.2 Member Function Documentation

#### 6.97.2.1 gptr<vdbl::database>* search_node::database () const [inline]

Definition at line 147 of file search_node.h.

#### 6.97.2.2 vdbl::userid search_node::get_dbuserid () const [inline]

Definition at line 143 of file search_node.h.

#### 6.97.2.3 search_node_id search_node::get_id () const [inline]

Definition at line 149 of file search_node.h.

#### 6.97.2.4 gptr<search_node>* search_node::global_model () const [inline]

Definition at line 145 of file search_node.h.

#### 6.97.2.5 virtual bool search_node::is_delta () const [inline, virtual]

Reimplemented in delta_node, and full_node.

Definition at line 100 of file search_node.h.

#### 6.97.2.6 void search_node::keep (const std::vector< annotation > & _anv) [inline]

Definition at line 153 of file search_node.h.

#### 6.97.2.7 void search_node::keep (const annotation & *_an*) [inline]

Definition at line 151 of file search_node.h.

#### 6.97.2.8 search_node_id search_node::node_id () const [inline, protected]

Definition at line 96 of file search_node.h.

#### 6.97.2.9 search_node_relation search_node::parent_relation () const [inline, protected]

Definition at line 97 of file search_node.h.

#### 6.97.2.10 void search_node::unkeep (const std::vector< annotation > & *_anv*) [inline]

Definition at line 167 of file search_node.h.

#### 6.97.2.11 void search_node::unkeep (const annotation & *_an*) [inline]

Definition at line 156 of file search_node.h.

### 6.97.3 Friends And Related Function Documentation

#### 6.97.3.1 friend class search_graph [friend]

Definition at line 174 of file search_node.h.

### 6.97.4 Member Data Documentation

#### 6.97.4.1 gptr<vdbl::database>* search_node::__dbase [protected]

Definition at line 89 of file search_node.h.

#### 6.97.4.2 gptr<search_node>* search_node::__global_model [protected]

Definition at line 88 of file search_node.h.

#### 6.97.4.3 vdbl::userid search_node::_dbuser [protected]

Definition at line 90 of file search_node.h.

#### 6.97.4.4 search_node_id search_node::_id [protected]

Definition at line 92 of file search_node.h.

#### 6.97.4.5 std::vector<annotation> search_node::_keep [protected]

Definition at line 93 of file search_node.h.

#### 6.97.4.6 search_node_relation search_node::_snr [protected]

Definition at line 91 of file search_node.h.

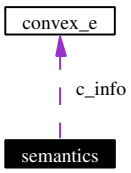
The documentation for this class was generated from the following file:

- search_node.h

## 6.98 semantics Class Reference

`#include <semantics.h>`

Collaboration diagram for semantics:




### Public Methods

- semantics ()
- semantics (const semantics &__s)
- ~semantics ()
- void read (char *c)
- void merge (const semantics &__s)
- const convex_e & convexity () const
- void convexity (const convex_e &__c)
- const tristate & separability () const
- void separability (const tristate &__c)
- const activity_descr & activity () const
- void activity (const activity_descr &__c)
- bool kj_flag () const
- void kj_flag (bool __c)
- bool integer_flag () const
- void integer_flag (bool __c)
- bool hard_flag () const
- void hard_flag (bool __c)
- bool is_at_any_bound () const
- void is_at_any_bound (bool __c)
- const type_annotation & type () const
- void type (const type_annotation &__c)
- bool redundancy () const
- bool inactive_hi () const
- bool inactive_lo () const
- bool inactive () const

### Public Attributes

- struct {
  convex_e c_info
  activity_descr act
  tristate separable
  tristate active
  bool is_at_any_bound
  } property_flags
- struct {
  bool kj
  bool integer
  type_annotation type
  bool hard
  } annotation_flags
- struct {
  tristate has_0chnbase
  } info_flags
- unsigned int _0chnbase
- int addinfo
- int degree
- int dim
- int stage

### Friends

- std::ostream & operator<< (std::ostream &o, const semantics &__s)

## 6.98.1 Constructor & Destructor Documentation

### 6.98.1.1 semantics::semantics () [inline]

Definition at line 222 of file semantics.h.

### 6.98.1.2 semantics::semantics (const semantics & __s) [inline]

Definition at line 244 of file semantics.h.

### 6.98.1.3 semantics::~semantics () [inline]

Definition at line 251 of file semantics.h.

## 6.98.2 Member Function Documentation

### 6.98.2.1 void semantics::activity (const activity_descr & __c) [inline]

Definition at line 336 of file semantics.h.

### 6.98.2.2 const activity_descr& semantics::activity () const [inline]

Definition at line 335 of file semantics.h.

#### 6.98.2.3 void semantics::convexity (const convex_e & __c) [inline]

Definition at line 330 of file semantics.h.

#### 6.98.2.4 const convex_e& semantics::convexity () const [inline]

Definition at line 329 of file semantics.h.

#### 6.98.2.5 void semantics::hard_flag (bool __c) [inline]

Definition at line 345 of file semantics.h.

#### 6.98.2.6 bool semantics::hard_flag () const [inline]

Definition at line 344 of file semantics.h.

#### 6.98.2.7 bool semantics::inactive () const [inline]

Definition at line 359 of file semantics.h.

#### 6.98.2.8 bool semantics::inactive_hi () const [inline]

Definition at line 355 of file semantics.h.

#### 6.98.2.9 bool semantics::inactive_lo () const [inline]

Definition at line 357 of file semantics.h.

#### 6.98.2.10 void semantics::integer_flag (bool __c) [inline]

Definition at line 342 of file semantics.h.

#### 6.98.2.11 bool semantics::integer_flag () const [inline]

Definition at line 341 of file semantics.h.

#### 6.98.2.12 void semantics::is_at_any_bound (bool __c) [inline]

Definition at line 348 of file semantics.h.

#### 6.98.2.13 bool semantics::is_at_any_bound () const [inline]

Definition at line 347 of file semantics.h.

#### 6.98.2.14 void semantics::kj_flag (bool __c) [inline]

Definition at line 339 of file semantics.h.

#### 6.98.2.15 bool semantics::kj_flag () const [inline]

Definition at line 338 of file semantics.h.

#### 6.98.2.16 void semantics::merge (const semantics & *__s*) [inline]

Definition at line 307 of file semantics.h.

#### 6.98.2.17 void semantics::read (char * *c*)

Definition at line 144 of file semantics.cc.

#### 6.98.2.18 bool semantics::redundancy () const [inline]

Definition at line 353 of file semantics.h.

#### 6.98.2.19 void semantics::separability (const tristate & *__c*) [inline]

Definition at line 333 of file semantics.h.

#### 6.98.2.20 const tristate& semantics::separability () const [inline]

Definition at line 332 of file semantics.h.

#### 6.98.2.21 void semantics::type (const type_annotation & *__c*) [inline]

Definition at line 351 of file semantics.h.

#### 6.98.2.22 const type_annotation& semantics::type () const [inline]

Definition at line 350 of file semantics.h.

### 6.98.3 Friends And Related Function Documentation

#### 6.98.3.1 std::ostream& operator<< (std::ostream & *o*, const semantics & *__s*) [friend]

Definition at line 224 of file semantics.cc.

### 6.98.4 Member Data Documentation

#### 6.98.4.1 unsigned int semantics::_0chnbase

Definition at line 211 of file semantics.h.

#### 6.98.4.2 activity_descr semantics::act

Definition at line 194 of file semantics.h.

#### 6.98.4.3 tristate semantics::active

Definition at line 196 of file semantics.h.

#### 6.98.4.4 int semantics::addinfo

Definition at line 213 of file semantics.h.

#### 6.98.4.5 struct { ... } semantics::annotation_flags

#### 6.98.4.6 convex_e semantics::c_info

Definition at line 193 of file semantics.h.

#### 6.98.4.7 int semantics::degree

Definition at line 217 of file semantics.h.

#### 6.98.4.8 int semantics::dim

Definition at line 218 of file semantics.h.

#### 6.98.4.9 bool semantics::hard

Definition at line 204 of file semantics.h.

#### 6.98.4.10 tristate semantics::has_0chnbase

Definition at line 208 of file semantics.h.

#### 6.98.4.11 struct { ... } semantics::info_flags

#### 6.98.4.12 bool semantics::integer

Definition at line 202 of file semantics.h.

#### 6.98.4.13 bool semantics::is_at_any_bound

Definition at line 197 of file semantics.h.

#### 6.98.4.14 bool semantics::kj

Definition at line 201 of file semantics.h.

#### 6.98.4.15 struct { ... } semantics::property_flags

#### 6.98.4.16 tristate semantics::separable

Definition at line 195 of file semantics.h.

#### 6.98.4.17 int semantics::stage

Definition at line 219 of file semantics.h.

#### 6.98.4.18 type_annotation semantics::type

Definition at line 203 of file semantics.h.

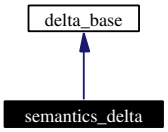
The documentation for this class was generated from the following files:

- semantics.h
- semantics.cc

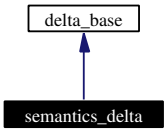
## 6.99 semantics_delta Class Reference

`#include <semantics_delta.h>`

Inheritance diagram for semantics_delta:




Collaboration diagram for semantics_delta:




### Public Methods

- uint32_t encode_convex (uint32_t e, const convex_e &c) const
- uint32_t encode_activity (uint32_t e, const activity_descr &a) const
- uint32_t encode_is_at_any_bound (uint32_t e, bool b) const
- uint32_t encode_integer (uint32_t e, bool b) const
- uint32_t encode_hard (uint32_t e, bool b) const
- uint32_t encode_separable (uint32_t e, const tristate &t) const
- uint32_t encode_type (uint32_t e, const type_annotation &a) const
- uint32_t encode (const semantics &s) const
- void decode_convex (uint32_t e, convex_e &c) const
- void decode_activity (uint32_t e, activity_descr &a) const
- void decode_is_at_any_bound (uint32_t e, bool &b) const
- void decode_integer (uint32_t e, bool &b) const
- void decode_hard (uint32_t e, bool &b) const
- void decode_separable (uint32_t e, tristate &t) const
- void decode_type (uint32_t e, type_annotation &a) const
- void decode (uint32_t e, semantics &s) const
- semantics_delta (const std::vector< unsigned int > &__i, const std::vector< uint32_t > &__b)
- semantics_delta (unsigned int __i, uint32_t __b)
- semantics_delta ()

- semantics_delta (const semantics_delta &__d)
- semantics_delta * new_copy () const
- void destroy_copy (semantics_delta *__d)
- void set (const std::vector< unsigned int > &_i, const std::vector< semantics > &_s)
- void set (const std::vector< unsigned int > &_i, const std::vector< uint32_t > &_s)
- void set (unsigned int _i, const semantics &_s)
- void set_convex (const std::vector< unsigned int > &_i, const std::vector< convex_e > &c)
- void set_convex (unsigned int _i, const convex_e &c)
- void set_activity (const std::vector< unsigned int > &_i, const std::vector< activity_descr > &a)
- void set_activity (unsigned int _i, const activity_descr &a)
- void set_separable (const std::vector< unsigned int > &_i, const std::vector< tristate > &t)
- void set_separable (unsigned int _i, const tristate &t)
- void set_is_at_any_bound (const std::vector< unsigned int > &_i, const std::vector< bool > &b)
- void set_is_at_any_bound (unsigned int _i, bool b)
- void set_integer (const std::vector< unsigned int > &_i, const std::vector< bool > &b)
- void set_integer (unsigned int _i, bool b)
- void set_hard (const std::vector< unsigned int > &_i, const std::vector< bool > &b)
- void set_hard (unsigned int _i, bool b)
- void set_type (const std::vector< unsigned int > &_i, const std::vector< type_annotation > &a)
- void set_type (unsigned int _i, const type_annotation &a)
- bool apply (work_node &_x, undelta_base *&_u) const
- virtual void destroy_copy (delta_base *__d)
- delta make_delta (const std::string &a)
- const std::string & get_action () const
- virtual void convert (work_node &_x, delta_base *&_d)
- virtual bool apply3 (work_node &_x, const work_node &_y, undelta_base *&_u) const

**Protected Attributes**

- std::string _action

**Friends**

- class semantics_undelta

### 6.99.1 Constructor & Destructor Documentation

#### 6.99.1.1 semantics_delta::semantics_delta (const std::vector< unsigned int > & *_i*, const std::vector< uint32_t > & *_b*)

Definition at line 30 of file semantics_delta.cc.

#### 6.99.1.2 semantics_delta::semantics_delta (unsigned int *_i*, uint32_t *_b*) [inline]

Definition at line 192 of file semantics_delta.h.

#### 6.99.1.3 semantics_delta::semantics_delta () [inline]

Definition at line 196 of file semantics_delta.h.

#### 6.99.1.4 semantics_delta::semantics_delta (const semantics_delta & *__d*) [inline]

Definition at line 197 of file semantics_delta.h.

### 6.99.2 Member Function Documentation

#### 6.99.2.1 bool semantics_delta::apply (work_node & *_x*, undelta_base *& *_u*) const [virtual]

Reimplemented from delta_base.

Definition at line 222 of file semantics_delta.cc.

#### 6.99.2.2 bool delta_base::apply3 (work_node & *_x*, const work_node & *_y*, undelta_base *& *_u*) const [inline, virtual, inherited]

Definition at line 63 of file api_delta.h.

#### 6.99.2.3 virtual void delta_base::convert (work_node & *_x*, delta_base *& *_d*) [inline, virtual, inherited]

Reimplemented in table_delta.

Definition at line 107 of file api_deltabase.h.

#### 6.99.2.4 void semantics_delta::decode (uint32_t *e*, semantics & *s*) const [inline]

Definition at line 177 of file semantics_delta.h.

#### 6.99.2.5 void semantics_delta::decode_activity (uint32_t *e*, activity_descr & *a*) const [inline]

Definition at line 165 of file semantics_delta.h.

#### 6.99.2.6 void semantics_delta::decode_convex (uint32_t *e*, convex_e & *c*) const [inline]

Definition at line 163 of file semantics_delta.h.

#### 6.99.2.7 void semantics_delta::decode_hard (uint32_t *e*, bool & *b*) const [inline]

Definition at line 171 of file semantics_delta.h.

#### 6.99.2.8 void semantics_delta::decode_integer (uint32_t *e*, bool & *b*) const [inline]

Definition at line 169 of file semantics_delta.h.

#### 6.99.2.9 void semantics_delta::decode_is_at_any_bound (uint32_t *e*, bool & *b*) const [inline]

Definition at line 167 of file semantics_delta.h.

#### 6.99.2.10 void semantics_delta::decode_separable (uint32_t *e*, tristate & *t*) const [inline]

Definition at line 173 of file semantics_delta.h.

#### 6.99.2.11 void semantics_delta::decode_type (uint32_t *e*, type_annotation & *a*) const [inline]

Definition at line 175 of file semantics_delta.h.

#### 6.99.2.12 virtual void delta_base::destroy_copy (delta_base * *__d*) [inline, virtual, inherited]

Definition at line 95 of file api_deltabase.h.

#### 6.99.2.13 void semantics_delta::destroy_copy (semantics_delta * *__d*) [inline]

Definition at line 207 of file semantics_delta.h.

#### 6.99.2.14 uint32_t semantics_delta::encode (const semantics & *s*) const [inline]

Definition at line 151 of file semantics_delta.h.

#### 6.99.2.15 uint32_t semantics_delta::encode_activity (uint32_t *e*, const activity_descr & *a*) const [inline]

Definition at line 139 of file semantics_delta.h.

#### 6.99.2.16 uint32_t semantics_delta::encode_convex (uint32_t *e*, const convex_e & *c*) const [inline]

Definition at line 137 of file semantics_delta.h.

#### 6.99.2.17 uint32_t semantics_delta::encode_hard (uint32_t *e*, bool *b*) const [inline]

Definition at line 145 of file semantics_delta.h.

#### 6.99.2.18 uint32_t semantics_delta::encode_integer (uint32_t *e*, bool *b*) const [inline]

Definition at line 143 of file semantics_delta.h.

#### 6.99.2.19 uint32_t semantics_delta::encode_is_at_any_bound (uint32_t *e*, bool *b*) const [inline]

Definition at line 141 of file semantics_delta.h.

#### 6.99.2.20 uint32_t semantics_delta::encode_separable (uint32_t *e*, const tristate & *t*) const [inline]

Definition at line 147 of file semantics_delta.h.

#### 6.99.2.21 uint32_t semantics_delta::encode_type (uint32_t *e*, const type_annotation & *a*) const [inline]

Definition at line 149 of file semantics_delta.h.

#### 6.99.2.22 const std::string& delta_base::get_action () const [inline, inherited]

Definition at line 105 of file api_deltabase.h.

#### 6.99.2.23 delta delta_base::make_delta (const std::string & *a*) [inline, inherited]

Definition at line 99 of file api_deltabase.h.

#### 6.99.2.24 semantics_delta* semantics_delta::new_copy () const [inline, virtual]

Reimplemented from delta_base.

Definition at line 206 of file semantics_delta.h.

#### 6.99.2.25 void semantics_delta::set (unsigned int *_i*, const semantics & *_s*)

Definition at line 124 of file semantics_delta.cc.

#### 6.99.2.26 void semantics_delta::set (const std::vector< unsigned int > & *_i*, const std::vector< uint32_t > & *_s*)

Definition at line 53 of file semantics_delta.cc.

#### 6.99.2.27 void semantics_delta::set (const std::vector< unsigned int > & *_i*, const std::vector< semantics > & *_s*)

Definition at line 43 of file semantics_delta.cc.

#### 6.99.2.28 void semantics_delta::set_activity (unsigned int *_i*, const activity_descr & *a*)

Definition at line 150 of file semantics_delta.cc.

#### 6.99.2.29 void semantics_delta::set_activity (const std::vector< unsigned int > & *_i*, const std::vector< activity_descr > & *a*)

Definition at line 72 of file semantics_delta.cc.

#### 6.99.2.30 void semantics_delta::set_convex (unsigned int *_i*, const convex_e & *c*)

Definition at line 136 of file semantics_delta.cc.

#### 6.99.2.31 void semantics_delta::set_convex (const std::vector< unsigned int > & *_i*, const std::vector< convex_e > & *c*)

Definition at line 62 of file semantics_delta.cc.

#### 6.99.2.32 void semantics_delta::set_hard (unsigned int *_i*, bool *b*)

Definition at line 193 of file semantics_delta.cc.

#### 6.99.2.33 void semantics_delta::set_hard (const std::vector< unsigned int > & *_i*, const std::vector< bool > & *b*)

Definition at line 104 of file semantics_delta.cc.

#### 6.99.2.34 void semantics_delta::set_integer (unsigned int *_i*, bool *b*)

Definition at line 179 of file semantics_delta.cc.

#### 6.99.2.35 void semantics_delta::set_integer (const std::vector< unsigned int > & *_i*, const std::vector< bool > & *b*)

Definition at line 93 of file semantics_delta.cc.

#### 6.99.2.36 void semantics_delta::set_is_at_any_bound (unsigned int *_i*, bool *b*)

#### 6.99.2.37 void semantics_delta::set_is_at_any_bound (const std::vector< unsigned int > & *_i*, const std::vector< bool > & *b*)

#### 6.99.2.38 void semantics_delta::set_separable (unsigned int *_i*, const tristate & *t*)

Definition at line 164 of file semantics_delta.cc.

#### 6.99.2.39 void semantics_delta::set_separable (const std::vector< unsigned int > & *_i*, const std::vector< tristate > & *t*)

Definition at line 82 of file semantics_delta.cc.

#### 6.99.2.40 void semantics_delta::set_type (unsigned int *_i*, const type_annotation & *a*)

Definition at line 207 of file semantics_delta.cc.

#### 6.99.2.41 void semantics_delta::set_type (const std::vector< unsigned int > & *_i*, const std::vector< type_annotation > & *a*)

Definition at line 114 of file semantics_delta.cc.

### 6.99.3 Friends And Related Function Documentation

#### 6.99.3.1 friend class semantics_undelta [friend]

Definition at line 238 of file semantics_delta.h.

### 6.99.4 Member Data Documentation

#### 6.99.4.1 std::string delta_base::_action [protected, inherited]

Definition at line 86 of file api_deltabase.h.

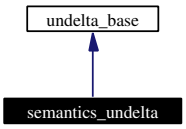
The documentation for this class was generated from the following files:

- semantics_delta.h
- semantics_delta.cc

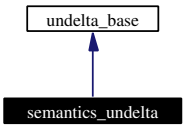
## 6.100 semantics_undelta Class Reference

`#include <semantics_delta.h>`

Inheritance diagram for semantics_undelta:




Collaboration diagram for semantics_undelta:




### Public Methods

- semantics_undelta (const std::vector< unsigned int > &__i, const std::vector< uint32_t > &__s)
- semantics_undelta (const std::vector< unsigned int > &__i)
- semantics_undelta (const semantics_undelta &__d)
- semantics_undelta * new_copy () const
- void destroy_copy (semantics_undelta *__d)
- bool unapply (work_node &_x) const
- virtual void destroy_copy (undelta_base *__d)
- undelta make_undelta ()
- virtual bool unapply3 (work_node &_x, const work_node &_y) const

### Friends

- class semantics_delta

### 6.100.1 Constructor & Destructor Documentation

#### 6.100.1.1 semantics_undelta::semantics_undelta (const std::vector< unsigned int > & *__i*, const std::vector< uint32_t > & *__s*) [inline]

Definition at line 42 of file semantics_delta.h.

#### 6.100.1.2 semantics_undelta::semantics_undelta (const std::vector< unsigned int > & *__i*) [inline]

Definition at line 47 of file semantics_delta.h.

#### 6.100.1.3 semantics_undelta::semantics_undelta (const semantics_undelta & *__d*) [inline]

Definition at line 51 of file semantics_delta.h.

### 6.100.2 Member Function Documentation

#### 6.100.2.1 virtual void undelta_base::destroy_copy (undelta_base * _d) [inline, virtual, inherited]

Definition at line 146 of file api_deltabase.h.

#### 6.100.2.2 void semantics_undelta::destroy_copy (semantics_undelta * _d) [inline]

Definition at line 61 of file semantics_delta.h.

#### 6.100.2.3 undelta undelta_base::make_undelta () [inline, inherited]

Definition at line 150 of file api_deltabase.h.

#### 6.100.2.4 semantics_undelta* semantics_undelta::new_copy () const [inline, virtual]

Reimplemented from undelta_base.

Definition at line 60 of file semantics_delta.h.

#### 6.100.2.5 bool semantics_undelta::unapply (work_node & _x) const [virtual]

Reimplemented from undelta_base.

Definition at line 252 of file semantics_delta.cc.

#### 6.100.2.6 bool undelta_base::unapply3 (work_node & _x, const work_node & _y) const [inline, virtual, inherited]

Definition at line 69 of file api_delta.h.

### 6.100.3 Friends And Related Function Documentation

#### 6.100.3.1 friend class semantics_delta [friend]

Definition at line 65 of file semantics_delta.h.

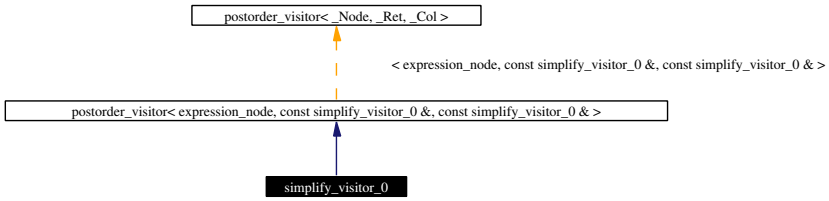
The documentation for this class was generated from the following files:

- semantics_delta.h
- semantics_delta.cc

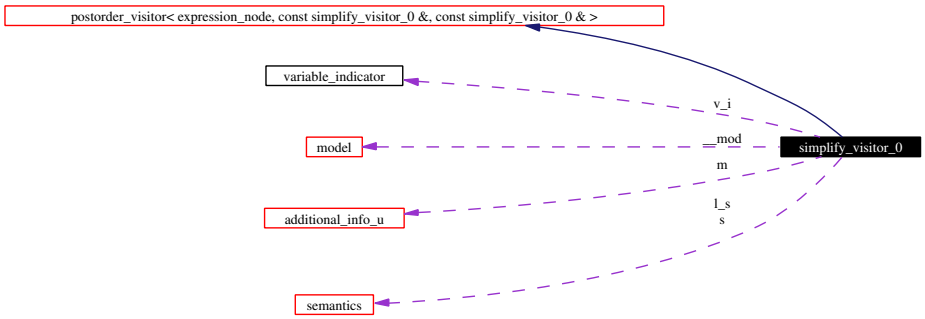
## 6.101 simplify_visitor_0 Class Reference

`#include <model-inline.h>`

Inheritance diagram for simplify_visitor_0:




Collaboration diagram for simplify_visitor_0:




### Public Methods

- simplify_visitor_0 ()
- simplify_visitor_0 (int __nv, std::vector< unsigned int > *__dn, std::vector< std::pair< unsigned int, unsigned int > > *__de, model *__m)
- simplify_visitor_0 (const simplify_visitor_0 &__x)
- void vinit ()
- void init ()
- void postorder_help (const expression_node &r, unsigned int n_chld)
- bool postorder (expression_node &r)
- const simplify_visitor_0 & value ()
- const simplify_visitor_0 & vvalue ()
- void simple_sum_prod_update (expression_node &r, const simplify_visitor_0 &__s)
- void transfer_ghost_down (unsigned int nnum, unsigned int pnum)
- void vcollect (const simplify_visitor_0 &__s)
- void collect (expression_node &r, const simplify_visitor_0 &__s)
- virtual bool postorder (const expression_node &__n)
- virtual void collect (const expression_node &__n, collect_value __r)

### 6.101.1 Constructor & Destructor Documentation

#### 6.101.1.1 simplify_visitor_0::simplify_visitor_0 () [inline]

Definition at line 585 of file model-inline.h.

