RingDecomposerLib 1.0

Generated by Doxygen 1.8.5

Wed Nov 2 2016 18:05:59

Contents

| 1 | Maiı | n Page | | 1 |
|---|------|----------|--|----|
| 2 | Mini | imal Exa | ample | 3 |
| 3 | Mini | imal RD | Kit Example | 5 |
| 4 | RDK | (it Exan | nple | 7 |
| 5 | Inst | allation | instructions | 11 |
| 6 | Hier | archica | ıl Index | 15 |
| | 6.1 | Class | Hierarchy | 15 |
| 7 | Clas | ss Index | (| 17 |
| | 7.1 | Class | List | 17 |
| 8 | File | Index | | 19 |
| | 8.1 | File Lis | st | 19 |
| 9 | Clas | ss Docu | umentation Control of the Control of | 21 |
| | 9.1 | py_rdl | .Calculator.Calculator Class Reference | 21 |
| | | 9.1.1 | Detailed Description | 22 |
| | | 9.1.2 | Constructor & Destructor Documentation | 22 |
| | | | 9.1.2.1init | 22 |
| | | 9.1.3 | Member Function Documentation | 22 |
| | | | 9.1.3.1getitem | 22 |
| | | | 9.1.3.2len | 22 |
| | | | 9.1.3.3 calculate | 22 |
| | | | 9.1.3.4 get_calculated_result | 23 |
| | | | 9.1.3.5 get_calculated_result_for_graph | 23 |
| | | | 9.1.3.6 get_edges_for_rcf | 23 |
| | | | 9.1.3.7 get_edges_for_ringsystem | 23 |
| | | | 9.1.3.8 get_edges_for_urf | 23 |
| | | | 9.1.3.9 get_nodes_for_rcf | 23 |

iv CONTENTS

| 9.1.3.11 get_nodes_for_urf 9.1.3.12 get_nof_rcf 9.1.3.13 get_nof_relevant_cycles 9.1.3.14 get_nof_relevant_cycles_for_urf 9.1.3.15 get_nof_ringsystems 9.1.3.16 get_nof_urf 9.1.3.17 get_rcfs_for_edge 9.1.3.18 get_rcfs_for_node 9.1.3.19 get_relevant_cycle_prototypes 9.1.3.20 get_relevant_cycles_for_rcf 9.1.3.21 get_relevant_cycles_for_urf 9.1.3.22 get_relevant_cycles_for_urf 9.1.3.23 get_sssr 9.1.3.24 get_urfs_for_edge 9.1.3.25 get_urfs_for_edge 9.1.3.26 get_weight_for_rcf 9.1.3.27 get_weight_for_urf 9.1.3.28 is_calculated 9.1.3.29 set_graph 9.1.4 Member Data Documentation 9.1.4.1 rcfs 9.1.4.2 ringsystems 9.1.4.3 urfs 9.2 py_rdl.wrapper.Cycle.Cycle Class Reference 9.2.1 Detailed Description | 24 24 24 24 24 24 25 25 25 25 |
|---|--|
| 9.1.3.13 get_nof_relevant_cycles 9.1.3.14 get_nof_relevant_cycles_for_urf 9.1.3.15 get_nof_ringsystems 9.1.3.16 get_nof_urf 9.1.3.17 get_cfs_for_edge 9.1.3.18 get_rcfs_for_node 9.1.3.19 get_relevant_cycle_prototypes 9.1.3.20 get_relevant_cycles_for_rcf 9.1.3.21 get_relevant_cycles_for_urf 9.1.3.22 get_relevant_cycles_for_urf 9.1.3.23 get_sssr 9.1.3.24 get_urfs_for_edge 9.1.3.25 get_urfs_for_node 9.1.3.26 get_weight_for_rcf 9.1.3.27 get_weight_for_urf 9.1.3.28 is_calculated 9.1.3.29 set_graph 9.1.4 Member Data Documentation 9.1.4.1 rcfs 9.1.4.2 ringsystems 9.1.4.3 urfs 9.2 py_rdl.wrapper.Cycle.Cycle Class Reference | 24 24 24 24 24 25 25 25 |
| 9.1.3.14 get_nof_relevant_cycles_for_urf 9.1.3.15 get_nof_ringsystems 9.1.3.16 get_nof_urf 9.1.3.17 get_rcfs_for_edge 9.1.3.18 get_rcfs_for_node 9.1.3.19 get_relevant_cycle_prototypes 9.1.3.20 get_relevant_cycles 9.1.3.21 get_relevant_cycles_for_rcf 9.1.3.22 get_relevant_cycles_for_urf 9.1.3.23 get_sssr 9.1.3.24 get_urfs_for_edge 9.1.3.25 get_urfs_for_node 9.1.3.26 get_weight_for_urf 9.1.3.27 get_weight_for_urf 9.1.3.28 is_calculated 9.1.3.29 set_graph 9.1.4 Member Data Documentation 9.1.4.1 rcfs 9.1.4.2 ringsystems 9.1.4.3 urfs | 24 24 24 24 25 25 25 25 |
| 9.1.3.15 get_nof_ringsystems 9.1.3.16 get_nof_urf 9.1.3.17 get_rcfs