#### 6.101.1.2 simplify_visitor_0::simplify_visitor_0 (int *__nv*, std::vector< unsigned int > * *__dn*, std::vector< std::pair< unsigned int, unsigned int > > * *__de*, model * *__m*) [inline]

Definition at line 591 of file model-inline.h.

#### 6.101.1.3 simplify_visitor_0::simplify_visitor_0 (const simplify_visitor_0 & __x) [inline]

Definition at line 600 of file model-inline.h.

### 6.101.2 Member Function Documentation

#### 6.101.2.1 virtual void postorder_visitor< expression_node, const simplify_visitor_0 &, const simplify_visitor_0 & >::collect (const expression_node & __n, collect_value __r) [virtual, inherited]

#### 6.101.2.2 void simplify_visitor_0::collect (expression_node & r, const simplify_visitor_0 & __s)

Definition at line 1244 of file model.cc.

#### 6.101.2.3 void simplify_visitor_0::init () [inline, virtual]

Reimplemented from **postorder_visitor< expression_node, const simplify_visitor_0 &, const simplify_visitor_0 & >**.

Definition at line 613 of file model-inline.h.

#### 6.101.2.4 virtual bool postorder_visitor< expression_node, const simplify_visitor_0 &, const simplify_visitor_0 & >::postorder (const expression_node & __n) [virtual, inherited]

#### 6.101.2.5 bool simplify_visitor_0::postorder (expression_node & r)

Definition at line 1120 of file model.cc.

#### 6.101.2.6 void simplify_visitor_0::postorder_help (const expression_node & r, unsigned int n_chld) [inline]

Definition at line 2386 of file model-inline.h.

#### 6.101.2.7 void simplify_visitor_0::simple_sum_prod_update (expression_node & r, const simplify_visitor_0 & __s) [inline]

Definition at line 621 of file model-inline.h.

#### 6.101.2.8 void simplify_visitor_0::transfer_ghost_down (unsigned int nnum, unsigned int pnum) [inline]

Definition at line 650 of file model-inline.h.

#### 6.101.2.9 const simplify_visitor_0& simplify_visitor_0::value () [inline]

Definition at line 618 of file model-inline.h.

#### 6.101.2.10 void simplify_visitor_0::vcollect (const simplify_visitor_0 & __s)

Definition at line 1148 of file model.cc.

#### 6.101.2.11 void simplify_visitor_0::vinit () [inline, virtual]

Reimplemented from **postorder_visitor< expression_node, const simplify_visitor_0 &, const simplify_visitor_0 & >**.

Definition at line 610 of file model-inline.h.

#### 6.101.2.12 const simplify_visitor_0& simplify_visitor_0::vvalue () [inline, virtual]

Reimplemented from **postorder_visitor< expression_node, const simplify_visitor_0 &, const simplify_visitor_0 & >**.

Definition at line 619 of file model-inline.h.

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

- model-inline.h
- model.cc

## 6.102 sort_constraints Class Reference

`#include <model-inline.h>`

### Public Methods

- bool operator() (const walker &_c1, const walker &_c2) const

### 6.102.1 Member Function Documentation

#### 6.102.1.1 bool sort_constraints::operator() (const walker & *_c1*, const walker & *_c2*) const [inline]

Definition at line 525 of file model-inline.h.

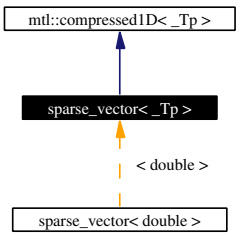
The documentation for this class was generated from the following file:

- model-inline.h

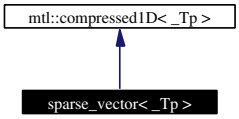
## 6.103 sparse_vector< _Tp > Class Template Reference

`#include <linalg.h>`

Inheritance diagram for sparse_vector< _Tp >:



Collaboration diagram for sparse_vector< _Tp >:




**template<class _Tp> class sparse_vector< _Tp >**

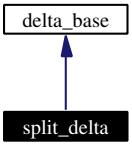
The documentation for this class was generated from the following file:

- linalg.h

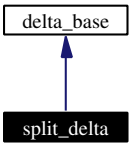
## 6.104 split_delta Class Reference

`#include <split_delta.h>`

Inheritance diagram for split_delta:




Collaboration diagram for split_delta:




### Public Methods

- split_delta ()
- split_delta (unsigned int _node_num, const interval &_l, const interval &_u)
- split_delta (unsigned int _node_num, const std::vector< interval > &_m)
- split_delta (const std::vector< unsigned int > &_i, const std::vector< interval > &_l, const std::vector< interval > &_u)
- split_delta (const std::list< std::vector< delta > > &__dl)
- ~split_delta ()
- split_delta (const split_delta &__s)
- split_delta * new_copy () const
- void destroy_copy (split_delta *__d)
- void add_delta (const delta &__d)

- void add_deltas (const std::vector< delta > &__d)
- void add_split (unsigned int _nnum, interval _l, interval _u)
- void add_split (const std::vector< unsigned int > &_i, const std::vector< interval > &_l, const std::vector< interval > &_u)
- bool apply (work_node &x, undelta_base *&_u) const
- virtual void destroy_copy (delta_base *__d)
- delta make_delta (const std::string &a)
- const std::string & get_action () const
- virtual void convert (work_node &_x, delta_base *&_d)
- virtual bool apply3 (work_node &_x, const work_node &_y, undelta_base *&_u) const

**Public Attributes**

- std::list< std::vector< delta > > splits

**Protected Attributes**

- std::string _action

### 6.104.1 Constructor & Destructor Documentation

#### 6.104.1.1 split_delta::split_delta () [inline]

Definition at line 77 of file split_delta.h.

#### 6.104.1.2 split_delta::split_delta (unsigned int *_node_num*, const interval & *_l*, const interval & *_u*) [inline]

Definition at line 83 of file split_delta.h.

#### 6.104.1.3 split_delta::split_delta (unsigned int *_node_num*, const std::vector< interval > & *_m*) [inline]

Definition at line 92 of file split_delta.h.

#### 6.104.1.4 split_delta::split_delta (const std::vector< unsigned int > & *_i*, const std::vector< interval > & *_l*, const std::vector< interval > & *_u*) [inline]

Definition at line 103 of file split_delta.h.

#### 6.104.1.5 split_delta::split_delta (const std::list< std::vector< delta > > & *__dl*) [inline]

Definition at line 111 of file split_delta.h.

#### 6.104.1.6 split_delta::~split_delta () [inline]

Definition at line 115 of file split_delta.h.

#### 6.104.1.7 split_delta::split_delta (const split_delta & *__s*) [inline]

Definition at line 117 of file split_delta.h.

### 6.104.2 Member Function Documentation

#### 6.104.2.1 void split_delta::add_delta (const delta & _d) [inline]

Definition at line 122 of file split_delta.h.

#### 6.104.2.2 void split_delta::add_deltas (const std::vector< delta > & _d) [inline]

Definition at line 127 of file split_delta.h.

#### 6.104.2.3 void split_delta::add_split (const std::vector< unsigned int > & _i, const std::vector< interval > & _l, const std::vector< interval > & _u) [inline]

Definition at line 138 of file split_delta.h.

#### 6.104.2.4 void split_delta::add_split (unsigned int _nnum, interval _l, interval _u) [inline]

Definition at line 132 of file split_delta.h.

#### 6.104.2.5 bool split_delta::apply (work_node & x, undelta_base *& _u) const [inline, virtual]

Reimplemented from delta_base.

Definition at line 146 of file split_delta.h.

#### 6.104.2.6 bool delta_base::apply3 (work_node & _x, const work_node & _y, undelta_base *& _u) const [inline, virtual, inherited]

Definition at line 63 of file api_delta.h.

#### 6.104.2.7 virtual void delta_base::convert (work_node & _x, delta_base *& _d) [inline, virtual, inherited]

Reimplemented in table_delta.

Definition at line 107 of file api_deltabase.h.

#### 6.104.2.8 virtual void delta_base::destroy_copy (delta_base * _d) [inline, virtual, inherited]

Definition at line 95 of file api_deltabase.h.

#### 6.104.2.9 void split_delta::destroy_copy (split_delta * _d) [inline]

Definition at line 120 of file split_delta.h.

#### 6.104.2.10 const std::string& delta_base::get_action () const [inline, inherited]

Definition at line 105 of file api_deltabase.h.

#### 6.104.2.11 delta delta_base::make_delta (const std::string & a) [inline, inherited]

Definition at line 99 of file api_deltabase.h.

#### 6.104.2.12 split_delta* split_delta::new_copy () const [inline, virtual]

Reimplemented from delta_base.

Definition at line 119 of file split_delta.h.

### 6.104.3 Member Data Documentation

#### 6.104.3.1 std::string delta_base::_action [protected, inherited]

Definition at line 86 of file api_deltabase.h.

#### 6.104.3.2 std::list<std::vector<delta> > split_delta::splits

Definition at line 70 of file split_delta.h.

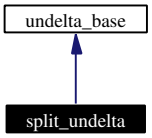
The documentation for this class was generated from the following file:

- split_delta.h

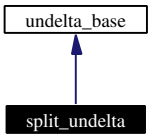
## 6.105 split_undelta Class Reference

`#include <split_delta.h>`

Inheritance diagram for split_undelta:




Collaboration diagram for split_undelta:




### Public Methods

- split_undelta (const work_node::transaction_number &_se)
- split_undelta (const split_undelta &_su)
- virtual ~split_undelta ()
- split_undelta * new_copy () const
- void destroy_copy (split_undelta *__d)
- bool unapply (work_node &x) const
- virtual void destroy_copy (undelta_base *__d)
- undelta make_undelta ()
- virtual bool unapply3 (work_node &_x, const work_node &_y) const

### 6.105.1 Constructor & Destructor Documentation

#### 6.105.1.1 split_undelta::split_undelta (const work_node::transaction_number & *_se*) [inline]

Definition at line 39 of file split_delta.h.

#### 6.105.1.2 split_undelta::split_undelta (const split_undelta & *_su*) [inline]

Definition at line 41 of file split_delta.h.

#### 6.105.1.3 virtual split_undelta::~split_undelta () [inline, virtual]

Definition at line 48 of file split_delta.h.

### 6.105.2 Member Function Documentation

#### 6.105.2.1 virtual void undelta_base::destroy_copy (undelta_base * *__d*) [inline, virtual, inherited]

Definition at line 146 of file api_deltabase.h.

#### 6.105.2.2 void split_undelta::destroy_copy (split_undelta * *__d*) [inline]

Definition at line 51 of file split_delta.h.

#### 6.105.2.3 undelta undelta_base::make_undelta () [inline, inherited]

Definition at line 150 of file api_deltabase.h.

#### 6.105.2.4 split_undelta* split_undelta::new_copy () const [inline, virtual]

Reimplemented from undelta_base.

Definition at line 50 of file split_delta.h.

#### 6.105.2.5 bool split_undelta::unapply (work_node & *x*) const [inline, virtual]

Reimplemented from undelta_base.

Definition at line 53 of file split_delta.h.

#### 6.105.2.6 bool undelta_base::unapply3 (work_node & *_x*, const work_node & *_y*) const [inline, virtual, inherited]

Definition at line 69 of file api_delta.h.

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

- split_delta.h

## 6.106 statistic_info Class Reference

`#include <ie_statistic.h>`

### Public Methods

- statistic_info ()
- virtual ~statistic_info ()

### Public Attributes

- double effectiveness
- int number_of_infers

#### 6.106.1 Constructor & Destructor Documentation

##### 6.106.1.1 statistic_info::statistic_info () [inline]

Definition at line 36 of file ie_statistic.h.

##### 6.106.1.2 virtual statistic_info::~statistic_info () [inline, virtual]

Definition at line 37 of file ie_statistic.h.

#### 6.106.2 Member Data Documentation

##### 6.106.2.1 double statistic_info::effectiveness

Definition at line 33 of file ie_statistic.h.

##### 6.106.2.2 int statistic_info::number_of_infers

Definition at line 34 of file ie_statistic.h.

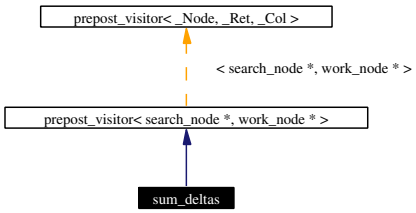
The documentation for this class was generated from the following file:

- ie_statistic.h

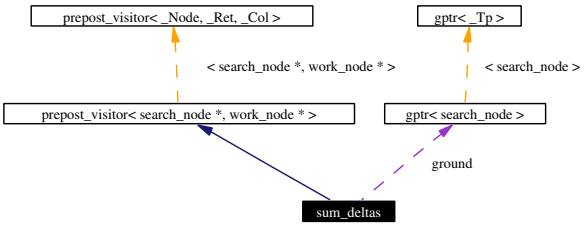
## 6.107 sum_deltas Class Reference

`#include <sum_deltas.h>`

Inheritance diagram for sum_deltas:




Collaboration diagram for sum_deltas:




### Public Methods

- sum_deltas (gptr< search_node > &search_graph_ground)
- void vinit ()
- void vcollect (return_value const &)
- return_value vvalue ()
- bool preorder (search_node *const &r)
- void collect (search_node *const &, return_value const &ancestor)
- bool postorder (search_node *const &r)
- return_value value ()
- virtual void collect (const search_node *&__n, collect_value __r)

### Public Attributes

- vector< return_value > ancestors
- return_value result
- gptr< search_node > & ground

## 6.107.1 Constructor & Destructor Documentation

### 6.107.1.1 sum_deltas::sum_deltas (gptr< search_node > & *search_graph_ground*) [inline]

Definition at line 54 of file sum_deltas.h.

## 6.107.2 Member Function Documentation

### 6.107.2.1 virtual void prepost_visitor< search_node *, work_node *, _Col >::collect (const search_node * & *__n*, collect_value *__r*) [virtual, inherited]

### 6.107.2.2 void sum_deltas::collect (search_node *const &, return_value const & *ancestor*) [inline]

Definition at line 82 of file sum_deltas.h.

### 6.107.2.3 bool sum_deltas::postorder (search_node *const & *r*) [inline, virtual]

Reimplemented from **prepost_visitor< search_node *, work_node * >**.

Definition at line 88 of file sum_deltas.h.

#### 6.107.2.4 bool sum_deltas::preorder (search_node *const & r) [inline, virtual]

Reimplemented from **prepost_visitor< search_node *, work_node * >**.

Definition at line 75 of file sum_deltas.h.

#### 6.107.2.5 return_value sum_deltas::value () [inline]

Definition at line 116 of file sum_deltas.h.

#### 6.107.2.6 void sum_deltas::vcollect (return_value const &) [inline]

Definition at line 62 of file sum_deltas.h.

#### 6.107.2.7 void sum_deltas::vinit () [inline, virtual]

Reimplemented from **prepost_visitor< search_node *, work_node * >**.

Definition at line 57 of file sum_deltas.h.

#### 6.107.2.8 return_value sum_deltas::vvalue () [inline, virtual]

Reimplemented from **prepost_visitor< search_node *, work_node * >**.

Definition at line 69 of file sum_deltas.h.

### 6.107.3 Member Data Documentation

#### 6.107.3.1 vector<return_value> sum_deltas::ancestors

Definition at line 50 of file sum_deltas.h.

#### 6.107.3.2 gptr<search_node>& sum_deltas::ground

Definition at line 52 of file sum_deltas.h.

#### 6.107.3.3 return_value sum_deltas::result

Definition at line 51 of file sum_deltas.h.

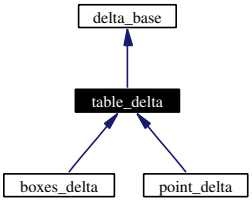
The documentation for this class was generated from the following file:

- sum_deltas.h

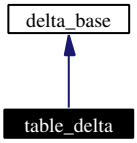
## 6.108 table_delta Class Reference

`#include <table_delta.h>`

Inheritance diagram for table_delta:




Collaboration diagram for table_delta:




### Public Types

- typedef std::pair< std::string, dbt_row > t_line
- typedef std::vector< t_line > t_ctr

### Public Methods

- table_delta (const std::string &a)
- table_delta (const std::string &a, const t_ctr &_n, const std::vector< annotation > &_r)
- table_delta (const table_delta &__d)
- table_delta (const std::string &a, const std::string &__tn, const dbt_row &__t)
- table_delta (const std::string &a, const std::string &__tn, const std::vector< dbt_row > &__t)
- void add (const t_line &_tl)
- void add (const std::string &_tn, const dbt_row &_r)
- void add (const std::vector< t_line > &_tlv)
- void rm (const annotation &_tr)
- void rm (const std::vector< annotation > &_trv)
- virtual table_delta * new_copy () const
- virtual void destroy_copy (table_delta *__d)
- virtual void create_table (work_node &_x, vdbl::standard_table *&ptb, const std::string &__t) const
- virtual bool apply (work_node &_x, undelta_base *&_u) const
- void convert (work_node &_x, delta_base *&_u)
- virtual void destroy_copy (delta_base *__d)
- delta make_delta (const std::string &a)
- const std::string & get_action () const
- virtual bool apply3 (work_node &_x, const work_node &_y, undelta_base *&_u) const

### Protected Attributes

- std::string _action

### 6.108.1 Member Typedef Documentation

#### 6.108.1.1 typedef std::vector<t_line> table_delta::t_ctr

Definition at line 36 of file table_delta.h.

#### 6.108.1.2 typedef std::pair<std::string,dbt_row> table_delta::t_line

Definition at line 35 of file table_delta.h.

### 6.108.2 Constructor & Destructor Documentation

#### 6.108.2.1 table_delta::table_delta (const std::string & *a*) [inline]

Definition at line 43 of file table_delta.h.

#### 6.108.2.2 table_delta::table_delta (const std::string & *a*, const t_ctr & *_n*, const std::vector< annotation > & *_r*) [inline]

Definition at line 45 of file table_delta.h.

#### 6.108.2.3 table_delta::table_delta (const table_delta & *__d*) [inline]

Definition at line 49 of file table_delta.h.

#### 6.108.2.4 table_delta::table_delta (const std::string & *a*, const std::string & *__tn*, const dbt_row & *__t*) [inline]

Definition at line 57 of file table_delta.h.

#### 6.108.2.5 table_delta::table_delta (const std::string & *a*, const std::string & *__tn*, const std::vector< dbt_row > & *__t*) [inline]

Definition at line 61 of file table_delta.h.

### 6.108.3 Member Function Documentation

#### 6.108.3.1 void table_delta::add (const std::vector< t_line > & *_tlv*) [inline]

Definition at line 73 of file table_delta.h.

#### 6.108.3.2 void table_delta::add (const std::string & *_tn*, const dbt_row & *_r*) [inline]

Definition at line 71 of file table_delta.h.

#### 6.108.3.3 void table_delta::add (const t_line & *_tl*) [inline]

Definition at line 70 of file table_delta.h.

#### 6.108.3.4 bool table_delta::apply (work_node & _x, undelta_base *& _u) const [virtual]

Reimplemented from delta_base.

Reimplemented in boxes_delta, and point_delta.

Definition at line 30 of file table_delta.cc.

#### 6.108.3.5 bool delta_base::apply3 (work_node & _x, const work_node & _y, undelta_base *& _u) const [inline, virtual, inherited]

Definition at line 63 of file api_delta.h.

#### 6.108.3.6 void table_delta::convert (work_node & _x, delta_base *& _u) [virtual]

Reimplemented from delta_base.

Definition at line 37 of file table_delta.cc.

#### 6.108.3.7 virtual void table_delta::create_table (work_node & _x, vdbl::standard_table *& *ptb*, const std::string & __t) const [inline, virtual]

Reimplemented in boxes_delta, and point_delta.

Definition at line 83 of file table_delta.h.

#### 6.108.3.8 virtual void delta_base::destroy_copy (delta_base * __d) [inline, virtual, inherited]

Definition at line 95 of file api_deltabase.h.

#### 6.108.3.9 virtual void table_delta::destroy_copy (table_delta * __d) [inline, virtual]

Definition at line 81 of file table_delta.h.

#### 6.108.3.10 const std::string& delta_base::get_action () const [inline, inherited]

Definition at line 105 of file api_deltabase.h.

#### 6.108.3.11 delta delta_base::make_delta (const std::string & *a*) [inline, inherited]

Definition at line 99 of file api_deltabase.h.

#### 6.108.3.12 virtual table_delta* table_delta::new_copy () const [inline, virtual]

Reimplemented from delta_base.

Reimplemented in boxes_delta, and point_delta.

Definition at line 79 of file table_delta.h.

#### 6.108.3.13 void table_delta::rm (const std::vector< annotation > & _trv) [inline]

Definition at line 76 of file table_delta.h.

#### 6.108.3.14 void table_delta::rm (const annotation & _tr) [inline]

Definition at line 75 of file table_delta.h.

### 6.108.4 Member Data Documentation

#### 6.108.4.1 std::string delta_base::_action [protected, inherited]

Definition at line 86 of file api_deltabase.h.

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

- table_delta.h
- table_delta.cc

## 6.109 termination_reason Class Reference

`#include <termreason.h>`

### Public Methods

- termination_reason ()
- termination_reason (int termr_r, const std::string &termr_ref)
- termination_reason (const termination_reason &__t)
- termination_reason & operator= (const termination_reason &__t)
- const std::string & get_message () const
- int get_code () const

### Friends

- std::ostream & operator<< (std::ostream &o, const termination_reason &__x)

### 6.109.1 Constructor & Destructor Documentation

#### 6.109.1.1 termination_reason::termination_reason () [inline]

Definition at line 45 of file termreason.h.

#### 6.109.1.2 termination_reason::termination_reason (int *termr_r*, const std::string & *termr_ref*) [inline]

Definition at line 48 of file termreason.h.

#### 6.109.1.3 termination_reason::termination_reason (const termination_reason & *__t*) [inline]

Definition at line 52 of file termreason.h.

### 6.109.2 Member Function Documentation

#### 6.109.2.1 int termination_reason::get_code () const [inline]

Definition at line 74 of file termreason.h.

#### 6.109.2.2 const std::string& termination_reason::get_message () const [inline]

Definition at line 64 of file termreason.h.

#### 6.109.2.3 termination_reason& termination_reason::operator= (const termination_reason & *_t*) [inline]

Definition at line 56 of file termreason.h.

### 6.109.3 Friends And Related Function Documentation

#### 6.109.3.1 std::ostream& operator<< (std::ostream & *o*, const termination_reason & *_x*) [friend]

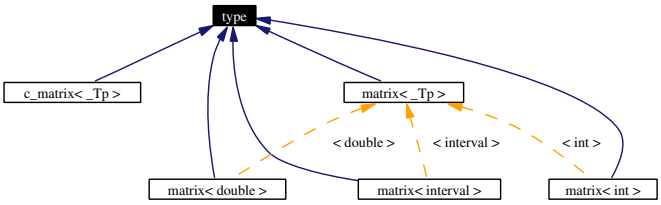
Definition at line 79 of file termreason.h.

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

- termreason.h

## 6.110 type Class Reference

Inheritance diagram for type:



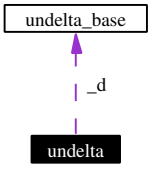

The documentation for this class was generated from the following file:

- linalg.h

## 6.111 undelta Class Reference

`#include <api_deltabase.h>`

Collaboration diagram for undelta:



### Public Methods

- undelta ()
- undelta (undelta_base *__d)
- undelta (const undelta &__d)
- ~undelta ()
- const undelta_base * get_base () const
- bool unapply (work_node &_x) const
- bool unapply3 (work_node &_x, const work_node &_y) const
- undelta & operator= (const undelta &_u)

#### 6.111.1 Constructor & Destructor Documentation

##### 6.111.1.1 undelta::undelta () [inline]

Definition at line 125 of file api_deltabase.h.

##### 6.111.1.2 undelta::undelta (undelta_base * __d) [inline]

Definition at line 126 of file api_deltabase.h.

##### 6.111.1.3 undelta::undelta (const undelta & __d) [inline]

Definition at line 161 of file api_delta.h.

##### 6.111.1.4 undelta::~undelta () [inline]

Definition at line 167 of file api_delta.h.

#### 6.111.2 Member Function Documentation

##### 6.111.2.1 const undelta_base * undelta::get_base () [inline]

Definition at line 182 of file api_delta.h.

##### 6.111.2.2 undelta & undelta::operator= (const undelta & _u) [inline]

Definition at line 173 of file api_delta.h.

##### 6.111.2.3 bool undelta::unapply (work_node & _x) const [inline]

Definition at line 184 of file api_delta.h.

##### 6.111.2.4 bool undelta::unapply3 (work_node & _x, const work_node & _y) const [inline]

Definition at line 187 of file api_delta.h.

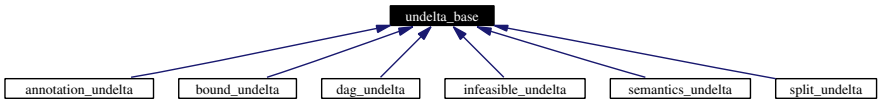
The documentation for this class was generated from the following files:

- api_deltabase.h
- api_delta.h

## 6.112 undelta_base Class Reference

`#include <api_deltabase.h>`

Inheritance diagram for undelta_base:




### Public Methods

- undelta_base ()
- undelta_base (const undelta_base &__d)
- virtual undelta_base * new_copy () const
- virtual void destroy_copy (undelta_base *__d)
- virtual ~undelta_base ()
- undelta make_undelta ()
- virtual bool unapply (work_node &_x) const
- virtual bool unapply3 (work_node &_x, const work_node &_y) const

### 6.112.1 Constructor & Destructor Documentation

#### 6.112.1.1 undelta_base::undelta_base () [inline]

Definition at line 142 of file api_deltabase.h.

#### 6.112.1.2 undelta_base::undelta_base (const undelta_base & *__d*) [inline]

Definition at line 143 of file api_deltabase.h.

#### 6.112.1.3 virtual undelta_base::~undelta_base () [inline, virtual]

Definition at line 148 of file api_deltabase.h.

### 6.112.2 Member Function Documentation

#### 6.112.2.1 virtual void undelta_base::destroy_copy (undelta_base * *__d*) [inline, virtual]

Definition at line 146 of file api_deltabase.h.

#### 6.112.2.2 undelta undelta_base::make_undelta () [inline]

Definition at line 150 of file api_deltabase.h.

#### 6.112.2.3 virtual undelta_base* undelta_base::new_copy () const [inline, virtual]

Reimplemented in annotation_undelta, bound_undelta, dag_undelta, infeasible_undelta, semantics_undelta, and split_undelta.

Definition at line 145 of file api_deltabase.h.

#### 6.112.2.4 virtual bool undelta_base::unapply (work_node & _x) const [inline, virtual]

Reimplemented in annotation_undelta, bound_undelta, dag_undelta, infeasible_undelta, semantics_undelta, and split_undelta.

Definition at line 156 of file api_deltabase.h.

#### 6.112.2.5 bool undelta_base::unapply3 (work_node & _x, const work_node & _y) const [inline, virtual]

Definition at line 69 of file api_delta.h.

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

- api_deltabase.h
- api_delta.h

## 6.113 variable_indicator Class Reference

`#include <evaluator.h>`

### Public Methods

- variable_indicator ()
- variable_indicator (int num_of_vars)
- variable_indicator (const std::vector< int > &__v, int num_of_vars)
- variable_indicator (const variable_indicator &__x)
- void reserve (int num_of_vars)
- void set_all (const variable_indicator &_vi)
- void set (int __i)
- void set (std::vector< int > __v)
- void set (int start_idx, int end_idx)
- void clear ()
- void unset (int __i)
- void unset (std::vector< int > __v)
- void unset (int start_idx, int end_idx)
- unsigned int sum (int start_idx, int end_idx) const
- bool test (int __i) const
- bool match (const variable_indicator &__v) const
- variable_indicator & operator= (const variable_indicator &__v)
- bool operator== (const variable_indicator &__v)
- std::vector< int > encode ()
- void decode (const std::vector< int > &__e)

### 6.113.1 Constructor & Destructor Documentation

#### 6.113.1.1 variable_indicator::variable_indicator () [inline]

Definition at line 63 of file evaluator.h.

#### 6.113.1.2 variable_indicator::variable_indicator (int *num_of_vars*) [inline]

Definition at line 65 of file evaluator.h.

#### 6.113.1.3 variable_indicator::variable_indicator (const std::vector< int > & __v, int *num_of_vars*) [inline]

Definition at line 67 of file evaluator.h.

#### 6.113.1.4 variable_indicator::variable_indicator (const variable_indicator & __x) [inline]

Definition at line 74 of file evaluator.h.

### 6.113.2 Member Function Documentation

#### 6.113.2.1 void variable_indicator::clear () [inline]

Definition at line 125 of file evaluator.h.

#### 6.113.2.2 void variable_indicator::decode (const std::vector< int > & __e) [inline]

Definition at line 229 of file evaluator.h.

#### 6.113.2.3 std::vector<int> variable_indicator::encode () [inline]

Definition at line 220 of file evaluator.h.

#### 6.113.2.4 bool variable_indicator::match (const variable_indicator & __v) const [inline]

Definition at line 195 of file evaluator.h.

#### 6.113.2.5 variable_indicator& variable_indicator::operator= (const variable_indicator & __v) [inline]

Definition at line 209 of file evaluator.h.

#### 6.113.2.6 bool variable_indicator::operator== (const variable_indicator & __v) [inline]

Definition at line 215 of file evaluator.h.

#### 6.113.2.7 void variable_indicator::reserve (int *num_of_vars*) [inline]

Definition at line 76 of file evaluator.h.

#### 6.113.2.8 void variable_indicator::set (int *start_idx*, int *end_idx*) [inline]

Definition at line 106 of file evaluator.h.

#### 6.113.2.9 void variable_indicator::set (std::vector< int > __v) [inline]

Definition at line 100 of file evaluator.h.

#### 6.113.2.10 void variable_indicator::set (int __i) [inline]

Definition at line 95 of file evaluator.h.

#### 6.113.2.11 void variable_indicator::set_all (const variable_indicator & _vi) [inline]

Definition at line 83 of file evaluator.h.

#### 6.113.2.12 unsigned int variable_indicator::sum (int *start_idx*, int *end_idx*) const [inline]

Definition at line 161 of file evaluator.h.

#### 6.113.2.13 bool variable_indicator::test (int __i) const [inline]

Definition at line 190 of file evaluator.h.

#### 6.113.2.14 void variable_indicator::unset (int *start_idx*, int *end_idx*) [inline]

Definition at line 142 of file evaluator.h.

#### 6.113.2.15 void variable_indicator::unset (std::vector< int > __v) [inline]

Definition at line 136 of file evaluator.h.

#### 6.113.2.16 void variable_indicator::unset (int __i) [inline]

Definition at line 131 of file evaluator.h.

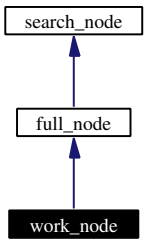
The documentation for this class was generated from the following file:

- evaluator.h

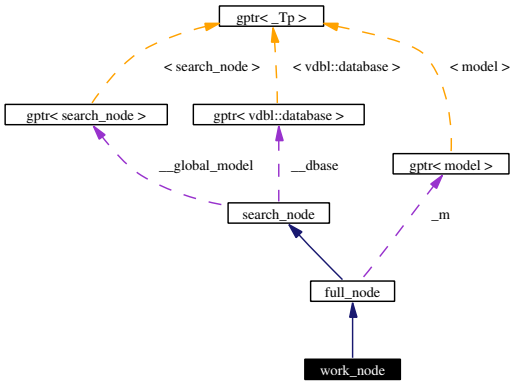
## 6.114 work_node Class Reference

`#include <search_node.h>`

Inheritance diagram for work_node:




Collaboration diagram for work_node:




## Public Types

- typedef constraint_iterator_base< model::walker, const std::vector< model::walker > *, const std::vector< model::walker > &, const model::walker *, const model::walker &, std::vector< model::walker >::const_iterator > constraint_const_iterator
- typedef constraint_iterator_base< model::walker, std::vector< model::walker > *, std::vector< model::walker > &, model::walker *, model::walker &, std::vector< model::walker >::iterator > constraint_iterator
- typedef uint32_t transaction_number

## Public Methods

- transaction_number get_transaction_number ()
- void init_cnumbers ()
- void reset_node_ranges ()
- void make_node_ranges (bool keep_old_ranges)
- double compute_log_volume (const std::vector< interval > &_r) const
- work_node (const work_node &__w)
- work_node (const search_node_id &_i, const **vdbl::userid** &_dui, gptr< model > &__m, gptr< vdbl::database > &__d, const std::vector< annotation > &__an, const std::list< delta_id > &__de, gptr< search_node > *_gm, search_node_relation snr=snr_worknode)
- virtual ~work_node ()
- const model * get_model () const
- model * get_model ()
- model get (unsigned int __type)
- constraint_const_iterator get_begin (unsigned int __type) const
- constraint_const_iterator get_end (unsigned int __type) const
- constraint_iterator get_begin (unsigned int __type)
- constraint_iterator get_end (unsigned int __type)
- delta get_delta (const delta_id &_id)
- const delta & get_delta (const delta_id &_id) const
- double log_volume () const
- double gain () const
- double reset_gain ()
- unsigned int n (unsigned int __type) const
- bool is_delta () const

- const annotation & get_annotation (unsigned int i) const
- const std::vector< annotation > & get_annotations () const
- const vdbl::database * get_database () const
- model * get_model_ptr () const
- vdbl::database * get_database_ptr () const
- **vdbl::userid** get_dbuserid () const
- gptr< search_node > * global_model () const
- gptr< vdbl::database > * database () const
- search_node_id get_id () const
- void keep (const annotation &_an)
- void keep (const std::vector< annotation > &_anv)
- void unkeep (const annotation &_an)
- void unkeep (const std::vector< annotation > &_anv)

### Public Attributes

- std::list< delta_id > deltas
- std::map< delta_id, undelta > undeltas
- vdbl::standard_table * dtable
- **vdbl::tableid** dtable_id
- std::vector< interval > node_ranges
- bool infeasible
- double log_vol
- double gain_factor
- std::map< transaction_number, std::list< std::vector< delta > > > proposed_splits
- std::vector< annotation > _ann

### Protected Methods

- search_node_id node_id () const
- search_node_relation parent_relation () const

### Protected Attributes

- gptr< model > * _m
- gptr< search_node > * __global_model
- gptr< vdbl::database > * __dbase
- **vdbl::userid** _dbuser
- search_node_relation _snr
- search_node_id _id
- std::vector< annotation > _keep

**Friends**

- class delta
- class undelta
- class work_node_comp_hook
- work_node operator+ (const work_node &_w, const delta_id &_d)
- work_node operator- (const work_node &_w, const delta_id &_d)
- work_node & operator+= (work_node &_w, const delta_id &_d)
- work_node & operator-= (work_node &_w, const delta_id &_d)
- template<template< class _Tp, class _A > class _Ctr, class _Al> work_node operator+ (const work_node &_w, const _Ctr< delta_id, _Al > &_d)
- template<template< class _Tp, class _A > class _Ctr, class _Al> work_node operator- (const work_node &_w, const _Ctr< delta_id, _Al > &_d)
- template<template< class _Tp, class _A > class _Ctr, class _Al> work_node & operator+= (work_node &_w, const _Ctr< delta_id, _Al > &_d)
- template<template< class _Tp, class _A > class _Ctr, class _Al> work_node & operator-= (work_node &_w, const _Ctr< delta_id, _Al > &_d)
- class delta_base
- class dag_delta
- class dag_undelta
- class search_graph

### 6.114.1 Member Typedef Documentation

#### 6.114.1.1 typedef constraint_iterator_base<model::walker,const std::vector<model::walker>*, const std::vector<model::walker>&,const model::walker*, const model::walker&, std::vector<model::walker>::const_iterator> work_node::constraint_const_iterator

Definition at line 300 of file search_node.h.

#### 6.114.1.2 typedef constraint_iterator_base<model::walker,std::vector<model::walker>*, std::vector<model::walker>&,model::walker*,model::walker&, std::vector<model::walker>::iterator> work_node::constraint_iterator

Definition at line 303 of file search_node.h.

#### 6.114.1.3 typedef uint32_t work_node::transaction_number

Definition at line 321 of file search_node.h.

### 6.114.2 Constructor & Destructor Documentation

#### 6.114.2.1 work_node::work_node (const work_node & *__w*) [inline]

Definition at line 346 of file search_node.h.

#### 6.114.2.2 work_node::work_node (const search_node_id & *_i*, const vdbl::userid & *_dui*, gptr< model > & *__m*, gptr< vdbl::database > & *__d*, const std::vector< annotation > & *__an*, const std::list< delta_id > & *__de*, gptr< search_node > * *_gm*, search_node_relation *snr* = snr_worknode)

Definition at line 89 of file search_node.cc.

#### 6.114.2.3 virtual work_node::~work_node () [inline, virtual]

Definition at line 371 of file search_node.h.

### 6.114.3 Member Function Documentation

#### 6.114.3.1 double work_node::compute_log_volume (const std::vector< interval > & *_r*) const

Definition at line 175 of file search_node.cc.

#### 6.114.3.2 gptr<vdbl::database>* search_node::database () const [inline, inherited]

Definition at line 147 of file search_node.h.

#### 6.114.3.3 double work_node::gain () const [inline]

Definition at line 403 of file search_node.h.

#### 6.114.3.4 model work_node::get (unsigned int *__type*)

#### 6.114.3.5 const annotation& full_node::get_annotation (unsigned int *i*) const [inline, inherited]

Definition at line 258 of file search_node.h.

#### 6.114.3.6 const std::vector<annotation>& full_node::get_annotations () const [inline, inherited]

Definition at line 263 of file search_node.h.

#### 6.114.3.7 work_node::constraint_iterator work_node::get_begin (unsigned int *__type*) [inline]

Definition at line 554 of file search_node.h.

#### 6.114.3.8 work_node::constraint_const_iterator work_node::get_begin (unsigned int *__type*) const [inline]

Definition at line 539 of file search_node.h.

#### 6.114.3.9 const vdbl::database* full_node::get_database () const [inline, inherited]

Definition at line 270 of file search_node.h.

#### 6.114.3.10 vdbl::database* full_node::get_database_ptr () const [inline, inherited]

Definition at line 278 of file search_node.h.

#### 6.114.3.11 vdbl::userid search_node::get_dbuserid () const [inline, inherited]

Definition at line 143 of file search_node.h.

#### 6.114.3.12 const delta & work_node::get_delta (const delta_id & *_id*) const [inline]

Definition at line 680 of file search_node.h.

#### 6.114.3.13 delta work_node::get_delta (const delta_id & *_id*) [inline]

Definition at line 666 of file search_node.h.

#### 6.114.3.14 work_node::constraint_iterator work_node::get_end (unsigned int *__type*) [inline]

Definition at line 561 of file search_node.h.

#### 6.114.3.15 work_node::constraint_const_iterator work_node::get_end (unsigned int *__type*) const [inline]

Definition at line 547 of file search_node.h.

#### 6.114.3.16 search_node_id search_node::get_id () const [inline, inherited]

Definition at line 149 of file search_node.h.

#### 6.114.3.17 model* work_node::get_model () [inline]

Definition at line 374 of file search_node.h.

#### 6.114.3.18 const model* work_node::get_model () const [inline]

Reimplemented from full_node.

Definition at line 373 of file search_node.h.

#### 6.114.3.19 model* full_node::get_model_ptr () const [inline, inherited]

Definition at line 273 of file search_node.h.

#### 6.114.3.20 transaction_number work_node::get_transaction_number () [inline]

Definition at line 327 of file search_node.h.

#### 6.114.3.21 gptr<search_node>* search_node::global_model () const [inline, inherited]

Definition at line 145 of file search_node.h.

#### 6.114.3.22 void work_node::init_cnumbers ()

Definition at line 129 of file search_node.cc.

#### 6.114.3.23 bool full_node::is_delta () const [inline, virtual, inherited]

Reimplemented from search_node.

Definition at line 256 of file search_node.h.

#### 6.114.3.24 void search_node::keep (const std::vector< annotation > & *anv*) [inline, inherited]

Definition at line 153 of file search_node.h.

#### 6.114.3.25 void search_node::keep (const annotation & *an*) [inline, inherited]

Definition at line 151 of file search_node.h.

#### 6.114.3.26 double work_node::log_volume () const [inline]

Definition at line 402 of file search_node.h.

#### 6.114.3.27 void work_node::make_node_ranges (bool *keep_old_ranges*)

Definition at line 151 of file search_node.cc.

#### 6.114.3.28 unsigned int work_node::n (unsigned int *__type*) const [inline]

Definition at line 704 of file search_node.h.

#### 6.114.3.29 search_node_id search_node::node_id () const [inline, protected, inherited]

Definition at line 96 of file search_node.h.

#### 6.114.3.30 search_node_relation search_node::parent_relation () const [inline, protected, inherited]

Definition at line 97 of file search_node.h.

#### 6.114.3.31 double work_node::reset_gain () [inline]

Definition at line 404 of file search_node.h.

#### 6.114.3.32 void work_node::reset_node_ranges () [inline]

Definition at line 694 of file search_node.h.

#### 6.114.3.33 void search_node::unkeep (const std::vector< annotation > & *anv*) [inline, inherited]

Definition at line 167 of file search_node.h.

#### 6.114.3.34 void search_node::unkeep (const annotation & *an*) [inline, inherited]

Definition at line 156 of file search_node.h.

### 6.114.4 Friends And Related Function Documentation

#### 6.114.4.1 friend class dag_delta [friend, inherited]

Definition at line 284 of file search_node.h.

#### 6.114.4.2 friend class dag_undelta [friend, inherited]

Definition at line 285 of file search_node.h.

#### 6.114.4.3 friend class delta [friend]

Definition at line 428 of file search_node.h.

#### 6.114.4.4 friend class delta_base [friend, inherited]

Definition at line 283 of file search_node.h.

#### 6.114.4.5 template<template< class _Tp, class _A > class _Ctr, class _Al> work_node operator+ (const work_node & *_w*, const _Ctr< delta_id, _Al > & *_d*) [friend]

Definition at line 590 of file search_node.h.

#### 6.114.4.6 work_node operator+ (const work_node & *_w*, const delta_id & *_i*) [friend]

Definition at line 570 of file search_node.h.

#### 6.114.4.7 template<template< class _Tp, class _A > class _Ctr, class _Al> work_node& operator+= (work_node & *_w*, const _Ctr< delta_id, _Al > & *_d*) [friend]

Definition at line 616 of file search_node.h.

#### 6.114.4.8 work_node& operator+= (work_node & *_w*, const delta_id & *_i*) [friend]

Definition at line 580 of file search_node.h.

#### 6.114.4.9 template<template< class _Tp, class _A > class _Ctr, class _Al> work_node operator- (const work_node & *_w*, const _Ctr< delta_id, _Al > & *_d*) [friend]

Definition at line 602 of file search_node.h.

#### 6.114.4.10 work_node operator- (const work_node & *_w*, const delta_id & *_d*) [friend]

Definition at line 30 of file search_node.cc.

#### 6.114.4.11 template<template< class _Tp, class _A > class _Ctr, class _Al> work_node& operator-= (work_node & *_w*, const _Ctr< delta_id, _Al > & *_d*) [friend]

Definition at line 627 of file search_node.h.

#### 6.114.4.12 work_node& operator-= (work_node & *_w*, const delta_id & *_d*) [friend]

Definition at line 60 of file search_node.cc.

#### 6.114.4.13 friend class search_graph [friend, inherited]

Definition at line 174 of file search_node.h.

#### 6.114.4.14 friend class undelta [friend]

Definition at line 429 of file search_node.h.

#### 6.114.4.15 friend class work_node_comp_hook [friend]

Definition at line 430 of file search_node.h.

### 6.114.5 Member Data Documentation

#### 6.114.5.1 gptr<vdbl::database>* search_node::__dbase [protected, inherited]

Definition at line 89 of file search_node.h.

#### 6.114.5.2 gptr<search_node>* search_node::__global_model [protected, inherited]

Definition at line 88 of file search_node.h.

#### 6.114.5.3 std::vector<annotation> full_node::_ann [inherited]

Definition at line 213 of file search_node.h.

#### 6.114.5.4 vdbl::userid search_node::_dbuser [protected, inherited]

Definition at line 90 of file search_node.h.

#### 6.114.5.5 search_node_id search_node::_id [protected, inherited]

Definition at line 92 of file search_node.h.

#### 6.114.5.6 std::vector<annotation> search_node::_keep [protected, inherited]

Definition at line 93 of file search_node.h.

#### 6.114.5.7 gptr<model>* full_node::_m [protected, inherited]

Definition at line 210 of file search_node.h.

#### 6.114.5.8 search_node_relation search_node::_snr [protected, inherited]

Definition at line 91 of file search_node.h.

#### 6.114.5.9 std::list<delta_id> work_node::deltas

Definition at line 306 of file search_node.h.

#### 6.114.5.10 vdbl::standard_table* work_node::dtable

Definition at line 308 of file search_node.h.

#### 6.114.5.11 vdbl::tableid work_node::dtable_id

Definition at line 309 of file search_node.h.

#### 6.114.5.12 double work_node::gain_factor

Definition at line 318 of file search_node.h.

#### 6.114.5.13 bool work_node::infeasible

Definition at line 315 of file search_node.h.

#### 6.114.5.14 double work_node::log_vol

Definition at line 317 of file search_node.h.

#### 6.114.5.15 std::vector<interval> work_node::node_ranges

Definition at line 314 of file search_node.h.

#### 6.114.5.16 std::map<transaction_number,std::list<std::vector<delta> > > work_node::proposed_splits

Definition at line 330 of file search_node.h.

#### 6.114.5.17 std::map<delta_id,undelta> work_node::undeltas

Definition at line 307 of file search_node.h.

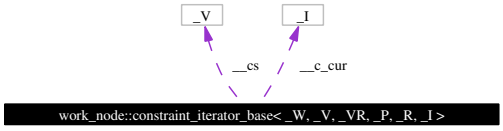
The documentation for this class was generated from the following files:

- search_node.h
- search_node.cc

## 6.115 work_node::constraint_iterator_base< _W, _V, _VR, _P, _R, _I > Class Template Reference

`#include <search_node.h>`

Collaboration diagram for work_node::constraint_iterator_base< _W, _V, _VR, _P, _R, _I >:




### Public Types

- typedef std::bidirectional_iterator_tag iterator_category
- typedef _W value_type
- typedef _R reference
- typedef _P pointer
- typedef size_t size_type
- typedef ptrdiff_t difference_type

**Public Methods**

- constraint_iterator_base ()
- constraint_iterator_base (_Reference cs, unsigned int tp)
- constraint_iterator_base (const _Self &__c)
- ~constraint_iterator_base ()
- reference operator * () const
- pointer operator -> () const
- bool operator== (const _Self &_c)
- bool operator!= (const _Self &_c)
- _Self & set (const _Iterator &_c)
- _Self & operator++ ()
- _Self operator++ (int)
- _Self & operator– ()
- _Self operator– (int)
- _Self & operator= (const _Self &_c)

**template<class _W, class _V, class _VR, class _P, class _R, class _I> class work_node::constraint_iterator_base< _W, _V, _VR, _P, _R, _I >**

### 6.115.1 Member Typedef Documentation

#### 6.115.1.1 template<class _W, class _V, class _VR, class _P, class _R, class _I> typedef ptrdiff_t work_node::constraint_iterator_base< _W, _V, _VR, _P, _R, _I >::difference_type

Definition at line 457 of file search_node.h.

#### 6.115.1.2 template<class _W, class _V, class _VR, class _P, class _R, class _I> typedef std::bidirectional_iterator_tag work_node::constraint_iterator_base< _W, _V, _VR, _P, _R, _I >::iterator_category

Definition at line 450 of file search_node.h.

#### 6.115.1.3 template<class _W, class _V, class _VR, class _P, class _R, class _I> typedef _P work_node::constraint_iterator_base< _W, _V, _VR, _P, _R, _I >::pointer

Definition at line 454 of file search_node.h.

#### 6.115.1.4 template<class _W, class _V, class _VR, class _P, class _R, class _I> typedef _R work_node::constraint_iterator_base< _W, _V, _VR, _P, _R, _I >::reference

Definition at line 453 of file search_node.h.

#### 6.115.1.5 template<class _W, class _V, class _VR, class _P, class _R, class _I> typedef size_t work_node::constraint_iterator_base< _W, _V, _VR, _P, _R, _I >::size_type

Definition at line 456 of file search_node.h.

#### 6.115.1.6 template<class _W, class _V, class _VR, class _P, class _R, class _I> typedef _W work_node::constraint_iterator_base< _W, _V, _VR, _P, _R, _I >::value_type

Definition at line 452 of file search_node.h.

### 6.115.2 Constructor & Destructor Documentation

**6.115.2.1 template<class _W, class _V, class _VR, class _P, class _R, class _I> work_node::constraint_iterator_base< _W, _V, _VR, _P, _R, _I >::constraint_iterator_base ()** `[inline]`

Definition at line 460 of file search_node.h.

**6.115.2.2 template<class _W, class _V, class _VR, class _P, class _R, class _I> work_node::constraint_iterator_base< _W, _V, _VR, _P, _R, _I >::constraint_iterator_base (_Reference *cs*, unsigned int *tp*)** `[inline]`

Definition at line 461 of file search_node.h.

**6.115.2.3 template<class _W, class _V, class _VR, class _P, class _R, class _I> work_node::constraint_iterator_base< _W, _V, _VR, _P, _R, _I >::constraint_iterator_base (const _Self & *__c*)** `[inline]`

Definition at line 463 of file search_node.h.

**6.115.2.4 template<class _W, class _V, class _VR, class _P, class _R, class _I> work_node::constraint_iterator_base< _W, _V, _VR, _P, _R, _I >::~constraint_iterator_base ()** `[inline]`

Definition at line 466 of file search_node.h.

### 6.115.3 Member Function Documentation

**6.115.3.1 template<class _W, class _V, class _VR, class _P, class _R, class _I> reference work_node::constraint_iterator_base< _W, _V, _VR, _P, _R, _I >::operator * () const** `[inline]`

Definition at line 468 of file search_node.h.

**6.115.3.2 template<class _W, class _V, class _VR, class _P, class _R, class _I> bool work_node::constraint_iterator_base< _W, _V, _VR, _P, _R, _I >::operator!= (const _Self & *_c*)** `[inline]`

Definition at line 474 of file search_node.h.

**6.115.3.3 template<class _W, class _V, class _VR, class _P, class _R, class _I> _Self work_node::constraint_iterator_base< _W, _V, _VR, _P, _R, _I >::operator++ (int)** `[inline]`

Definition at line 499 of file search_node.h.

**6.115.3.4 template<class _W, class _V, class _VR, class _P, class _R, class _I> _Self& work_node::constraint_iterator_base< _W, _V, _VR, _P, _R, _I >::operator++ ()** `[inline]`

Definition at line 487 of file search_node.h.

**6.115.3.5 template<class _W, class _V, class _VR, class _P, class _R, class _I> _Self work_node::constraint_iterator_base< _W, _V, _VR, _P, _R, _I >::operator– (int)** `[inline]`

Definition at line 522 of file search_node.h.

**6.115.3.6 template<class _W, class _V, class _VR, class _P, class _R, class _I> _Self& work_node::constraint_iterator_base< _W, _V, _VR, _P, _R, _I >::operator– ()** `[inline]`

Definition at line 506 of file search_node.h.

**6.115.3.7 template<class _W, class _V, class _VR, class _P, class _R, class _I> pointer work_node::constraint_iterator_base< _W, _V, _VR, _P, _R, _I >::operator → () const** `[inline]`

Definition at line 469 of file search_node.h.

**6.115.3.8 template<class _W, class _V, class _VR, class _P, class _R, class _I> _Self& work_node::constraint_iterator_base< _W, _V, _VR, _P, _R, _I >::operator= (const _Self & _c)** `[inline]`

Definition at line 529 of file search_node.h.

**6.115.3.9 template<class _W, class _V, class _VR, class _P, class _R, class _I> bool work_node::constraint_iterator_base< _W, _V, _VR, _P, _R, _I >::operator== (const _Self & _c)** `[inline]`

Definition at line 471 of file search_node.h.

**6.115.3.10 template<class _W, class _V, class _VR, class _P, class _R, class _I> _Self& work_node::constraint_iterator_base< _W, _V, _VR, _P, _R, _I >::set (const _Iterator & _c)** `[inline]`

Definition at line 477 of file search_node.h.

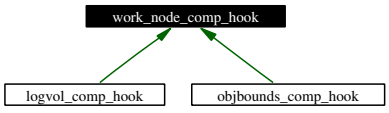
The documentation for this class was generated from the following file:

- search_node.h

## 6.116 work_node_comp_hook Class Reference

`#include <comp_hook.h>`

Inheritance diagram for work_node_comp_hook:




### Public Methods

- work_node_comp_hook (const std::string &name)
- virtual ~work_node_comp_hook ()
- virtual void operator() (const work_node &wn, dbt_row &dbr)=0
- virtual bool init_columns (vdbl::standard_table &stable)
- virtual bool drop_columns (vdbl::standard_table &stable)
- const std::string & name ()

**Protected Methods**

- template<class _CI> bool _init_column (vdbl::standard_table &stable, const std::string &colname, const _CI &c)
- template<class _CI> bool _init_column (vdbl::standard_table &stable, const char *colname, const _CI &c)
- bool _drop_columns (vdbl::standard_table &stable)
- search_node_relation parent_relation (const work_node &wn) const
- search_node_id node_id (const work_node &wn) const

### 6.116.1 Constructor & Destructor Documentation

#### 6.116.1.1 work_node_comp_hook::work_node_comp_hook (const std::string & *name*) [inline]

Definition at line 57 of file comp_hook.h.

#### 6.116.1.2 virtual work_node_comp_hook::~work_node_comp_hook () [inline, virtual]

Definition at line 60 of file comp_hook.h.

### 6.116.2 Member Function Documentation

#### 6.116.2.1 bool work_node_comp_hook::_drop_columns (vdbl::standard_table & *stable*) [protected]

Definition at line 42 of file comp_hook.cc.

#### 6.116.2.2 template<class _CI> bool work_node_comp_hook::_init_column (vdbl::standard_table & *stable*, const char * *colname*, const _CI & *c*) [inline, protected]

Definition at line 45 of file comp_hook.h.

#### 6.116.2.3 template<class _CI> bool work_node_comp_hook::_init_column (vdbl::standard_table & *stable*, const std::string & *colname*, const _CI & *c*) [protected]

Definition at line 31 of file comp_hook.cc.

#### 6.116.2.4 virtual bool work_node_comp_hook::drop_columns (vdbl::standard_table & *stable*) [inline, virtual]

Reimplemented in logvol_comp_hook, and objbounds_comp_hook.

Definition at line 66 of file comp_hook.h.

#### 6.116.2.5 virtual bool work_node_comp_hook::init_columns (vdbl::standard_table & *stable*) [inline, virtual]

Reimplemented in logvol_comp_hook, and objbounds_comp_hook.

Definition at line 64 of file comp_hook.h.

#### 6.116.2.6 const std::string& work_node_comp_hook::name () [inline]

Definition at line 68 of file comp_hook.h.

#### 6.116.2.7 search_node_id work_node_comp_hook::node_id (const work_node & *wn*) const [protected]

Definition at line 51 of file comp_hook.cc.

#### 6.116.2.8 virtual void work_node_comp_hook::operator() (const work_node & *wn*, dbt_row & *dbr*) [pure virtual]

Implemented in logvol_comp_hook, and objbounds_comp_hook.

#### 6.116.2.9 search_node_relation work_node_comp_hook::parent_relation (const work_node & *wn*) const [protected]

Definition at line 54 of file comp_hook.cc.

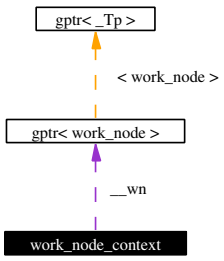
The documentation for this class was generated from the following files:

- comp_hook.h
- comp_hook.cc

## 6.117 work_node_context Class Reference

`#include <dbtools.h>`

Collaboration diagram for work_node_context:




### Public Methods

- work_node_context ()
- work_node_context (const gptr< work_node > *_i)
- work_node_context (const work_node_context &_w)
- virtual ~work_node_context ()
- work_node_context & operator= (const work_node_context &_w)
- const gptr< work_node > * wn () const

### 6.117.1 Constructor & Destructor Documentation

#### 6.117.1.1 work_node_context::work_node_context () [inline]

Definition at line 39 of file dbtools.h.

#### 6.117.1.2 work_node_context::work_node_context (const gptr< work_node > * *_i*) [inline]

Definition at line 40 of file dbtools.h.

#### 6.117.1.3 work_node_context::work_node_context (const work_node_context & *_w*) [inline]

Definition at line 41 of file dbtools.h.

#### 6.117.1.4 virtual work_node_context::~work_node_context () [inline, virtual]

Definition at line 42 of file dbtools.h.

### 6.117.2 Member Function Documentation

#### 6.117.2.1 work_node_context& work_node_context::operator= (const work_node_context & *_w*) [inline]

Definition at line 44 of file dbtools.h.

#### 6.117.2.2 const gptr<work_node>* work_node_context::wn () const [inline]

Definition at line 47 of file dbtools.h.

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

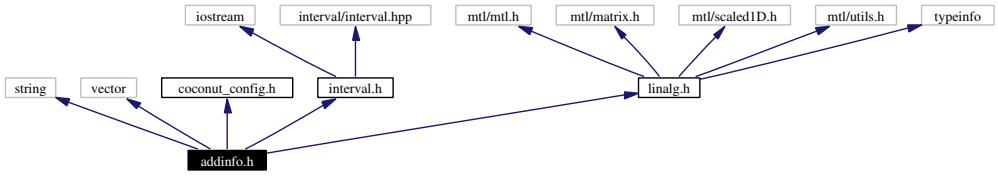
- dbtools.h

# 7 COCONUT API File Documentation

## 7.1 addinfo.h File Reference

```
#include <string>
#include <vector>
#include <coconut_config.h>
#include <interval.h>
#include <linalg.h>
```

Include dependency graph for addinfo.h:




### Compounds

- class additional_info_u

### Defines

- #define ADDINFO_INTERVAL -5
- #define ADDINFO_DOUBLE -4
- #define ADDINFO_UINT -3
- #define ADDINFO_INT -2
- #define ADDINFO_BOOL -1
- #define ADDINFO_EMPTY 0
- #define ADDINFO_ALLOCED_B 1
- #define ADDINFO_ALLOCED_N 2
- #define ADDINFO_ALLOCED_U 3
- #define ADDINFO_ALLOCED_D 4
- #define ADDINFO_ALLOCED_I 5
- #define ADDINFO_ALLOCED_S 6
- #define ADDINFO_ALLOCED_M 7
- #define ADDINFO_ALLOCED_NM 8
- #define ADDINFO_ALLOCED_IM 9

#### 7.1.1 Detailed Description

Definition in file addinfo.h.

#### 7.1.2 Define Documentation

##### 7.1.2.1 #define ADDINFO_ALLOCED_B 1

Definition at line 45 of file addinfo.h.

##### 7.1.2.2 #define ADDINFO_ALLOCED_D 4

Definition at line 48 of file addinfo.h.

##### 7.1.2.3 #define ADDINFO_ALLOCED_I 5

Definition at line 49 of file addinfo.h.

##### 7.1.2.4 #define ADDINFO_ALLOCED_IM 9

Definition at line 53 of file addinfo.h.

#### 7.1.2.5 #define ADDINFO_ALLOCED_M 7

Definition at line 51 of file addinfo.h.

#### 7.1.2.6 #define ADDINFO_ALLOCED_N 2

Definition at line 46 of file addinfo.h.

#### 7.1.2.7 #define ADDINFO_ALLOCED_NM 8

Definition at line 52 of file addinfo.h.

#### 7.1.2.8 #define ADDINFO_ALLOCED_S 6

Definition at line 50 of file addinfo.h.

#### 7.1.2.9 #define ADDINFO_ALLOCED_U 3

Definition at line 47 of file addinfo.h.

#### 7.1.2.10 #define ADDINFO_BOOL -1

Definition at line 41 of file addinfo.h.

#### 7.1.2.11 #define ADDINFO_DOUBLE -4

Definition at line 38 of file addinfo.h.

#### 7.1.2.12 #define ADDINFO_EMPTY 0

Definition at line 43 of file addinfo.h.

#### 7.1.2.13 #define ADDINFO_INT -2

Definition at line 40 of file addinfo.h.

#### 7.1.2.14 #define ADDINFO_INTERVAL -5

Definition at line 37 of file addinfo.h.

#### 7.1.2.15 #define ADDINFO_UINT -3

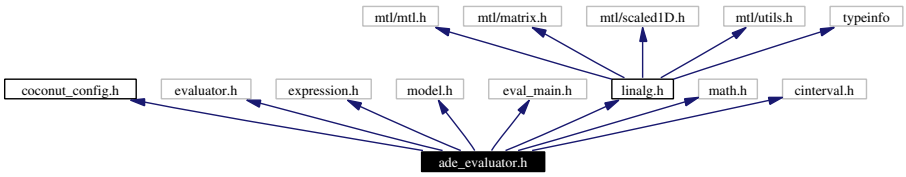
Definition at line 39 of file addinfo.h.

## 7.2 ade_evaluator.h File Reference

```
#include <coconut_config.h>
#include <evaluator.h>
#include <expression.h>
#include <model.h>
```

```
#include <eval_main.h>
#include <linalg.h>
#include <math.h>
#include <cinterval.h>
```

Include dependency graph for ade_evaluator.h:




### Namespaces

- namespace vgtl

### Compounds

- class analyticd_eval
- struct analyticd_eval_type

### Typedefs

- typedef analyticd(∗ analyticd_evaluator )(const std::vector< analyticd > ∗__x, const variable_indicator &__v)

#### 7.2.1 Detailed Description

Definition in file ade_evaluator.h.

#### 7.2.2 Typedef Documentation

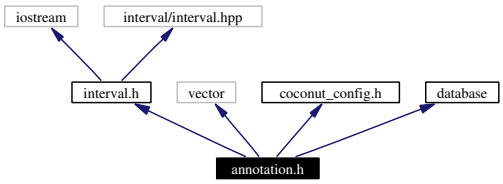
##### 7.2.2.1 typedef analyticd(∗ analyticd_evaluator)(const std::vector<analyticd>∗ __x, const variable_indicator& __v)

Definition at line 41 of file ade_evaluator.h.

## 7.3 annotation.h File Reference

```
#include <interval.h>
#include <vector>
#include <coconut_config.h>
#include <database>
```

Include dependency graph for annotation.h:




**Compounds**

- class annotation

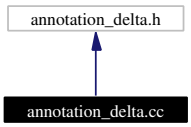
### 7.3.1 Detailed Description

Definition in file annotation.h.

## 7.4 annotation_delta.cc File Reference

`#include <annotation_delta.h>`

Include dependency graph for annotation_delta.cc:



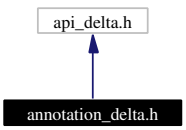

### 7.4.1 Detailed Description

Definition in file annotation_delta.cc.

## 7.5 annotation_delta.h File Reference

`#include <api_delta.h>`

Include dependency graph for annotation_delta.h:




**Compounds**

- class annotation_delta
- class annotation_undelta

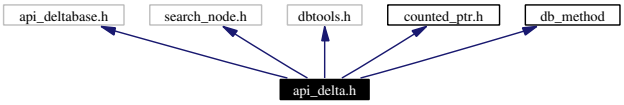
### 7.5.1 Detailed Description

Definition in file annotation_delta.h.

## 7.6 api_delta.h File Reference

```
#include <api_deltabase.h>
#include <search_node.h>
#include <dbtools.h>
#include <counted_ptr.h>
#include <db_method>
```

Include dependency graph for api_delta.h:




### Compounds

- class delta_get_action

### 7.6.1 Detailed Description

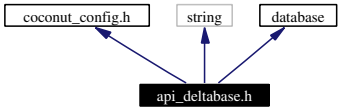
This is an internal header file, which is not intended for use outside the API.

Definition in file api_delta.h.

## 7.7 api_deltabase.h File Reference

```
#include <coconut_config.h>
#include <string>
#include <database>
```

Include dependency graph for api_deltabase.h:




### Compounds

- class delta
- class delta_base

- class undelta
- class undelta_base

**Defines**

- #define DEBUG_DELTA 0

**Typedefs**

- typedef **vdbl::rowid** delta_id

**Functions**

- std::ostream & operator<< (std::ostream &o, const delta &t)

### 7.7.1 Detailed Description

This is an internal header file not intended for use outside the API

Definition in file api_deltabase.h.

### 7.7.2 Define Documentation

#### 7.7.2.1 #define DEBUG_DELTA 0

Definition at line 37 of file api_deltabase.h.

### 7.7.3 Typedef Documentation

#### 7.7.3.1 typedef vdbl::rowid delta_id

Definition at line 45 of file api_deltabase.h.

### 7.7.4 Function Documentation

#### 7.7.4.1 std::ostream& operator<< (std::ostream & *o*, const delta & *t*) [inline]

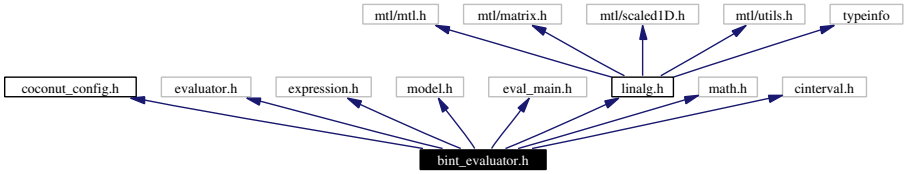
Definition at line 77 of file api_deltabase.h.

## 7.8 bint_evaluator.h File Reference

```
#include <coconut_config.h>
#include <evaluator.h>
#include <expression.h>
#include <model.h>
#include <eval_main.h>
#include <linalg.h>
```

```
#include <math.h>
#include <cinterval.h>
```

Include dependency graph for bint_evaluator.h:




### Compounds

- class b_interval_eval
- struct b_interval_eval_type

### Typedefs

- typedef b_interval(* b_interval_evaluator )(const std::vector< b_interval > *__x, const variable_indicator &__v)

#### 7.8.1 Detailed Description

Definition in file bint_evaluator.h.

#### 7.8.2 Typedef Documentation

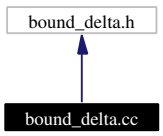
##### 7.8.2.1 typedef b_interval(* b_interval_evaluator)(const std::vector<b_interval>* __x, const variable_indicator& __v)

Definition at line 41 of file bint_evaluator.h.

## 7.9 bound_delta.cc File Reference

```
#include <bound_delta.h>
```

Include dependency graph for bound_delta.cc:



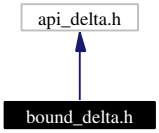

#### 7.9.1 Detailed Description

Definition in file bound_delta.cc.

## 7.10 bound_delta.h File Reference

`#include <api_delta.h>`

Include dependency graph for bound_delta.h:




### Compounds

- class bound_delta
- class bound_undelta

### 7.10.1 Detailed Description

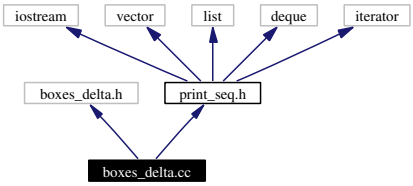
Definition in file bound_delta.h.

## 7.11 boxes_delta.cc File Reference

`#include <boxes_delta.h>`

`#include <print_seq.h>`

Include dependency graph for boxes_delta.cc:




### 7.11.1 Detailed Description

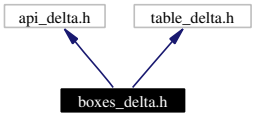
Definition in file boxes_delta.cc.

## 7.12 boxes_delta.h File Reference

`#include <api_delta.h>`

`#include <table_delta.h>`

Include dependency graph for boxes_delta.h:




### Compounds

- class boxes_delta

#### 7.12.1 Detailed Description

Definition in file boxes_delta.h.

## 7.13 cdat-inline.h File Reference

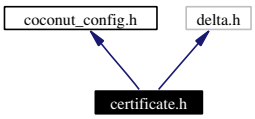
#### 7.13.1 Detailed Description

Definition in file cdat-inline.h.

## 7.14 certificate.h File Reference

```
#include <coconut_config.h>
#include <delta.h>
```

Include dependency graph for certificate.h:




### Compounds

- class certificate

#### 7.14.1 Detailed Description

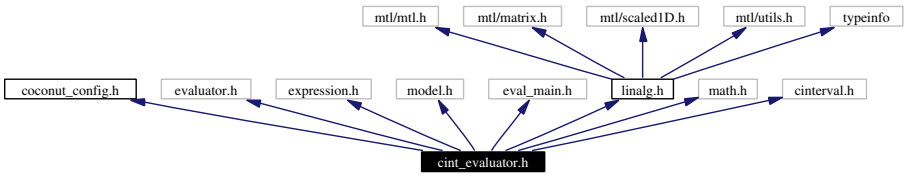
Definition in file certificate.h.

## 7.15 cint_evaluator.h File Reference

```
#include <coconut_config.h>
#include <evaluator.h>
#include <expression.h>
```

```
#include <model.h>
#include <eval_main.h>
#include <linalg.h>
#include <math.h>
#include <cinterval.h>
```

Include dependency graph for cint_evaluator.h:




### Compounds

- class cinterval_eval
- struct cinterval_eval_type

### Typedefs

- typedef cinterval(∗ cinterval_evaluator )(const std::vector< cinterval > ∗__x, const variable_indicator &__v)

#### 7.15.1 Detailed Description

Definition in file cint_evaluator.h.

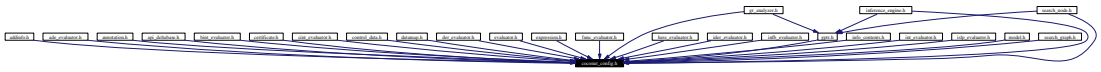
#### 7.15.2 Typedef Documentation

##### 7.15.2.1 typedef cinterval(∗ cinterval_evaluator)(const std::vector<cinterval>∗ __x, const variable_indicator& __v)

Definition at line 41 of file cint_evaluator.h.

## 7.16 coconut_config.h File Reference

This graph shows which files directly or indirectly include this file:



### Defines

- #define PURE_VIRTUAL { throw ”Pure virtual function called!”; }

### 7.16.1 Define Documentation

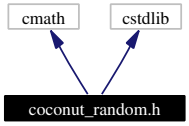
#### 7.16.1.1 #define PURE_VIRTUAL { throw "Pure virtual function called!"; }

Definition at line 6 of file coconut_config.h.

## 7.17 coconut_random.h File Reference

```
#include <cmath>
#include <cstdlib>
```

Include dependency graph for coconut_random.h:




### Defines

- #define DEBUG_RANDOM 1
- #define coconut_random random
- #define coconut_seed(A) srandom(A)
- #define COCONUT_RAND_MAX RAND_MAX
- #define COCONUT_RRAND_MIN -1.e08
- #define COCONUT_RRAND_MAX +1.e08
- #define INIT_SEED 1
- #define coconut_init_random() srandom(INIT_SEED)

### Typedefs

- typedef long int rand_t

### Functions

- double d_random ()
- double r_random (double l, double u)
- double r_random (const interval &_i)

### 7.17.1 Detailed Description

Definition in file coconut_random.h.

### 7.17.2 Define Documentation

#### 7.17.2.1 #define coconut_init_random() srandom(INIT_SEED)

Definition at line 50 of file coconut_random.h.

#### 7.17.2.2 #define COCONUT_RAND_MAX RAND_MAX

Definition at line 39 of file coconut_random.h.

#### 7.17.2.3 #define coconut_random random

Definition at line 37 of file coconut_random.h.

#### 7.17.2.4 #define COCONUT_RRAND_MAX +1.e08

Definition at line 42 of file coconut_random.h.

#### 7.17.2.5 #define COCONUT_RRAND_MIN -1.e08

Definition at line 41 of file coconut_random.h.

#### 7.17.2.6 #define coconut_seed(A) srandom(A)

Definition at line 38 of file coconut_random.h.

#### 7.17.2.7 #define DEBUG_RANDOM 1

Definition at line 31 of file coconut_random.h.

#### 7.17.2.8 #define INIT_SEED 1

Definition at line 45 of file coconut_random.h.

### 7.17.3 Typedef Documentation

#### 7.17.3.1 typedef long int rand_t

Definition at line 52 of file coconut_random.h.

### 7.17.4 Function Documentation

#### 7.17.4.1 double d_random () [inline]

Definition at line 54 of file coconut_random.h.

#### 7.17.4.2 double r_random (const interval & *_i*) [inline]

Definition at line 93 of file coconut_random.h.

#### 7.17.4.3 double r_random (double *l*, double *u*) [inline]

Definition at line 65 of file coconut_random.h.

## 7.18 coconut_types.h File Reference

### Typedefs

- typedef enum tristate_e tristate

### Enumerations

- enum tristate_e { t_true = 1, t_false = -1, t_maybe = 0 }

#### 7.18.1 Detailed Description

Definition in file coconut_types.h.

#### 7.18.2 Typedef Documentation

##### 7.18.2.1 typedef enum tristate_e tristate

#### 7.18.3 Enumeration Type Documentation

##### 7.18.3.1 enum tristate_e

**Enumeration values:**

**t_true**

**t_false**

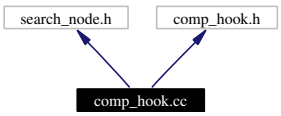
**t_maybe**

Definition at line 30 of file coconut_types.h.

## 7.19 comp_hook.cc File Reference

```
#include <search_node.h>
#include <comp_hook.h>
```

Include dependency graph for comp_hook.cc:



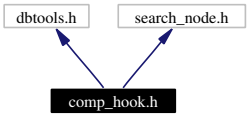

#### 7.19.1 Detailed Description

Definition in file comp_hook.cc.

## 7.20 comp_hook.h File Reference

```
#include <dbtools.h>
#include <search_node.h>
```

Include dependency graph for comp_hook.h:




### Compounds

- class work_node_comp_hook

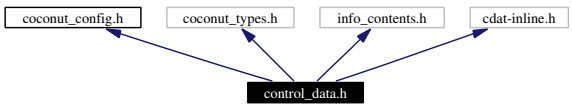
#### 7.20.1 Detailed Description

Definition in file comp_hook.h.

## 7.21 control_data.h File Reference

```
#include <coconut_config.h>
#include <coconut_types.h>
#include <info_contents.h>
#include <cdat-inline.h>
```

Include dependency graph for control_data.h:




### Compounds

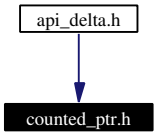
- class control_data

#### 7.21.1 Detailed Description

Definition in file control_data.h.

## 7.22 counted_ptr.h File Reference

This graph shows which files directly or indirectly include this file:



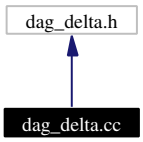

**Compounds**

- class counted_ptr
- struct **counter**

## 7.23 dag_delta.cc File Reference

`#include <dag_delta.h>`

Include dependency graph for dag_delta.cc:



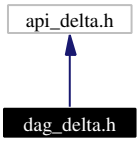

### 7.23.1 Detailed Description

Definition in file dag_delta.cc.

## 7.24 dag_delta.h File Reference

`#include <api_delta.h>`

Include dependency graph for dag_delta.h:




**Compounds**

- class **__check_nodes**
- struct **__docompare_nodes**
- class dag_delta
- class **__check_walkers**
- class dag_undelta

### 7.24.1 Detailed Description

Definition in file dag_delta.h.

## 7.25 datamap-inline.h File Reference

**Defines**

- #define __DATAMAP_SEPARATOR "%"
- #define __DATAMAP_IDXSPEC "IDX"

### 7.25.1 Detailed Description

Definition in file datamap-inline.h.

### 7.25.2 Define Documentation

#### 7.25.2.1 #define __DATAMAP_IDXSPEC "IDX"

Definition at line 31 of file datamap-inline.h.

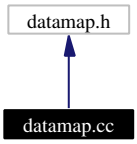
#### 7.25.2.2 #define __DATAMAP_SEPARATOR "%"

Definition at line 30 of file datamap-inline.h.

## 7.26 datamap.cc File Reference

```
#include <datamap.h>
```

Include dependency graph for datamap.cc:




### 7.26.1 Detailed Description

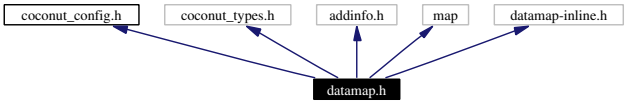
Definition in file datamap.cc.

## 7.27 datamap.h File Reference

```
#include <coconut_config.h>
#include <coconut_types.h>
#include <addinfo.h>
#include <map>
```

`#include <datamap-inline.h>`

Include dependency graph for datamap.h:




### Compounds

- class datamap

#### 7.27.1 Detailed Description

Definition in file datamap.h.

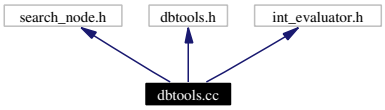
## 7.28 dbtools.cc File Reference

`#include <search_node.h>`

`#include <dbtools.h>`

`#include <int_evaluator.h>`

Include dependency graph for dbtools.cc:




#### 7.28.1 Detailed Description

Definition in file dbtools.cc.

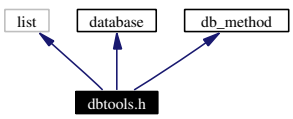
## 7.29 dbtools.h File Reference

`#include <list>`

`#include <database>`

`#include <db_method>`

Include dependency graph for dbtools.h:



### Compounds

- class box_check_intersection
- class point_check_feasibility
- class work_node_context

### Typedefs

- typedef std::list< vdbl::col_spec > dbt_row

### Functions

- template<class _C> void add_to_dbt_row (dbt_row &dbr, const std::string &nm, const _C &cont)
- template<class _C> void add_to_dbt_row (dbt_row &dbr, const char *nm, const _C &cont)

#### 7.29.1 Detailed Description

Definition in file dbtools.h.

#### 7.29.2 Typedef Documentation

##### 7.29.2.1 typedef std::list<vdbl::col_spec> dbt_row

Definition at line 103 of file dbtools.h.

#### 7.29.3 Function Documentation

##### 7.29.3.1 template<class _C> void add_to_dbt_row (dbt_row & *dbr*, const char * *nm*, const _C & *cont*) [inline]

Definition at line 112 of file dbtools.h.

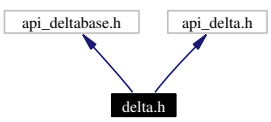
##### 7.29.3.2 template<class _C> void add_to_dbt_row (dbt_row & *dbr*, const std::string & *nm*, const _C & *cont*) [inline]

Definition at line 106 of file dbtools.h.

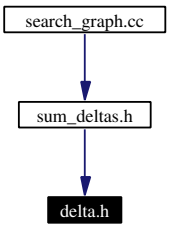
## 7.30 delta.h File Reference

```
#include <api_deltabase.h>
#include <api_delta.h>
```

Include dependency graph for delta.h:



This graph shows which files directly or indirectly include this file:



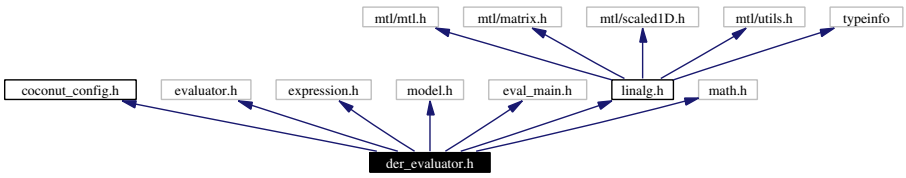

### 7.30.1 Detailed Description

Definition in file delta.h.

## 7.31 der_evaluator.h File Reference

```
#include <coconut_config.h>
#include <evaluator.h>
#include <expression.h>
#include <model.h>
#include <eval_main.h>
#include <linalg.h>
#include <math.h>
```

Include dependency graph for der_evaluator.h:




### Compounds

- class der_eval
- struct der_eval_type
- class func_d_eval
- struct func_d_eval_type
- class prep_d_eval

### Typedefs

- typedef bool(∗ prep_d_evaluator )()

- typedef double(* func_d_evaluator )(const std::vector< double > *__x, const variable_indicator &__v, std::vector< double > &__d_data)
- typedef std::vector< double > &(* der_evaluator )(const std::vector< double > &__d_dat, const variable_indicator &__v)

### 7.31.1 Detailed Description

Definition in file der_evaluator.h.

### 7.31.2 Typedef Documentation

#### 7.31.2.1 typedef std::vector<double>&(* der_evaluator)(const std::vector<double>& __d_dat, const variable_indicator& __v)

Definition at line 44 of file der_evaluator.h.

#### 7.31.2.2 typedef double(* func_d_evaluator)(const std::vector<double>* __x, const variable_indicator& __v, std::vector<double>& __d_data)

Definition at line 41 of file der_evaluator.h.

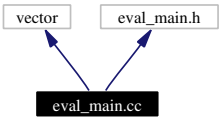
#### 7.31.2.3 typedef bool(* prep_d_evaluator)()

Definition at line 40 of file der_evaluator.h.

## 7.32 eval_main.cc File Reference

```
#include <vector>
#include <eval_main.h>
```

Include dependency graph for eval_main.cc:




### Functions

- std::vector< std::vector< void * > > standard_evaluators (NUM_EVALUATORS)

### 7.32.1 Detailed Description

Definition in file eval_main.cc.

### 7.32.2 Function Documentation

#### 7.32.2.1 std::vector<std::vector<void *> > standard_evaluators (NUM_EVALUATORS)

## 7.33 eval_main.h File Reference

### Defines

- #define FUNC_EVALUATOR 0
- #define FUNC_D_EVALUATOR 1
- #define FUNC_RANGE 2
- #define FUNC_D_RANGE 3
- #define DER_EVALUATOR 4
- #define DER_RANGE 5
- #define REDUCE_RANGE 6
- #define PRINT 7
- #define FUNC_FW_PREP 8
- #define FUNC_ISLP_EVALUATOR 9
- #define ISLP_EVALUATOR 10
- #define FUNC_ID_EVALUATOR 11
- #define IDER_EVALUATOR 12
- #define FUNC_CRANGE 13
- #define FUNC_BRANGE 14
- #define FUNC_ANALYTICD 15
- #define FUNC_INFB 16
- #define CONVEX_EVALUATOR 17
- #define NUM_EVALUATORS 18

### Variables

- std::vector< std::vector< void * > > standard_evaluators

### 7.33.1 Detailed Description

Definition in file eval_main.h.

### 7.33.2 Define Documentation

#### 7.33.2.1 #define CONVEX_EVALUATOR 17

Definition at line 47 of file eval_main.h.

#### 7.33.2.2 #define DER_EVALUATOR 4

Definition at line 34 of file eval_main.h.

#### 7.33.2.3 #define DER_RANGE 5

Definition at line 35 of file eval_main.h.

#### 7.33.2.4 #define FUNC_ANALYTICD 15

Definition at line 45 of file eval_main.h.

#### 7.33.2.5 #define FUNC_BRANGE 14

Definition at line 44 of file eval_main.h.

#### 7.33.2.6 #define FUNC_CRANGE 13

Definition at line 43 of file eval_main.h.

#### 7.33.2.7 #define FUNC_D_EVALUATOR 1

Definition at line 31 of file eval_main.h.

#### 7.33.2.8 #define FUNC_D_RANGE 3

Definition at line 33 of file eval_main.h.

#### 7.33.2.9 #define FUNC_EVALUATOR 0

Definition at line 30 of file eval_main.h.

#### 7.33.2.10 #define FUNC_FW_PREP 8

Definition at line 38 of file eval_main.h.

#### 7.33.2.11 #define FUNC_ID_EVALUATOR 11

Definition at line 41 of file eval_main.h.

#### 7.33.2.12 #define FUNC_INFB 16

Definition at line 46 of file eval_main.h.

#### 7.33.2.13 #define FUNC_ISLP_EVALUATOR 9

Definition at line 39 of file eval_main.h.

#### 7.33.2.14 #define FUNC_RANGE 2

Definition at line 32 of file eval_main.h.

#### 7.33.2.15 #define IDER_EVALUATOR 12

Definition at line 42 of file eval_main.h.

#### 7.33.2.16 #define ISLP_EVALUATOR 10

Definition at line 40 of file eval_main.h.

#### 7.33.2.17 #define NUM_EVALUATORS 18

Definition at line 49 of file eval_main.h.

#### 7.33.2.18 #define PRINT 7

Definition at line 37 of file eval_main.h.

#### 7.33.2.19 #define REDUCE_RANGE 6

Definition at line 36 of file eval_main.h.

### 7.33.3 Variable Documentation

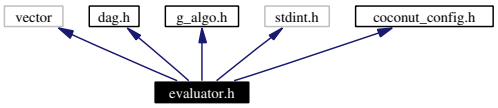
#### 7.33.3.1 std::vector<std::vector<void *> > standard_evaluators

Definition at line 51 of file eval_main.h.

## 7.34 evaluator.h File Reference

```
#include <vector>
#include <dag.h>
#include <g_algo.h>
#include <stdint.h>
#include <coconut_config.h>
```

Include dependency graph for evaluator.h:




### Compounds

- class _evaluator_base
- class backward_evaluator_base
- class cached_backward_evaluator_base
- class cached_evaluator_base
- class cached_forward_evaluator_base
- class evaluator_base
- class forward_evaluator_base
- class variable_indicator

### Functions

- template<class _Walker, class _Visitor> _Visitor::return_value recursive_short_cut_walk (_Walker __w, _Visitor __f)
- template<class _Walker, class _Visitor> _Visitor::return_value _recursive_short_cut_walk (_Walker __w, _Visitor __f)
- template<class _Visitor, class _Walker> _Visitor::return_value evaluate (_Visitor __v, _Walker __start)

### 7.34.1 Detailed Description

Definition in file evaluator.h.

### 7.34.2 Function Documentation

#### 7.34.2.1 template<class _Walker, class _Visitor> _Visitor::return_value _recursive_short_cut_walk (_Walker __w, _Visitor __f)

Definition at line 546 of file evaluator.h.

#### 7.34.2.2 template<class _Visitor, class _Walker> _Visitor::return_value evaluate (_Visitor __v, _Walker __start)

Definition at line 588 of file evaluator.h.

#### 7.34.2.3 template<class _Walker, class _Visitor> _Visitor::return_value recursive_short_cut_walk (_Walker __w, _Visitor __f)

Definition at line 486 of file evaluator.h.

## 7.35 expr-inline.h File Reference

### Compounds

- class children_compare
- class expression_print_visitor
- class parents_compare
- class parents_compare_eq

### Functions

- std::ostream & __wr_interval (std::ostream &o, const interval &__i)

### 7.35.1 Detailed Description

Definition in file expr-inline.h.

### 7.35.2 Function Documentation

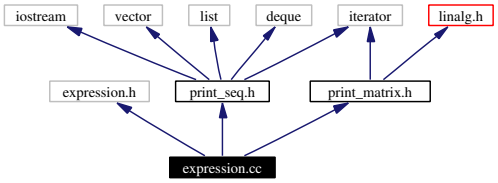
#### 7.35.2.1 std::ostream& __wr_interval (std::ostream & *o*, const interval & *__i*) [inline]

Definition at line 252 of file expr-inline.h.

## 7.36 expression.cc File Reference

```
#include <expression.h>
#include <print_seq.h>
#include <print_matrix.h>
```

Include dependency graph for expression.cc:




### Functions

- std::ostream & operator<< (std::ostream &o, const expression_node &_x)

### Variables

- const char * expr_names [ ]

#### 7.36.1 Detailed Description

Definition in file expression.cc.

#### 7.36.2 Function Documentation

##### 7.36.2.1 std::ostream& operator<< (std::ostream & *o*, const expression_node & *_x*)

Definition at line 195 of file expression.cc.

#### 7.36.3 Variable Documentation

##### 7.36.3.1 const char* expr_names[ ]

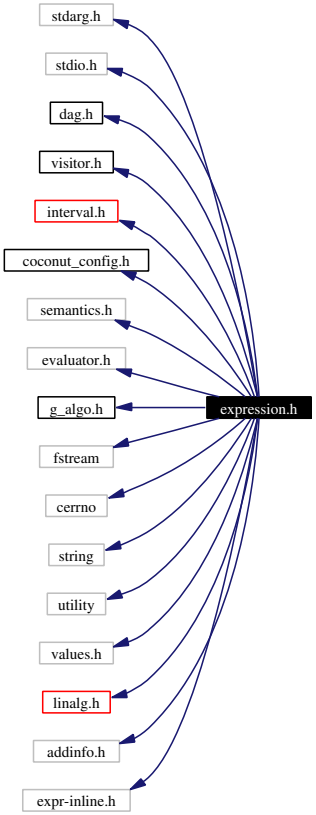
Definition at line 31 of file expression.cc.

## 7.37 expression.h File Reference

```
#include <stdarg.h>
#include <stdio.h>
#include <dag.h>
#include <visitor.h>
#include <interval.h>
#include <coconut_config.h>
#include <semantics.h>
#include <evaluator.h>
#include <g_algo.h>
#include <fstream>
```

```
#include <cerrno>
#include <string>
#include <utility>
#include <values.h>
#include <linalg.h>
#include <addinfo.h>
#include <expr-inline.h>
```

Include dependency graph for expression.h:




## Compounds

- class expression_node

## Defines

- #define EXPRINFO_GHOST 0
- #define EXPRINFO_CONSTANT -1
- #define EXPRINFO_VARIABLE -2
- #define EXPRINFO_SUM -3
- #define EXPRINFO_MEAN -4
- #define EXPRINFO_PROD -5

- #define EXPRINFO_MAX -6
- #define EXPRINFO_MIN -7
- #define EXPRINFO_MONOME -8
- #define EXPRINFO_SCPROD -9
- #define EXPRINFO_NORM -10
- #define EXPRINFO_INVERT -11
- #define EXPRINFO_SQUARE -12
- #define EXPRINFO_SQROOT -13
- #define EXPRINFO_ABS -14
- #define EXPRINFO_INTPOWER -15
- #define EXPRINFO_EXP -16
- #define EXPRINFO_LOG -17
- #define EXPRINFO_SIN -18
- #define EXPRINFO_COS -19
- #define EXPRINFO_GAUSS -20
- #define EXPRINFO_POLY -21
- #define EXPRINFO_POW -22
- #define EXPRINFO_DIV -23
- #define EXPRINFO_ATAN2 -24
- #define EXPRINFO_LIN -25
- #define EXPRINFO_QUAD -26
- #define EXPRINFO_RE -27
- #define EXPRINFO_IM -28
- #define EXPRINFO_ARG -29
- #define EXPRINFO_CPLXCONJ -30
- #define EXPRINFO_LOOKUP -31
- #define EXPRINFO_PWLIN -32
- #define EXPRINFO_SPLINE -33
- #define EXPRINFO_PWCONSTLC -34
- #define EXPRINFO_PWCONSTRC -35
- #define EXPRINFO_IN -36
- #define EXPRINFO_IF -37
- #define EXPRINFO_AND -38
- #define EXPRINFO_OR -39
- #define EXPRINFO_NOT -40
- #define EXPRINFO_IMPLIES -41
- #define EXPRINFO_COUNT -42
- #define EXPRINFO_ALLDIFF -43
- #define EXPRINFO_HISTOGRAM -44
- #define EXPRINFO_LEVEL -45
- #define EXPRINFO_NEIGHBOR -46
- #define EXPRINFO_NOGOOD -47
- #define EXPRINFO_EXPECTATION -48
- #define EXPRINFO_INTEGRAL -49
- #define EXPRINFO_DET -50
- #define EXPRINFO_COND -51
- #define EXPRINFO_PSD -52
- #define EXPRINFO_MPROD -53
- #define EXPRINFO_FEM -54
- #define EXPRINFO_CMPROD -55

- #define EXPRINFO_CGFEM -56
- #define EXPRINFO_UNDEFINED -57
- #define EXPRINFO_NUMOFPREDEF -(EXPRINFO_UNDEFINED)
- #define EXPR_LASTARG NULL

### Typedefs

- typedef interval rhs_t
- typedef std::vector< void * > evaluator_v

### Enumerations

- enum { ex_bound = 1, ex_linear = 1<<1, ex_quadratic = 1<<2, ex_polynomial = 1<<3, ex_other = 1<<4, ex_kj = 1<<7, ex_org = 1<<8, ex_redundant = 1<<9, ex_notredundant = 1<<10, ex_active_lo = 1<<11, ex_inactive_lo = 1<<12, ex_active_hi = 1<<13, ex_inactive_hi = 1<<14, ex_active = ex_active_lo|ex_active_hi, ex_inactive = ex_inactive_lo|ex_inactive_hi, ex_integer = 1<<15, ex_exists = 1<<16, ex_forall = 1<<17, ex_free = 1<<18, ex_stochastic = 1<<19, ex_convex = 1<<20, ex_concave = 1<<21, ex_inequality = 1<<28, ex_equality = 1<<29, ex_leftbound = 1<<30, ex_rightbound = 1<<31, ex_atmlin = ex_bound|ex_linear, ex_atmquad = ex_atmlin|ex_quadratic, ex_atmpoly = ex_atmquad|ex_polynomial, ex_nonlin = ex_quadratic|ex_polynomial|ex_other, ex_nonbnd = ex_linear|ex_nonlin, ex_any = ex_atmlin|ex_nonlin, ex_bothbound = ex_leftbound|ex_rightbound }

#### 7.37.1 Detailed Description

Definition in file expression.h.

#### 7.37.2 Define Documentation

##### 7.37.2.1 #define EXPR_LASTARG NULL

Definition at line 148 of file expression.h.

##### 7.37.2.2 #define EXPRINFO_ABS -14

Definition at line 76 of file expression.h.

##### 7.37.2.3 #define EXPRINFO_ALLDIFF -43

Definition at line 118 of file expression.h.

##### 7.37.2.4 #define EXPRINFO_AND -38

Definition at line 113 of file expression.h.

##### 7.37.2.5 #define EXPRINFO_ARG -29

Definition at line 98 of file expression.h.

#### 7.37.2.6 #define EXPRINFO_ATAN2 -24

Definition at line 88 of file expression.h.

#### 7.37.2.7 #define EXPRINFO_CGFEM -56

Definition at line 142 of file expression.h.

#### 7.37.2.8 #define EXPRINFO_CMPROD -55

Definition at line 141 of file expression.h.

#### 7.37.2.9 #define EXPRINFO_COND -51

Definition at line 135 of file expression.h.

#### 7.37.2.10 #define EXPRINFO_CONSTANT -1

Definition at line 57 of file expression.h.

#### 7.37.2.11 #define EXPRINFO_COS -19

Definition at line 81 of file expression.h.

#### 7.37.2.12 #define EXPRINFO_COUNT -42

Definition at line 117 of file expression.h.

#### 7.37.2.13 #define EXPRINFO_CPLXCONJ -30

Definition at line 101 of file expression.h.

#### 7.37.2.14 #define EXPRINFO_DET -50

Definition at line 134 of file expression.h.

#### 7.37.2.15 #define EXPRINFO_DIV -23

Definition at line 87 of file expression.h.

#### 7.37.2.16 #define EXPRINFO_EXP -16

Definition at line 78 of file expression.h.

#### 7.37.2.17 #define EXPRINFO_EXPECTATION -48

Definition at line 127 of file expression.h.

#### 7.37.2.18 #define EXPRINFO_FEM -54

Definition at line 138 of file expression.h.

#### 7.37.2.19 #define EXPRINFO_GAUSS -20

Definition at line 82 of file expression.h.

#### 7.37.2.20 #define EXPRINFO_GHOST 0

Definition at line 55 of file expression.h.

#### 7.37.2.21 #define EXPRINFO_HISTOGRAM -44

Definition at line 119 of file expression.h.

#### 7.37.2.22 #define EXPRINFO_IF -37

Definition at line 112 of file expression.h.

#### 7.37.2.23 #define EXPRINFO_IM -28

Definition at line 97 of file expression.h.

#### 7.37.2.24 #define EXPRINFO_IMPLIES -41

Definition at line 116 of file expression.h.

#### 7.37.2.25 #define EXPRINFO_IN -36

Definition at line 111 of file expression.h.

#### 7.37.2.26 #define EXPRINFO_INTEGRAL -49

Definition at line 128 of file expression.h.

#### 7.37.2.27 #define EXPRINFO_INTPOWER -15

Definition at line 77 of file expression.h.

#### 7.37.2.28 #define EXPRINFO_INVERT -11

Definition at line 73 of file expression.h.

#### 7.37.2.29 #define EXPRINFO_LEVEL -45

Definition at line 120 of file expression.h.

#### 7.37.2.30 #define EXPRINFO_LIN -25

Definition at line 91 of file expression.h.

#### 7.37.2.31 #define EXPRINFO_LOG -17

Definition at line 79 of file expression.h.

#### 7.37.2.32 #define EXPRINFO_LOOKUP -31

Definition at line 104 of file expression.h.

#### 7.37.2.33 #define EXPRINFO_MAX -6

Definition at line 66 of file expression.h.

#### 7.37.2.34 #define EXPRINFO_MEAN -4

Definition at line 64 of file expression.h.

#### 7.37.2.35 #define EXPRINFO_MIN -7

Definition at line 67 of file expression.h.

#### 7.37.2.36 #define EXPRINFO_MONOME -8

Definition at line 68 of file expression.h.

#### 7.37.2.37 #define EXPRINFO_MPROD -53

Definition at line 137 of file expression.h.

#### 7.37.2.38 #define EXPRINFO_NEIGHBOR -46

Definition at line 123 of file expression.h.

#### 7.37.2.39 #define EXPRINFO_NOGOOD -47

Definition at line 124 of file expression.h.

#### 7.37.2.40 #define EXPRINFO_NORM -10

Definition at line 70 of file expression.h.

#### 7.37.2.41 #define EXPRINFO_NOT -40

Definition at line 115 of file expression.h.

#### 7.37.2.42 #define EXPRINFO_NUMOFPREDEF -(EXPRINFO_UNDEFINED)

Definition at line 146 of file expression.h.

#### 7.37.2.43 #define EXPRINFO_OR -39

Definition at line 114 of file expression.h.

#### 7.37.2.44 #define EXPRINFO_POLY -21

Definition at line 83 of file expression.h.

#### 7.37.2.45 #define EXPRINFO_POW -22

Definition at line 86 of file expression.h.

#### 7.37.2.46 #define EXPRINFO_PROD -5

Definition at line 65 of file expression.h.

#### 7.37.2.47 #define EXPRINFO_PSD -52

Definition at line 136 of file expression.h.

#### 7.37.2.48 #define EXPRINFO_PWCONSTLC -34

Definition at line 107 of file expression.h.

#### 7.37.2.49 #define EXPRINFO_PWCONSTRC -35

Definition at line 108 of file expression.h.

#### 7.37.2.50 #define EXPRINFO_PWLIN -32

Definition at line 105 of file expression.h.

#### 7.37.2.51 #define EXPRINFO_QUAD -26

Definition at line 92 of file expression.h.

#### 7.37.2.52 #define EXPRINFO_RE -27

Definition at line 96 of file expression.h.

#### 7.37.2.53 #define EXPRINFO_SCPROD -9

Definition at line 69 of file expression.h.

#### 7.37.2.54 #define EXPRINFO_SIN -18

Definition at line 80 of file expression.h.

#### 7.37.2.55 #define EXPRINFO_SPLINE -33

Definition at line 106 of file expression.h.

#### 7.37.2.56 #define EXPRINFO_SQROOT -13

Definition at line 75 of file expression.h.

#### 7.37.2.57 #define EXPRINFO_SQUARE -12

Definition at line 74 of file expression.h.

#### 7.37.2.58 #define EXPRINFO_SUM -3

Definition at line 63 of file expression.h.

#### 7.37.2.59 #define EXPRINFO_UNDEFINED -57

Definition at line 145 of file expression.h.

#### 7.37.2.60 #define EXPRINFO_VARIABLE -2

Definition at line 58 of file expression.h.

### 7.37.3 Typedef Documentation

#### 7.37.3.1 typedef std::vector<void*> evaluator_v

Definition at line 200 of file expression.h.

#### 7.37.3.2 typedef interval rhs_t

Definition at line 198 of file expression.h.

### 7.37.4 Enumeration Type Documentation

#### 7.37.4.1 anonymous enum

**Enumeration values:**

- **ex_bound**
- **ex_linear**
- **ex_quadratic**
- **ex_polynomial**
- **ex_other**
- **ex_kj**
- **ex_org**
- **ex_redundant**
- **ex_notredundant**
- **ex_active_lo**
- **ex_inactive_lo**
- **ex_active_hi**
- **ex_inactive_hi**
- **ex_active**
- **ex_inactive**
- **ex_integer**
- **ex_exists**
- **ex_forall**
- **ex_free**
- **ex_stochastic**

**ex_convex**
**ex_concave**
**ex_inequality**
**ex_equality**
**ex_leftbound**
**ex_rightbound**
**ex_atmlin**
**ex_atmquad**
**ex_atmpoly**
**ex_nonlin**
**ex_nonbnd**
**ex_any**
**ex_bothbound**

Definition at line 153 of file expression.h.

## 7.38 exprnames.cc File Reference

`#include <stdlib.h>`

Include dependency graph for exprnames.cc:


### Variables

- const char * expr_names [ ]

### 7.38.1 Detailed Description

Definition in file exprnames.cc.

### 7.38.2 Variable Documentation

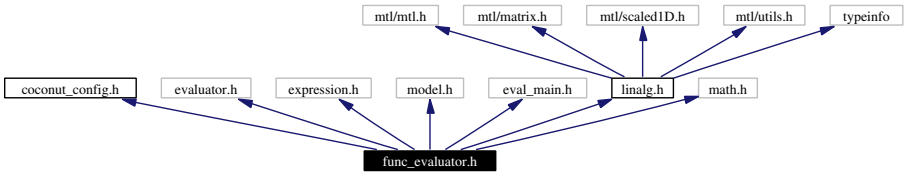
#### 7.38.2.1 const char* expr_names[ ]

Definition at line 29 of file exprnames.cc.

## 7.39 func_evaluator.h File Reference

```
#include <coconut_config.h>
#include <evaluator.h>
#include <expression.h>
#include <model.h>
#include <eval_main.h>
#include <linalg.h>
#include <math.h>
```

Include dependency graph for func_evaluator.h:




### Compounds

- class func_eval
- struct func_eval_type

### Typedefs

- typedef double(∗ func_evaluator )(const std::vector< double > ∗__x, const variable_indicator &__v)

#### 7.39.1 Detailed Description

Definition in file func_evaluator.h.

#### 7.39.2 Typedef Documentation

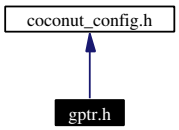
##### 7.39.2.1 typedef double(∗ func_evaluator)(const std::vector<double>∗ __x, const variable_indicator& __v)

Definition at line 40 of file func_evaluator.h.

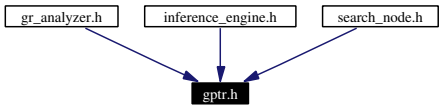
## 7.40 gptr.h File Reference

```
#include <coconut_config.h>
```

Include dependency graph for gptr.h:




This graph shows which files directly or indirectly include this file:



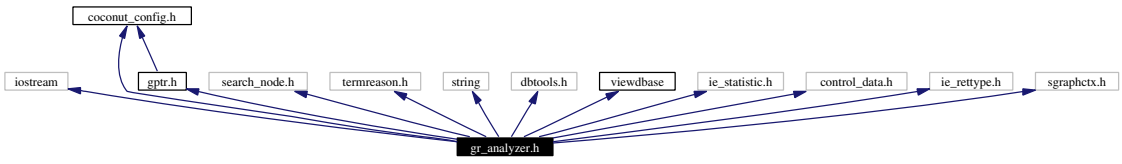

### Compounds

- class gptr
- class ptr

## 7.41 gr_analyzer.h File Reference

```
#include <iostream>
#include <coconut_config.h>
#include <search_node.h>
#include <termreason.h>
#include <gptr.h>
#include <string>
#include <dbtools.h>
#include <viewdbase>
#include <ie_statistic.h>
#include <control_data.h>
#include <ie_rettype.h>
#include <sgraphctx.h>
```

Include dependency graph for gr_analyzer.h:




### Compounds

- class graph_analyzer

### Typedefs

- typedef ie_return_type ga_return_type

#### 7.41.1 Detailed Description

Definition in file gr_analyzer.h.

#### 7.41.2 Typedef Documentation

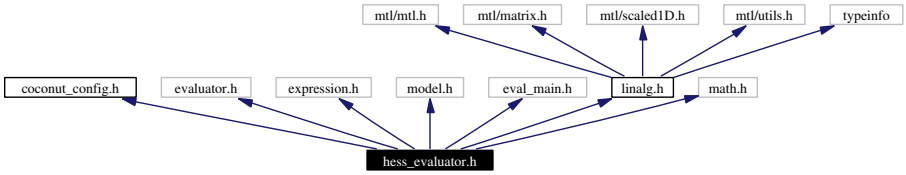
##### 7.41.2.1 typedef ie_return_type ga_return_type

Definition at line 45 of file gr_analyzer.h.

## 7.42 hess_evaluator.h File Reference

```
#include <coconut_config.h>
#include <evaluator.h>
#include <expression.h>
#include <model.h>
#include <eval_main.h>
#include <linalg.h>
#include <math.h>
```

Include dependency graph for hess_evaluator.h:




### Compounds

- class der_eval
- struct der_eval_type
- class func_h_eval
- struct func_h_eval_ret
- struct func_h_eval_type
- struct prep_h_eval_tp

### Typedefs

- typedef bool(* prep_h_evaluator )()
- typedef double(* func_d_evaluator )(const std::vector< double > *__x, const variable_indicator &__v, std::vector< double > &__d_data)

- typedef std::vector< double > &(* der_evaluator )(const std::vector< double > &__d_dat, const variable_indicator &__v)

**Functions**

- prep_h_eval (std::vector< std::vector< double > > &__d, std::vector< matrix< double > > &__h, unsigned int _num_of_nodes)
- prep_h_eval (const prep_h_eval &__x)
- ~prep_h_eval ()
- void initialize ()
- bool is_cached (const expression_node &__data)
- void retrieve_from_cache (const expression_node &__data)
- int initialize (const expression_node &__data)
- void calculate (const expression_node &__data)
- int update (bool __rval)
- int update (const expression_node &__data, bool __rval)
- bool calculate_value (bool eval_all)

**Variables**

- prep_h_eval_tp prep_h_eval_tp
- prep_h_eval_tp expression_node
- prep_h_eval_tp bool
- prep_h_eval_tp _Base

### 7.42.1 Detailed Description

Definition in file hess_evaluator.h.

### 7.42.2 Typedef Documentation

#### 7.42.2.1 typedef std::vector<double>&(* der_evaluator)(const std::vector<double>& __d_dat, const variable_indicator& __v)

Definition at line 46 of file hess_evaluator.h.

#### 7.42.2.2 typedef double(* func_d_evaluator)(const std::vector<double>* __x, const variable_indicator& __v, std::vector<double>& __d_data)

Definition at line 43 of file hess_evaluator.h.

#### 7.42.2.3 typedef bool(* prep_h_evaluator)()

Definition at line 42 of file hess_evaluator.h.

### 7.42.3 Function Documentation

#### 7.42.3.1 void calculate (const expression_node & *__data*)

Definition at line 100 of file hess_evaluator.h.

#### 7.42.3.2 bool calculate_value (bool *eval_all*)

Definition at line 107 of file hess_evaluator.h.

#### 7.42.3.3 int initialize (const expression_node & *__data*)

Definition at line 91 of file hess_evaluator.h.

#### 7.42.3.4 void initialize ()

Definition at line 82 of file hess_evaluator.h.

#### 7.42.3.5 bool is_cached (const expression_node & *__data*)

Definition at line 84 of file hess_evaluator.h.

#### 7.42.3.6 prep_h_eval (const prep_h_eval & *__x*)

Definition at line 77 of file hess_evaluator.h.

#### 7.42.3.7 prep_h_eval (std::vector< std::vector< double > > & *__d*, std::vector< matrix< double > > & *__h*, unsigned int *_num_of_nodes*)

Definition at line 63 of file hess_evaluator.h.

#### 7.42.3.8 void retrieve_from_cache (const expression_node & *__data*)

Definition at line 89 of file hess_evaluator.h.

#### 7.42.3.9 int update (const expression_node & *__data*, bool *__rval*)

Definition at line 104 of file hess_evaluator.h.

#### 7.42.3.10 int update (bool *__rval*)

Definition at line 102 of file hess_evaluator.h.

#### 7.42.3.11 ~prep_h_eval ()

Definition at line 80 of file hess_evaluator.h.

### 7.42.4 Variable Documentation

#### 7.42.4.1 struct prep_h_eval_tp _Base

#### 7.42.4.2 struct prep_h_eval_tp bool

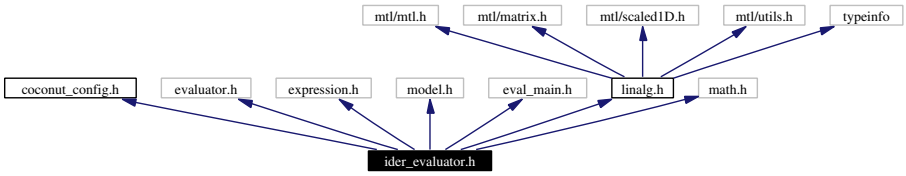
#### 7.42.4.3 struct prep_h_eval_tp expression_node

#### 7.42.4.4 struct prep_h_eval_tp prep_h_eval_tp

## 7.43 ider_evaluator.h File Reference

```
#include <coconut_config.h>
#include <evaluator.h>
#include <expression.h>
#include <model.h>
#include <eval_main.h>
#include <linalg.h>
#include <math.h>
```

Include dependency graph for ider_evaluator.h:




### Compounds

- class func_id_eval
- struct func_id_eval_type
- class ider_eval
- struct ider_eval_type
- class prep_id_eval

### Typedefs

- typedef bool(∗ prep_id_evaluator )()
- typedef interval(∗ func_id_evaluator )(const std::vector< interval > ∗__x, const variable_indicator &__v, std::vector< interval > &__id_data)
- typedef std::vector< interval > &(∗ ider_evaluator )(const std::vector< interval > &__d_dat, const variable_indicator &__v)

#### 7.43.1 Detailed Description

Definition in file ider_evaluator.h.

#### 7.43.2 Typedef Documentation

##### 7.43.2.1 typedef interval(∗ func_id_evaluator)( const std::vector<interval>∗ __x, const variable_indicator& __v, std::vector<interval>& __id_data)

Definition at line 41 of file ider_evaluator.h.

#### 7.43.2.2 typedef std::vector<interval>&(* ider_evaluator)(const std::vector<interval>& __d_dat, const variable_indicator& __v)

Definition at line 44 of file ider_evaluator.h.

#### 7.43.2.3 typedef bool(* prep_id_evaluator)()

Definition at line 40 of file ider_evaluator.h.

## 7.44 ie_rettype.cc File Reference

```
#include <ie_rettype.h>
```

Include dependency graph for ie_rettype.cc:

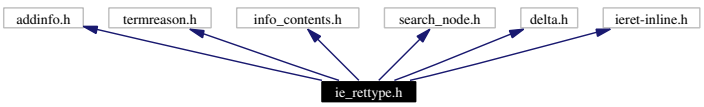

### 7.44.1 Detailed Description

Definition in file ie_rettype.cc.

## 7.45 ie_rettype.h File Reference

```
#include <addinfo.h>
#include <termreason.h>
#include <info_contents.h>
#include <search_node.h>
#include <delta.h>
#include <ieret-inline.h>
```

Include dependency graph for ie_rettype.h:




### Compounds

- class ie_return_type

### 7.45.1 Detailed Description

Definition in file ie_rettype.h.

## 7.46 ie_statistic.h File Reference

### Compounds

- class statistic_info

### 7.46.1 Detailed Description

Definition in file ie_statistic.h.

## 7.47 ieret-inline.h File Reference

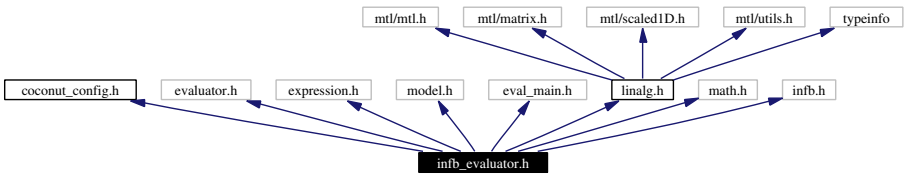
### 7.47.1 Detailed Description

Definition in file ieret-inline.h.

## 7.48 infb_evaluator.h File Reference

```
#include <coconut_config.h>
#include <evaluator.h>
#include <expression.h>
#include <model.h>
#include <eval_main.h>
#include <linalg.h>
#include <math.h>
#include <infb.h>
```

Include dependency graph for infb_evaluator.h:




### Compounds

- class infbound_eval
- struct infbound_eval_type

### Typedefs

- typedef infbound(* infbound_evaluator )(const std::vector< infbound > *__x, const variable_indicator &__v)

#### 7.48.1 Detailed Description

Definition in file infb_evaluator.h.

#### 7.48.2 Typedef Documentation

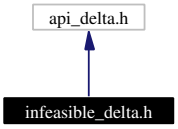
**7.48.2.1 typedef infbound(* infbound_evaluator)(const std::vector<infbound>* __x, const variable_indicator& __v)**

Definition at line 41 of file infb_evaluator.h.

## 7.49 infeasible_delta.h File Reference

```
#include <api_delta.h>
```

Include dependency graph for infeasible_delta.h:




### Compounds

- class infeasible_delta
- class infeasible_undelta

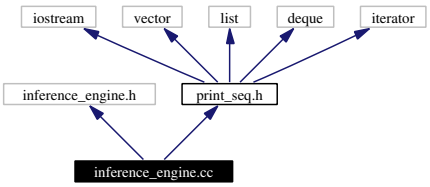
#### 7.49.1 Detailed Description

Definition in file infeasible_delta.h.

## 7.50 inference_engine.cc File Reference

```
#include <inference_engine.h>
#include <print_seq.h>
```

Include dependency graph for inference_engine.cc:



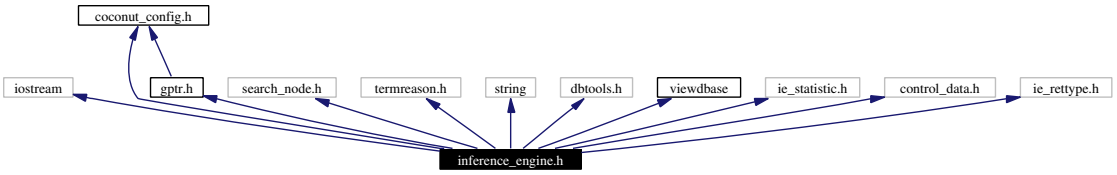

### 7.50.1 Detailed Description

Definition in file inference_engine.cc.

## 7.51 inference_engine.h File Reference

```
#include <iostream>
#include <coconut_config.h>
#include <search_node.h>
#include <termreason.h>
#include <gptr.h>
#include <string>
#include <dbtools.h>
#include <viewdbase>
#include <ie_statistic.h>
#include <control_data.h>
#include <ie_rettype.h>
```

Include dependency graph for inference_engine.h:




### Compounds

- class inference_engine

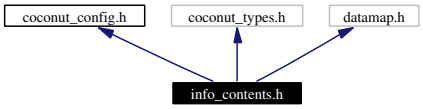
### 7.51.1 Detailed Description

Definition in file inference_engine.h.

## 7.52 info_contents.h File Reference

```
#include <coconut_config.h>
#include <coconut_types.h>
#include <datamap.h>
```

Include dependency graph for info_contents.h:




### Compounds

- class info_contents

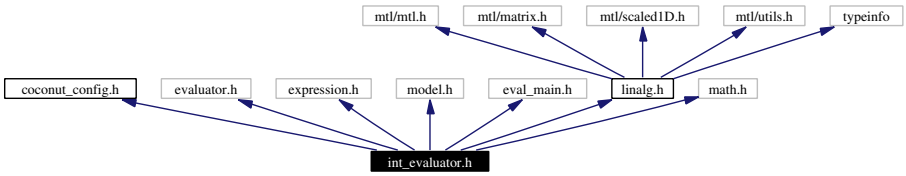
### 7.52.1 Detailed Description

Definition in file info_contents.h.

## 7.53 int_evaluator.h File Reference

```
#include <coconut_config.h>
#include <evaluator.h>
#include <expression.h>
#include <model.h>
#include <eval_main.h>
#include <linalg.h>
#include <math.h>
```

Include dependency graph for int_evaluator.h:




### Compounds

- class interval_eval
- struct interval_eval_type

### Typedefs

- typedef interval(∗ interval_evaluator )(const std::vector< interval > ∗__x, const variable_indicator &__v)

#### 7.53.1 Detailed Description

Definition in file int_evaluator.h.

#### 7.53.2 Typedef Documentation

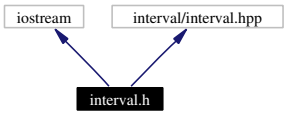
##### 7.53.2.1 typedef interval(∗ interval_evaluator)(const std::vector<interval>∗ __x, const variable_indicator& __v)

Definition at line 40 of file int_evaluator.h.

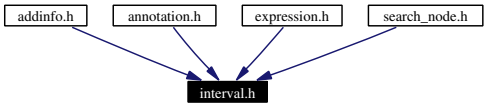
## 7.54 interval.h File Reference

```
#include <iostream>
#include <interval/interval.hpp>
```

Include dependency graph for interval.h:




This graph shows which files directly or indirectly include this file:




### Compounds

- class interval
- struct interval_st

### Defines

- #define FILIB_USE_MACROS 0
- #define FILIB_EXTENDED 1
- #define INFINITY filib::fp_traits_base<double>::infinity()
- #define L_PI filib::fp_traits_base<double>::l_pi()
- #define U_PI filib::fp_traits_base<double>::u_pi()
- #define sgn(x) ((x)>0?1.:(x)<0?-1.:0.)

- #define is_integer(x) (rint((x))==(x)&&rint((x))<MAXINT&&rint((x))>-MAXINT)
- #define get_integer(x) ((int)rint((x)))

**Functions**

- interval ipow (const interval &__i, int __n)
- interval gauss (const interval &__i)
- interval atan2 (const interval &__i, const interval &__j)
- double safeguarded_mid (const interval &__i)
- double absmin (const interval &__i)
- template<class _C> bool operator== (const interval &__i, const _C &__d)
- template<class _C> bool operator!= (const interval &__i, const _C &__d)
- double gainfactor (const interval &_old, const interval &_new)

### 7.54.1 Define Documentation

#### 7.54.1.1 #define FILIB_EXTENDED 1

Definition at line 5 of file interval.h.

#### 7.54.1.2 #define FILIB_USE_MACROS 0

Definition at line 4 of file interval.h.

#### 7.54.1.3 #define get_integer(x) ((int)rint((x)))

Definition at line 33 of file interval.h.

#### 7.54.1.4 #define INFINITY filib::fp_traits_base<double>::infinity()

Definition at line 17 of file interval.h.

#### 7.54.1.5 #define is_integer(x) (rint((x))==(x)&&rint((x))<MAXINT&&rint((x))>-MAXINT)

Definition at line 32 of file interval.h.

#### 7.54.1.6 #define L_PI filib::fp_traits_base<double>::l_pi()

Definition at line 22 of file interval.h.

#### 7.54.1.7 #define sgn(x) ((x)>0?1.:(x)<0?-1.:0.)

Definition at line 31 of file interval.h.

#### 7.54.1.8 #define U_PI filib::fp_traits_base<double>::u_pi()

Definition at line 27 of file interval.h.

### 7.54.2 Function Documentation

#### 7.54.2.1 double absmin (const interval & __i) [inline]

Definition at line 401 of file interval.h.

#### 7.54.2.2 interval atan2 (const interval & __i, const interval & __j) [inline]

Definition at line 375 of file interval.h.

#### 7.54.2.3 double gainfactor (const interval & _old, const interval & _new) [inline]

Definition at line 423 of file interval.h.

#### 7.54.2.4 interval gauss (const interval & __i) [inline]

Definition at line 370 of file interval.h.

#### 7.54.2.5 interval ipow (const interval & __i, int __n) [inline]

Definition at line 365 of file interval.h.

#### 7.54.2.6 template<class _C> bool operator!= (const interval & __i, const _C & __d) [inline]

Definition at line 418 of file interval.h.

#### 7.54.2.7 template<class _C> bool operator== (const interval & __i, const _C & __d) [inline]

Definition at line 412 of file interval.h.

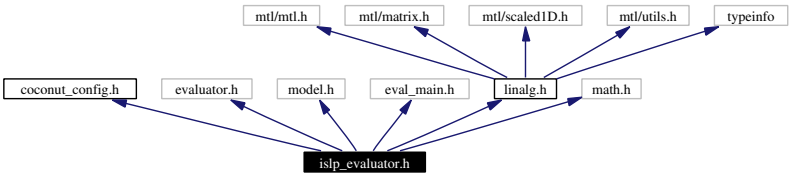
#### 7.54.2.8 double safeguarded_mid (const interval & __i) [inline]

Definition at line 381 of file interval.h.

## 7.55 islp_evaluator.h File Reference

```
#include <coconut_config.h>
#include <evaluator.h>
#include <model.h>
#include <eval_main.h>
#include <linalg.h>
#include <math.h>
```

Include dependency graph for islp_evaluator.h:




### Compounds

- class func_islp_eval
- struct func_islp_eval_type
- struct func_islp_return_type
- class islp_eval
- struct islp_eval_type
- class prep_islp_eval

### Typedefs

- typedef bool(∗ prep_islp_evaluator )()
- typedef func_islp_return_type(∗ func_islp_evaluator )(const std::vector< interval > ∗__x, const variable_indicator &__v, std::vector< interval > &__islp_data)
- typedef std::vector< interval > &(∗ islp_evaluator )(const std::vector< interval > &__d_dat, const variable_indicator &__v)

#### 7.55.1 Detailed Description

Definition in file islp_evaluator.h.

#### 7.55.2 Typedef Documentation

##### 7.55.2.1 typedef func_islp_return_type(∗ func_islp_evaluator)( const std::vector<interval>∗ __x, const variable_indicator& __v, std::vector<interval>& __islp_data)

Definition at line 47 of file islp_evaluator.h.

##### 7.55.2.2 typedef std::vector<interval>&(∗ islp_evaluator)( const std::vector<interval>& __d_dat, const variable_indicator& __v)

Definition at line 50 of file islp_evaluator.h.

##### 7.55.2.3 typedef bool(∗ prep_islp_evaluator)()

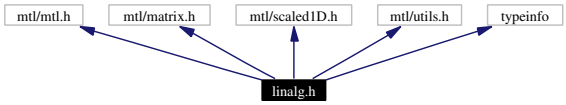
Definition at line 46 of file islp_evaluator.h.

## 7.56 linalg.h File Reference

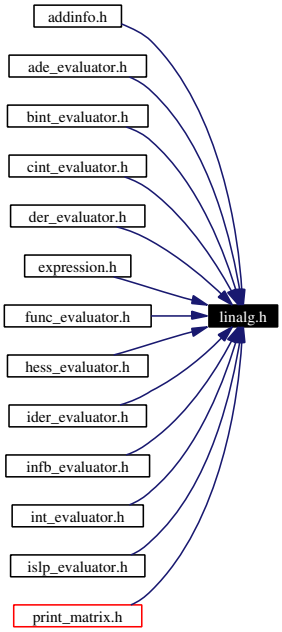
```
#include <mtl/mtl.h>
#include <mtl/matrix.h>
```

```
#include <mtl/scaled1D.h>
#include <mtl/utils.h>
#include <typeinfo>
```

Include dependency graph for linalg.h:




This graph shows which files directly or indirectly include this file:




## Compounds

- class __linalg_cvec
- class __linalg_vec
- class c_matrix
- class matrix
- class sparse_vector

## Defines

- #define __LINALG_RANGE_CHECK 0
- #define linalg_scale(X, Y) mtl::scaled(X, Y)
- #define linalg_trans(A) mtl::trans((A))

### Functions

- template<class _Tp> bool operator== (const sparse_vector< _Tp > &a, const sparse_vector< _Tp > &b)
- template<class _S> _S linalg_dot (const std::vector< _S > &a, const std::vector< _S > &b, _S s)
- template<class _Va, class _S> _S linalg_dot (const _Va &a, const std::vector< _S > &b, _S s)
- template<class _Vb, class _S> _S linalg_dot (const std::vector< _S > &a, const _Vb &b, _S s)
- template<class _Va, class _Vb, class _S> _S linalg_dot (const _Va &a, const _Vb &b, _S s)
- template<class _S> void linalg_add (const std::vector< _S > &a, const std::vector< _S > &b, std::vector< _S > &c)
- template<class _Va, class _S> void linalg_add (const _Va &a, const std::vector< _S > &b, std::vector< _S > &c)
- template<class _Vb, class _S> void linalg_add (const std::vector< _S > &a, const _Vb &b, std::vector< _S > &c)
- template<class _S, class _Vc> void linalg_add (const std::vector< _S > &a, const std::vector< _S > &b, _Vc &c)
- template<class _Va, class _Vb, class _S> void linalg_add (const _Va &a, const _Vb &b, std::vector< _S > &c)
- template<class _Va, class _S, class _Vc> void linalg_add (const _Va &a, const std::vector< _S > &b, _Vc &c)
- template<class _S, class _Vb, class _Vc> void linalg_add (const std::vector< _S > &a, const _Vb &b, _Vc &c)
- template<class _Va, class _Vb, class _Vc> void linalg_add (const _Va &a, const _Vb &b, _Vc &c)
- template<class _Matrix, class _S> void linalg_matvec (const _Matrix &A, const std::vector< _S > &a, std::vector< _S > &c)
- template<class _Matrix, class _S, class _Va> void linalg_matvec (const _Matrix &A, const _Va &a, std::vector< _S > &c)
- template<class _Matrix, class _S, class _Vc> void linalg_matvec (const _Matrix &A, const std::vector< _S > &a, _Vc &c)
- template<class _Matrix, class _Va, class _Vc> void linalg_matvec (const _Matrix &A, const _Va &a, _Vc &c)
- template<class _Matrix, class _S> void linalg_matvecv (const _Matrix &A, const std::vector< _S > &a, const std::vector< _S > &b, std::vector< _S > &c)
- template<class _Matrix, class _S, class _Va> void linalg_matvecv (const _Matrix &A, const _Va &a, const std::vector< _S > &b, std::vector< _S > &c)
- template<class _Matrix, class _S, class _Vb> void linalg_matvecv (const _Matrix &A, const std::vector< _S > &a, const _Vb &b, std::vector< _S > &c)
- template<class _Matrix, class _S, class _Vc> void linalg_matvecv (const _Matrix &A, const std::vector< _S > &a, const std::vector< _S > &b, _Vc &c)
- template<class _Matrix, class _Va, class _Vb, class _S> void linalg_matvecv (const _Matrix &A, _Va &a, const _Vb &b, const std::vector< _S > &c)
- template<class _Matrix, class _Va, class _S, class _Vc> void linalg_matvecv (const _Matrix &A, const _Va &a, const std::vector< _S > &b, _Vc &c)
- template<class _Matrix, class _S, class _Vb, class _Vc> void linalg_matvecv (const _Matrix &A, const std::vector< _S > &a, const _Vb &b, _Vc &c)
- template<class _Matrix, class _Va, class _Vb, class _Vc> void linalg_matvecv (const _Matrix &A, const _Va &a, const _Vb &b, _Vc &c)
- template<class _Ve> std::vector< _Ve > & linalg_ssum (std::vector< _Ve > &a, _Ve b, const std::vector< _Ve > &c)
- template<class _Ve> std::vector< _Ve > & linalg_smult (std::vector< _Ve > &a, _Ve b)

### 7.56.1 Define Documentation

#### 7.56.1.1 #define __LINALG_RANGE_CHECK 0

Definition at line 4 of file linalg.h.

#### 7.56.1.2 #define linalg_scale(X, Y) mtl::scaled(X, Y)

Definition at line 104 of file linalg.h.

#### 7.56.1.3 #define linalg_trans(A) mtl::trans((A))

Definition at line 352 of file linalg.h.

### 7.56.2 Function Documentation

#### 7.56.2.1 template<class _Va, class _Vb, class _Vc> void linalg_add (const _Va & *a*, const _Vb & *b*, _Vc & *c*) [inline]

Definition at line 262 of file linalg.h.

#### 7.56.2.2 template<class _S, class _Vb, class _Vc> void linalg_add (const std::vector< _S > & *a*, const _Vb & *b*, _Vc & *c*) [inline]

Definition at line 256 of file linalg.h.

#### 7.56.2.3 template<class _Va, class _S, class _Vc> void linalg_add (const _Va & *a*, const std::vector< _S > & *b*, _Vc & *c*) [inline]

Definition at line 250 of file linalg.h.

#### 7.56.2.4 template<class _Va, class _Vb, class _S> void linalg_add (const _Va & *a*, const _Vb & *b*, std::vector< _S > & *c*) [inline]

Definition at line 244 of file linalg.h.

#### 7.56.2.5 template<class _S, class _Vc> void linalg_add (const std::vector< _S > & *a*, const std::vector< _S > & *b*, _Vc & *c*) [inline]

Definition at line 238 of file linalg.h.

#### 7.56.2.6 template<class _Vb, class _S> void linalg_add (const std::vector< _S > & *a*, const _Vb & *b*, std::vector< _S > & *c*) [inline]

Definition at line 232 of file linalg.h.

#### 7.56.2.7 template<class _Va, class _S> void linalg_add (const _Va & *a*, const std::vector< _S > & *b*, std::vector< _S > & *c*) [inline]

Definition at line 226 of file linalg.h.

#### 7.56.2.8 template<class _S> void linalg_add (const std::vector< _S > & *a*, const std::vector< _S > & *b*, std::vector< _S > & *c*) [inline]

Definition at line 220 of file linalg.h.

#### 7.56.2.9 template<class _Va, class _Vb, class _S> _S linalg_dot (const _Va & *a*, const _Vb & *b*, _S *s*) [inline]

Definition at line 214 of file linalg.h.

#### 7.56.2.10 template<class _Vb, class _S> _S linalg_dot (const std::vector< _S > & *a*, const _Vb & *b*, _S *s*) [inline]

Definition at line 208 of file linalg.h.

#### 7.56.2.11 template<class _Va, class _S> _S linalg_dot (const _Va & *a*, const std::vector< _S > & *b*, _S *s*) [inline]

Definition at line 201 of file linalg.h.

#### 7.56.2.12 template<class _S> _S linalg_dot (const std::vector< _S > & *a*, const std::vector< _S > & *b*, _S *s*) [inline]

Definition at line 195 of file linalg.h.

#### 7.56.2.13 template<class _Matrix, class _Va, class _Vc> void linalg_matvec (const _Matrix & *A*, const _Va & *a*, _Vc & *c*) [inline]

Definition at line 290 of file linalg.h.

#### 7.56.2.14 template<class _Matrix, class _S, class _Vc> void linalg_matvec (const _Matrix & *A*, const std::vector< _S > & *a*, _Vc & *c*) [inline]

Definition at line 283 of file linalg.h.

#### 7.56.2.15 template<class _Matrix, class _S, class _Va> void linalg_matvec (const _Matrix & *A*, const _Va & *a*, std::vector< _S > & *c*) [inline]

Definition at line 276 of file linalg.h.

#### 7.56.2.16 template<class _Matrix, class _S> void linalg_matvec (const _Matrix & *A*, const std::vector< _S > & *a*, std::vector< _S > & *c*) [inline]

Definition at line 269 of file linalg.h.

#### 7.56.2.17 template<class _Matrix, class _Va, class _Vb, class _Vc> void linalg_matvecv (const _Matrix & *A*, const _Va & *a*, const _Vb & *b*, _Vc & *c*) [inline]

Definition at line 346 of file linalg.h.

**7.56.2.18 template<class _Matrix, class _S, class _Vb, class _Vc> void linalg_matvecv (const _Matrix & *A*, const std::vector< _S > & *a*, const _Vb & *b*, _Vc & *c*)** `[inline]`

Definition at line 339 of file linalg.h.

**7.56.2.19 template<class _Matrix, class _Va, class _S, class _Vc> void linalg_matvecv (const _Matrix & *A*, const _Va & *a*, const std::vector< _S > & *b*, _Vc & *c*)** `[inline]`

Definition at line 332 of file linalg.h.

**7.56.2.20 template<class _Matrix, class _Va, class _Vb, class _S> void linalg_matvecv (const _Matrix & *A*, _Va & *a*, const _Vb & *b*, const std::vector< _S > & *c*)** `[inline]`

Definition at line 325 of file linalg.h.

**7.56.2.21 template<class _Matrix, class _S, class _Vc> void linalg_matvecv (const _Matrix & *A*, const std::vector< _S > & *a*, const std::vector< _S > & *b*, _Vc & *c*)** `[inline]`

Definition at line 318 of file linalg.h.

**7.56.2.22 template<class _Matrix, class _S, class _Vb> void linalg_matvecv (const _Matrix & *A*, const std::vector< _S > & *a*, const _Vb & *b*, std::vector< _S > & *c*)** `[inline]`

Definition at line 311 of file linalg.h.

**7.56.2.23 template<class _Matrix, class _S, class _Va> void linalg_matvecv (const _Matrix & *A*, const _Va & *a*, const std::vector< _S > & *b*, std::vector< _S > & *c*)** `[inline]`

Definition at line 304 of file linalg.h.

**7.56.2.24 template<class _Matrix, class _S> void linalg_matvecv (const _Matrix & *A*, const std::vector< _S > & *a*, const std::vector< _S > & *b*, std::vector< _S > & *c*)** `[inline]`

Definition at line 296 of file linalg.h.

**7.56.2.25 template<class _Ve> std::vector<_Ve>& linalg_smult (std::vector< _Ve > & *a*, _Ve *b*)** `[inline]`

Definition at line 371 of file linalg.h.

**7.56.2.26 template<class _Ve> std::vector<_Ve>& linalg_ssum (std::vector< _Ve > & *a*, _Ve *b*, const std::vector< _Ve > & *c*)** `[inline]`

Definition at line 356 of file linalg.h.

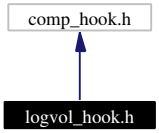
**7.56.2.27 template<class _Tp> bool operator== (const sparse_vector< _Tp > & *a*, const sparse_vector< _Tp > & *b*)**

Definition at line 46 of file linalg.h.

## 7.57 logvol_hook.h File Reference

`#include <comp_hook.h>`

Include dependency graph for logvol_hook.h:




### Compounds

- class logvol_comp_hook

#### 7.57.1 Detailed Description

Definition in file logvol_hook.h.

## 7.58 model-inline.h File Reference

### Compounds

- struct __docompare_nodes
- struct __docompare_variables
- class detect_0chain_visitor
- struct detect_0chain_visitor_st
- class lincoeff_visitor
- struct lincoeff_visitor_ret
- struct lincoeff_visitor_st
- class simplify_visitor_0
- class sort_constraints

### Defines

- #define MODEL_INLINE_DEBUG 0
- #define MODEL_INLINE_DEBUG_SIMPLIFY 0
- #define SIMPLIFY_0_IS_CONST 1
- #define SIMPLIFY_0_CONST_IS_INTEGER (1<<24)
- #define SIMPLIFY_0_IS_VAR (1<<1)
- #define SIMPLIFY_0_IS_SUM (1<<2)
- #define SIMPLIFY_0_SUM_IS_SIMPLE (1<<24)
- #define SIMPLIFY_0_IS_PROD (1<<3)
- #define SIMPLIFY_0_PROD_IS_SIMPLE (1<<24)
- #define SIMPLIFY_0_IS_MULTIPLICATIVE (1<<4)
- #define SIMPLIFY_0_IS_CORRECTEDMULT (1<<5)
- #define SIMPLIFY_0_IS_GHOST (1<<31)

### 7.58.1 Detailed Description

Definition in file model-inline.h.

### 7.58.2 Define Documentation

#### 7.58.2.1 #define MODEL_INLINE_DEBUG 0

Definition at line 30 of file model-inline.h.

#### 7.58.2.2 #define MODEL_INLINE_DEBUG_SIMPLIFY 0

Definition at line 31 of file model-inline.h.

#### 7.58.2.3 #define SIMPLIFY_0_CONST_IS_INTEGER (1<<24)

Definition at line 549 of file model-inline.h.

#### 7.58.2.4 #define SIMPLIFY_0_IS_CONST 1

Definition at line 548 of file model-inline.h.

#### 7.58.2.5 #define SIMPLIFY_0_IS_CORRECTEDMULT (1<<5)

Definition at line 560 of file model-inline.h.

#### 7.58.2.6 #define SIMPLIFY_0_IS_GHOST (1<<31)

Definition at line 562 of file model-inline.h.

#### 7.58.2.7 #define SIMPLIFY_0_IS_MULTIPLICATIVE (1<<4)

Definition at line 559 of file model-inline.h.

#### 7.58.2.8 #define SIMPLIFY_0_IS_PROD (1<<3)

Definition at line 556 of file model-inline.h.

#### 7.58.2.9 #define SIMPLIFY_0_IS_SUM (1<<2)

Definition at line 553 of file model-inline.h.

#### 7.58.2.10 #define SIMPLIFY_0_IS_VAR (1<<1)

Definition at line 551 of file model-inline.h.

#### 7.58.2.11 #define SIMPLIFY_0_PROD_IS_SIMPLE (1<<24)

Definition at line 557 of file model-inline.h.

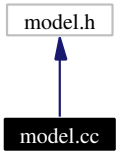
#### 7.58.2.12 #define SIMPLIFY_0_SUM_IS_SIMPLE (1<<24)

Definition at line 554 of file model-inline.h.

## 7.59 model.cc File Reference

`#include <model.h>`

Include dependency graph for model.cc:




### Defines

- #define throw_read(MSG)
- #define MT_DBL 1
- #define MT_INTV 2
- #define MT_INT 3
- #define MT_STORED 8

### Variables

- const char * expr_names [ ]

### 7.59.1 Detailed Description

Definition in file model.cc.

### 7.59.2 Define Documentation

#### 7.59.2.1 #define MT_DBL 1

Definition at line 351 of file model.cc.

#### 7.59.2.2 #define MT_INT 3

Definition at line 353 of file model.cc.

#### 7.59.2.3 #define MT_INTV 2

Definition at line 352 of file model.cc.

#### 7.59.2.4 #define MT_STORED 8

Definition at line 354 of file model.cc.

#### 7.59.2.5 #define throw_read(MSG)

**Value:**

```
do { std::cerr << "Line " << ln << ": " << MSG;\
                              throw "Malformed Input!"; } while(0)
```

Definition at line 349 of file model.cc.

### 7.59.3 Variable Documentation

#### 7.59.3.1 const char* expr_names[ ]

Definition at line 29 of file model.cc.

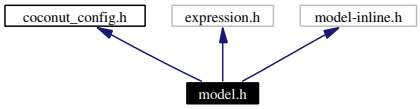
## 7.60 model.h File Reference

`#include <coconut_config.h>`

`#include <expression.h>`

`#include <model-inline.h>`

Include dependency graph for model.h:




### Compounds

- class model
- class model_gid
- class model_iddata

### Typedefs

- typedef model::walker expression_walker
- typedef model::const_walker expression_const_walker

### 7.60.1 Detailed Description

Definition in file model.h.

### 7.60.2 Typedef Documentation

#### 7.60.2.1 typedef model::const_walker expression_const_walker

Definition at line 461 of file model.h.

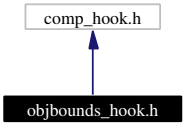
#### 7.60.2.2 typedef model::walker expression_walker

Definition at line 460 of file model.h.

## 7.61 objbounds_hook.h File Reference

`#include <comp_hook.h>`

Include dependency graph for objbounds_hook.h:




### Compounds

- class objbounds_comp_hook

### 7.61.1 Detailed Description

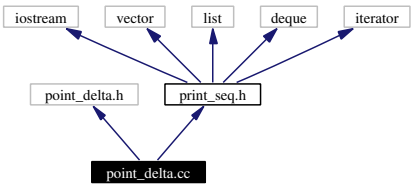
Definition in file objbounds_hook.h.

## 7.62 point_delta.cc File Reference

`#include <point_delta.h>`

`#include <print_seq.h>`

Include dependency graph for point_delta.cc:

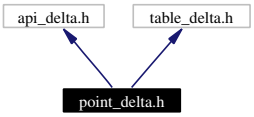



## 7.63 point_delta.h File Reference

`#include <api_delta.h>`

`#include <table_delta.h>`

Include dependency graph for point_delta.h:




### Compounds

- class point_delta

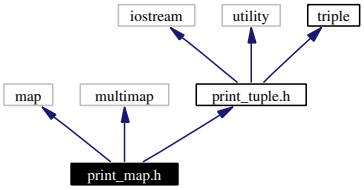
#### 7.63.1 Detailed Description

Definition in file point_delta.h.

## 7.64 print_map.h File Reference

```
#include <map>
#include <multimap>
#include <print_tuple.h>
```

Include dependency graph for print_map.h:



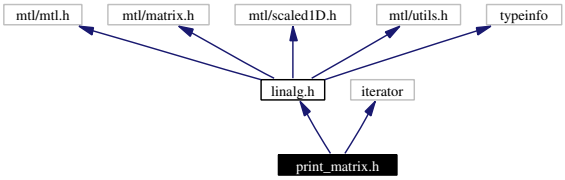

#### 7.64.1 Detailed Description

Definition in file print_map.h.

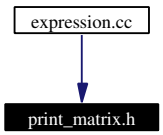
## 7.65 print_matrix.h File Reference

```
#include <linalg.h>
#include <iterator>
```

Include dependency graph for print_matrix.h:



This graph shows which files directly or indirectly include this file:




### Namespaces

- namespace std

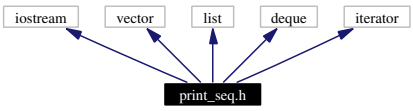
#### 7.65.1 Detailed Description

Definition in file print_matrix.h.

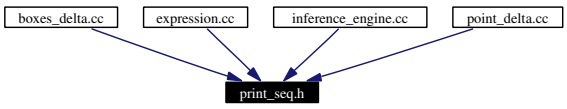
## 7.66 print_seq.h File Reference

```
#include <iostream>
#include <vector>
#include <list>
#include <deque>
#include <iterator>
```

Include dependency graph for print_seq.h:




This graph shows which files directly or indirectly include this file:




### Namespaces

- namespace std

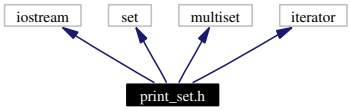
#### 7.66.1 Detailed Description

Definition in file print_seq.h.

## 7.67 print_set.h File Reference

```
#include <iostream>
#include <set>
#include <multiset>
#include <iterator>
```

Include dependency graph for print_set.h:



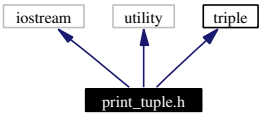

### 7.67.1 Detailed Description

Definition in file print_set.h.

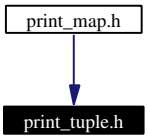
## 7.68 print_tuple.h File Reference

```
#include <iostream>
#include <utility>
#include <triple>
```

Include dependency graph for print_tuple.h:




This graph shows which files directly or indirectly include this file:



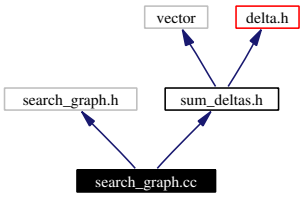

### 7.68.1 Detailed Description

Definition in file print_tuple.h.

## 7.69 search_graph.cc File Reference

```
#include <search_graph.h>
#include "sum_deltas.h"
```

Include dependency graph for search_graph.cc:



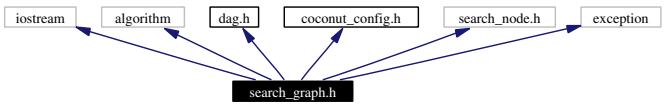

### 7.69.1 Detailed Description

Definition in file search_graph.cc.

## 7.70 search_graph.h File Reference

```
#include <iostream>
#include <algorithm>
#include <dag.h>
#include <coconut_config.h>
#include <search_node.h>
#include <exception>
```

Include dependency graph for search_graph.h:




### Compounds

- class search_graph
- class search_graph_exception

### Typedefs

- typedef **search_graph::const_walker** search_inspector
- typedef **search_graph::walker** search_focus

### 7.70.1 Detailed Description

Definition in file search_graph.h.

### 7.70.2 Typedef Documentation

#### 7.70.2.1 typedef search_graph::walker search_focus

Definition at line 176 of file search_graph.h.

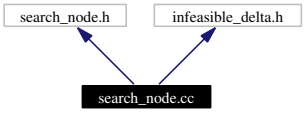
#### 7.70.2.2 typedef search_graph::const_walker search_inspector

Definition at line 175 of file search_graph.h.

## 7.71 search_node.cc File Reference

```
#include <search_node.h>
#include <infeasible_delta.h>
```

Include dependency graph for search_node.cc:




### Defines

- #define MAX_VOL_COMP 1.e12
- #define MIN_VOL_WIDTH DBL_MIN

### Functions

- work_node operator- (const work_node &_w, const delta_id &_d)
- work_node & operator-= (work_node &_w, const delta_id &_d)

### 7.71.1 Detailed Description

Definition in file search_node.cc.

### 7.71.2 Define Documentation

#### 7.71.2.1 #define MAX_VOL_COMP 1.e12

Definition at line 172 of file search_node.cc.

#### 7.71.2.2 #define MIN_VOL_WIDTH DBL_MIN

Definition at line 173 of file search_node.cc.

### 7.71.3 Function Documentation

#### 7.71.3.1 work_node operator- (const work_node & _w, const delta_id & _d)

Definition at line 30 of file search_node.cc.

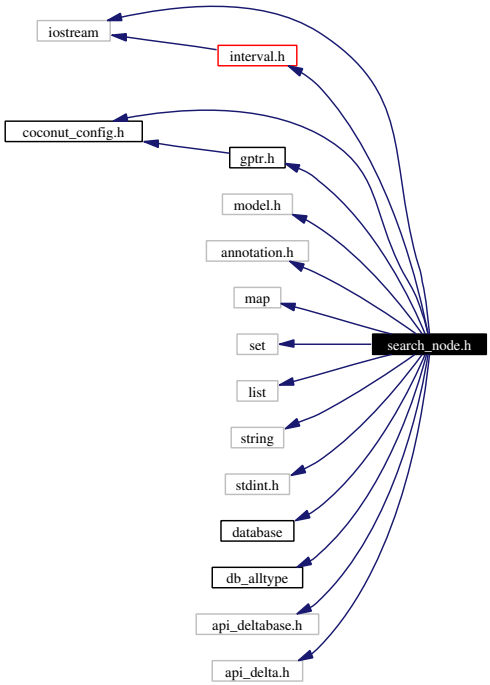
#### 7.71.3.2 work_node& operator-= (work_node & _w, const delta_id & _d)

Definition at line 60 of file search_node.cc.

## 7.72 search_node.h File Reference

```
#include <iostream>
#include <interval.h>
#include <coconut_config.h>
#include <model.h>
#include <annotation.h>
#include <map>
#include <set>
#include <list>
#include <string>
#include <gptr.h>
#include <stdint.h>
#include <database>
#include <db_alltype>
#include <api_deltabase.h>
#include <api_delta.h>
```

Include dependency graph for search_node.h:




## Compounds

- class constraint_iterator_base
- class delta_node
- class full_node
- class search_node
- class work_node

## Typedefs

- typedef uint32_t search_node_id

## Enumerations

- enum search_node_relation { snr_root, snr_reduction, snr_relaxation, snr_split, snr_glue, snr_worknode, snr_virtual }

## Functions

- work_node operator+ (const work_node &_w, const delta_id &_i)
- work_node & operator+= (work_node &_w, const delta_id &_i)
- template<template< class _Tp, class _A > class _Ctr, class _Al> work_node operator+ (const work_node &_w, const _Ctr< delta_id, _Al > &_d)
- template<template< class _Tp, class _A > class _Ctr, class _Al> work_node operator- (const work_node &_w, const _Ctr< delta_id, _Al > &_d)
- template<template< class _Tp, class _A > class _Ctr, class _Al> work_node & operator+= (work_node &_w, const _Ctr< delta_id, _Al > &_d)

- template<template< class _Tp, class _A > class _Ctr, class _Al> work_node & operator-= (work_node &_w, const _Ctr< delta_id, _Al > &_d)

### 7.72.1 Detailed Description

Definition in file search_node.h.

### 7.72.2 Typedef Documentation

#### 7.72.2.1 typedef uint32_t search_node_id

Definition at line 65 of file search_node.h.

### 7.72.3 Enumeration Type Documentation

#### 7.72.3.1 enum search_node_relation

**Enumeration values:**

**snr_root**

**snr_reduction**

**snr_relaxation**

**snr_split**

**snr_glue**

**snr_worknode**

**snr_virtual**

Definition at line 55 of file search_node.h.

### 7.72.4 Function Documentation

#### 7.72.4.1 template<template< class _Tp, class _A > class _Ctr, class _Al> work_node operator+ (const work_node & *_w*, const _Ctr< delta_id, _Al > & *_d*) [inline]

Definition at line 590 of file search_node.h.

#### 7.72.4.2 work_node operator+ (const work_node & *_w*, const delta_id & *_i*) [inline]

Definition at line 570 of file search_node.h.

#### 7.72.4.3 template<template< class _Tp, class _A > class _Ctr, class _Al> work_node& operator+= (work_node & *_w*, const _Ctr< delta_id, _Al > & *_d*) [inline]

Definition at line 616 of file search_node.h.

#### 7.72.4.4 work_node& operator+= (work_node & *_w*, const delta_id & *_i*) [inline]

Definition at line 580 of file search_node.h.

#### 7.72.4.5 template<template< class _Tp, class _A > class _Ctr, class _Al> work_node operator-(const work_node & _w, const _Ctr< delta_id, _Al > & _d) [inline]

Definition at line 602 of file search_node.h.

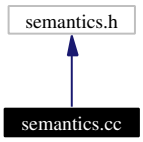
#### 7.72.4.6 template<template< class _Tp, class _A > class _Ctr, class _Al> work_node& operator-=(work_node & _w, const _Ctr< delta_id, _Al > & _d) [inline]

Definition at line 627 of file search_node.h.

## 7.73 semantics.cc File Reference

```
#include <semantics.h>
```

Include dependency graph for semantics.cc:




### Functions

- std::ostream & operator<< (std::ostream &o, const semantics &__s)

### 7.73.1 Detailed Description

Definition in file semantics.cc.

### 7.73.2 Function Documentation

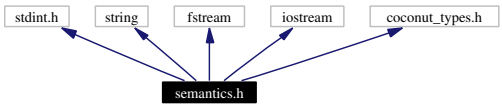
#### 7.73.2.1 std::ostream& operator<< (std::ostream & *o*, const semantics & *__s*)

Definition at line 224 of file semantics.cc.

## 7.74 semantics.h File Reference

```
#include <stdint.h>
#include <string>
#include <fstream>
#include <iostream>
#include <coconut_types.h>
```

Include dependency graph for semantics.h:




### Compounds

- class convex_e
- class semantics

### Enumerations

- enum convex_info { c_convex = 1, c_linear = 0, c_concave = -1, c_maybe = 2 }
- enum type_annotation { v_exists = 0, v_forall = 1, v_free = 2, v_stochastic = 3 }
- enum activity_descr { a_redundant = 1, a_active_lo = 2, a_active_lo_red = a_active_lo|a_redundant, a_active_hi = 4, a_active_hi_red = a_active_hi|a_redundant, a_active = a_active_lo|a_active_hi, a_active_red = a_active|a_redundant }

### Functions

- bool operator== (const convex_e &__c, const convex_e &__d)
- bool operator== (const convex_e &__c, const convex_info &__d)
- bool operator== (const convex_info &__c, const convex_e &__d)
- bool operator!= (const convex_e &__c, const convex_e &__d)
- bool operator!= (const convex_e &__c, const convex_info &__d)
- bool operator!= (const convex_info &__c, const convex_e &__d)
- std::ostream & operator<< (std::ostream &o, const convex_e &__s)
- std::ostream & operator<< (std::ostream &o, const semantics &__s)

#### 7.74.1 Detailed Description

Definition in file semantics.h.

#### 7.74.2 Enumeration Type Documentation

##### 7.74.2.1 enum activity_descr

This enum describes the activity state known about a constraint and whether it is redundant or not. The meaning of the enum entries is:

- `a_redundant`: The constraint is known to be redundant. There can be two reasons for that. It can be inactive on both sides, or it was constructed, knowing that it would be redundant (e.g. cuts).
- `a_active_lo`: The `work_node` may contain points, for which this constraint is active at the lower bound but no points, for which the constraint is active at the upper bound.
- `a_active_hi`: The `work_node` may contain points, for which this constraint is active at the upper bound but no points, for which the constraint is active at the lower bound.
- `a_active`: The `work_node` may contain points, for which this constraint is active at the upper bound and points, for which the constraint is active at the lower bound.

- ..._red: A combination with _red means that although the inactivity of the constraint could not be proved, it is still known to be redundant.

**Enumeration values:**

**a_redundant**

**a_active_lo**

**a_active_lo_red**

**a_active_hi**

**a_active_hi_red**

**a_active**

**a_active_red**

Definition at line 66 of file semantics.h.

#### 7.74.2.2 enum convex_info

**Enumeration values:**

**c_convex**

**c_linear**

**c_concave**

**c_maybe**

Definition at line 36 of file semantics.h.

#### 7.74.2.3 enum type_annotation

**Enumeration values:**

**v_exists**

**v_forall**

**v_free**

**v_stochastic**

Definition at line 38 of file semantics.h.

### 7.74.3 Function Documentation

#### 7.74.3.1 bool operator!= (const convex_info & __c, const convex_e & __d) [inline]

Definition at line 177 of file semantics.h.

#### 7.74.3.2 bool operator!= (const convex_e & __c, const convex_info & __d) [inline]

Definition at line 172 of file semantics.h.

#### 7.74.3.3 bool operator!= (const convex_e & __c, const convex_e & __d) [inline]

Definition at line 167 of file semantics.h.

#### 7.74.3.4 std::ostream& operator<< (std::ostream & *o*, const semantics & *__s*)

Definition at line 224 of file semantics.cc.

#### 7.74.3.5 std::ostream& operator<< (std::ostream & *o*, const convex_e & *__s*) [inline]

Definition at line 182 of file semantics.h.

#### 7.74.3.6 bool operator== (const convex_info & *__c*, const convex_e & *__d*) [inline]

Definition at line 162 of file semantics.h.

#### 7.74.3.7 bool operator== (const convex_e & *__c*, const convex_info & *__d*) [inline]

Definition at line 157 of file semantics.h.

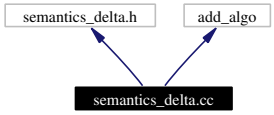
#### 7.74.3.8 bool operator== (const convex_e & *__c*, const convex_e & *__d*) [inline]

Definition at line 152 of file semantics.h.

## 7.75 semantics_delta.cc File Reference

```
#include <semantics_delta.h>
#include <add_algo>
```

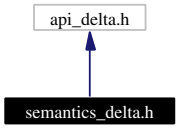
Include dependency graph for semantics_delta.cc:




## 7.76 semantics_delta.h File Reference

```
#include <api_delta.h>
```

Include dependency graph for semantics_delta.h:




### Compounds

- class semantics_delta
- class semantics_undelta

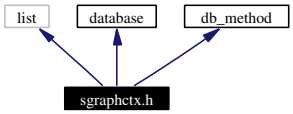
### 7.76.1 Detailed Description

Definition in file semantics_delta.h.

## 7.77 sgraphctx.h File Reference

```
#include <list>
#include <database>
#include <db_method>
```

Include dependency graph for sgraphctx.h:




### Compounds

- class search_graph_context

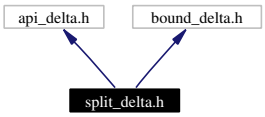
### 7.77.1 Detailed Description

Definition in file sgraphctx.h.

## 7.78 split_delta.h File Reference

```
#include <api_delta.h>
#include <bound_delta.h>
```

Include dependency graph for split_delta.h:




### Compounds

- class split_delta
- class split_undelta

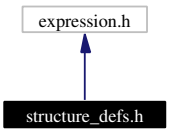
### 7.78.1 Detailed Description

Definition in file split_delta.h.

## 7.79 structure_defs.h File Reference

`#include <expression.h>`

Include dependency graph for structure_defs.h:




### Defines

- #define CTR_NODES 0
- #define CTR_INTEGER 1
- #define CTR_KJNODES 2
- #define CTR_FREE 3
- #define CTR_EXISTS 4
- #define CTR_FORALL 5
- #define CTR_STOCHASTIC 6
- #define NUM_CTR 7
- #define CTR_VARS 0
- #define CTR_VINTEGER 1
- #define CTR_VFREE 2
- #define CTR_VEXISTS 3
- #define CTR_VFORALL 4
- #define CTR_VSTOCHASTIC 5
- #define CTR_VKJ 6
- #define CTR_V2BOUNDED 7
- #define CTR_V1BOUNDED 8
- #define CTR_VUBOUNDED 9
- #define NUM_VCTR 10
- #define CTR_OBJ 0
- #define CTR_OINTEGER 1
- #define CTR_OFREE 2
- #define CTR_OEXISTS 3
- #define CTR_OFORALL 4
- #define CTR_OSTOCHASTIC 5
- #define CTR_O2BOUNDED 6
- #define CTR_O1BOUNDED 7
- #define CTR_OUBOUNDED 8
- #define CTR_ODIM1 9
- #define CTR_ODIM2 10
- #define CTR_OMTDIM2 11
- #define CTR_ODEG1 12
- #define CTR_ODEG2 13
- #define CTR_ODEG3 14
- #define CTR_OMTDEG3P 15
- #define CTR_ONLN 16

- #define CTR_OCONVEX 17
- #define CTR_OCONCAVE 18
- #define CTR_MIN 19
- #define NUM_OCTR 20
- #define CTR_CONSTR 0
- #define CTR_CEQU 1
- #define CTR_CINEQ 2
- #define CTR_EBOUND 3
- #define CTR_EMTDEG3P 7
- #define CTR_ENLN 8
- #define CTR_EDIM1 9
- #define CTR_EDIM2 10
- #define CTR_EMTDIM2 11
- #define CTR_EINTEGER 12
- #define CTR_IBOUND 13
- #define CTR_IMTDEG3P 17
- #define CTR_INLN 18
- #define CTR_IDIM1 19
- #define CTR_IDIM2 20
- #define CTR_IMTDIM2 21
- #define CTR_IINTEGER 22
- #define CTR_ICONVEX 23
- #define CTR_ICONCAVE 24
- #define CTR_IACTIVE 25
- #define CTR_IINACTIVE 26
- #define CTR_IACTIVE_LO 27
- #define CTR_IINACTIVE_LO 28
- #define CTR_IACTIVE_HI 29
- #define CTR_IINACTIVE_HI 30
- #define CTR_IREDUNDANT 31
- #define CTR_IRESTRICT 32
- #define NUM_CCTR 33
- #define CTR_EDGES 0
- #define CTR_LEAVES 1
- #define CTR_ROOTS 2
- #define NUM_ECTR 3
- #define CTR_EXPRINFO_USERDEFINED (EXPRINFO_NUMOFPREDEF+1)

### 7.79.1 Detailed Description

Definition in file structure_defs.h.

### 7.79.2 Define Documentation

#### 7.79.2.1 #define CTR_CEQU 1

Definition at line 76 of file structure_defs.h.

#### 7.79.2.2 #define CTR_CINEQ 2

Definition at line 77 of file structure_defs.h.

#### 7.79.2.3 #define CTR_CONSTR 0

Definition at line 75 of file structure_defs.h.

#### 7.79.2.4 #define CTR_EBOUND 3

Definition at line 78 of file structure_defs.h.

#### 7.79.2.5 #define CTR_EDGES 0

Definition at line 104 of file structure_defs.h.

#### 7.79.2.6 #define CTR_EDIM1 9

Definition at line 81 of file structure_defs.h.

#### 7.79.2.7 #define CTR_EDIM2 10

Definition at line 82 of file structure_defs.h.

#### 7.79.2.8 #define CTR_EINTEGER 12

Definition at line 84 of file structure_defs.h.

#### 7.79.2.9 #define CTR_EMTDEG3P 7

Definition at line 79 of file structure_defs.h.

#### 7.79.2.10 #define CTR_EMTDIM2 11

Definition at line 83 of file structure_defs.h.

#### 7.79.2.11 #define CTR_ENLN 8

Definition at line 80 of file structure_defs.h.

#### 7.79.2.12 #define CTR_EXISTS 4

Definition at line 36 of file structure_defs.h.

#### 7.79.2.13 #define CTR_EXPRINFO_USERDEFINED (EXPRINFO_NUMOFPREDEF+1)

Definition at line 109 of file structure_defs.h.

#### 7.79.2.14 #define CTR_FORALL 5

Definition at line 37 of file structure_defs.h.

#### 7.79.2.15 #define CTR_FREE 3

Definition at line 35 of file structure_defs.h.

#### 7.79.2.16 #define CTR_IACTIVE 25

Definition at line 94 of file structure_defs.h.

#### 7.79.2.17 #define CTR_IACTIVE_HI 29

Definition at line 98 of file structure_defs.h.

#### 7.79.2.18 #define CTR_IACTIVE_LO 27

Definition at line 96 of file structure_defs.h.

#### 7.79.2.19 #define CTR_IBOUND 13

Definition at line 85 of file structure_defs.h.

#### 7.79.2.20 #define CTR_ICONCAVE 24

Definition at line 93 of file structure_defs.h.

#### 7.79.2.21 #define CTR_ICONVEX 23

Definition at line 92 of file structure_defs.h.

#### 7.79.2.22 #define CTR_IDIM1 19

Definition at line 88 of file structure_defs.h.

#### 7.79.2.23 #define CTR_IDIM2 20

Definition at line 89 of file structure_defs.h.

#### 7.79.2.24 #define CTR_IINACTIVE 26

Definition at line 95 of file structure_defs.h.

#### 7.79.2.25 #define CTR_IINACTIVE_HI 30

Definition at line 99 of file structure_defs.h.

#### 7.79.2.26 #define CTR_IINACTIVE_LO 28

Definition at line 97 of file structure_defs.h.

#### 7.79.2.27 #define CTR_IINTEGER 22

Definition at line 91 of file structure_defs.h.

#### 7.79.2.28 #define CTR_IMTDEG3P 17

Definition at line 86 of file structure_defs.h.

#### 7.79.2.29 #define CTR_IMTDIM2 21

Definition at line 90 of file structure_defs.h.

#### 7.79.2.30 #define CTR_INLN 18

Definition at line 87 of file structure_defs.h.

#### 7.79.2.31 #define CTR_INTEGER 1

Definition at line 33 of file structure_defs.h.

#### 7.79.2.32 #define CTR_IREDUNDANT 31

Definition at line 100 of file structure_defs.h.

#### 7.79.2.33 #define CTR_IRESTRICT 32

Definition at line 101 of file structure_defs.h.

#### 7.79.2.34 #define CTR_KJNODES 2

Definition at line 34 of file structure_defs.h.

#### 7.79.2.35 #define CTR_LEAVES 1

Definition at line 105 of file structure_defs.h.

#### 7.79.2.36 #define CTR_MIN 19

Definition at line 72 of file structure_defs.h.

#### 7.79.2.37 #define CTR_NODES 0

Definition at line 32 of file structure_defs.h.

#### 7.79.2.38 #define CTR_O1BOUNDED 7

Definition at line 60 of file structure_defs.h.

#### 7.79.2.39 #define CTR_O2BOUNDED 6

Definition at line 59 of file structure_defs.h.

#### 7.79.2.40 #define CTR_OBJ 0

Definition at line 53 of file structure_defs.h.

#### 7.79.2.41 #define CTR_OCONCAVE 18

Definition at line 71 of file structure_defs.h.

#### 7.79.2.42 #define CTR_OCONVEX 17

Definition at line 70 of file structure_defs.h.

#### 7.79.2.43 #define CTR_ODEG1 12

Definition at line 65 of file structure_defs.h.

#### 7.79.2.44 #define CTR_ODEG2 13

Definition at line 66 of file structure_defs.h.

#### 7.79.2.45 #define CTR_ODEG3 14

Definition at line 67 of file structure_defs.h.

#### 7.79.2.46 #define CTR_ODIM1 9

Definition at line 62 of file structure_defs.h.

#### 7.79.2.47 #define CTR_ODIM2 10

Definition at line 63 of file structure_defs.h.

#### 7.79.2.48 #define CTR_OEXISTS 3

Definition at line 56 of file structure_defs.h.

#### 7.79.2.49 #define CTR_OFORALL 4

Definition at line 57 of file structure_defs.h.

#### 7.79.2.50 #define CTR_OFREE 2

Definition at line 55 of file structure_defs.h.

#### 7.79.2.51 #define CTR_OINTEGER 1

Definition at line 54 of file structure_defs.h.

#### 7.79.2.52 #define CTR_OMTDEG3P 15

Definition at line 68 of file structure_defs.h.

#### 7.79.2.53 #define CTR_OMTDIM2 11

Definition at line 64 of file structure_defs.h.

#### 7.79.2.54 #define CTR_ONLN 16

Definition at line 69 of file structure_defs.h.

#### 7.79.2.55 #define CTR_OSTOCHASTIC 5

Definition at line 58 of file structure_defs.h.

#### 7.79.2.56 #define CTR_OUBOUNDED 8

Definition at line 61 of file structure_defs.h.

#### 7.79.2.57 #define CTR_ROOTS 2

Definition at line 106 of file structure_defs.h.

#### 7.79.2.58 #define CTR_STOCHASTIC 6

Definition at line 38 of file structure_defs.h.

#### 7.79.2.59 #define CTR_V1BOUNDED 8

Definition at line 49 of file structure_defs.h.

#### 7.79.2.60 #define CTR_V2BOUNDED 7

Definition at line 48 of file structure_defs.h.

#### 7.79.2.61 #define CTR_VARS 0

Definition at line 41 of file structure_defs.h.

#### 7.79.2.62 #define CTR_VEXISTS 3

Definition at line 44 of file structure_defs.h.

#### 7.79.2.63 #define CTR_VFORALL 4

Definition at line 45 of file structure_defs.h.

#### 7.79.2.64 #define CTR_VFREE 2

Definition at line 43 of file structure_defs.h.

#### 7.79.2.65 #define CTR_VINTEGER 1

Definition at line 42 of file structure_defs.h.

#### 7.79.2.66 #define CTR_VKJ 6

Definition at line 47 of file structure_defs.h.

#### 7.79.2.67 #define CTR_VSTOCHASTIC 5

Definition at line 46 of file structure_defs.h.

#### 7.79.2.68 #define CTR_VUBOUNDED 9

Definition at line 50 of file structure_defs.h.

#### 7.79.2.69 #define NUM_CCTR 33

Definition at line 102 of file structure_defs.h.

#### 7.79.2.70 #define NUM_CTR 7

Definition at line 39 of file structure_defs.h.

#### 7.79.2.71 #define NUM_ECTR 3

Definition at line 107 of file structure_defs.h.

#### 7.79.2.72 #define NUM_OCTR 20

Definition at line 73 of file structure_defs.h.

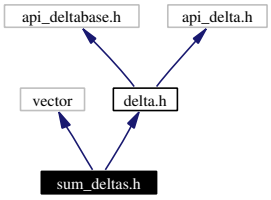
#### 7.79.2.73 #define NUM_VCTR 10

Definition at line 51 of file structure_defs.h.

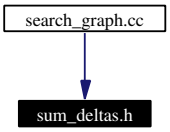
## 7.80 sum_deltas.h File Reference

```
#include <vector>
#include "delta.h"
```

Include dependency graph for sum_deltas.h:




This graph shows which files directly or indirectly include this file:




### Compounds

- class sum_deltas

### Functions

- work_node full_node_to_work_node (full_node &n_full, gptr< search_node > &ground)

#### 7.80.1 Detailed Description

Definition in file sum_deltas.h.

#### 7.80.2 Function Documentation

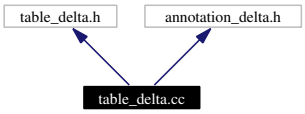
##### 7.80.2.1 work_node full_node_to_work_node (full_node & *n_full*, gptr< search_node > & *ground*) [inline]

Definition at line 33 of file sum_deltas.h.

## 7.81 table_delta.cc File Reference

```
#include <table_delta.h>
#include <annotation_delta.h>
```

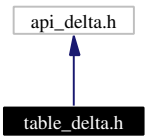
Include dependency graph for table_delta.cc:




## 7.82 table_delta.h File Reference

```
#include <api_delta.h>
```

Include dependency graph for table_delta.h:




### Compounds

- class table_delta

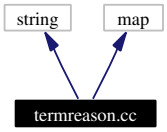
#### 7.82.1 Detailed Description

Definition in file table_delta.h.

## 7.83 termreason.cc File Reference

```
#include <string>
#include <map>
```

Include dependency graph for termreason.cc:




### Variables

- std::map< std::string, std::string > __termr_message__

#### 7.83.1 Detailed Description

Definition in file termreason.cc.

#### 7.83.2 Variable Documentation

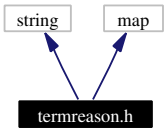
##### 7.83.2.1 std::map<std::string,std::string> __termr_message__

Definition at line 30 of file termreason.cc.

## 7.84 termreason.h File Reference

```
#include <string>
#include <map>
```

Include dependency graph for termreason.h:




### Compounds

- class termination_reason

### Functions

- std::ostream & operator<< (std::ostream &o, const termination_reason &__x)

### Variables

- std::map< std::string, std::string > __termr_message__

#### 7.84.1 Detailed Description

Definition in file termreason.h.

#### 7.84.2 Function Documentation

##### 7.84.2.1 std::ostream& operator<< (std::ostream & *o*, const termination_reason & *__x*) [inline]

Definition at line 79 of file termreason.h.

#### 7.84.3 Variable Documentation

##### 7.84.3.1 std::map<std::string,std::string> __termr_message__

Definition at line 35 of file termreason.h.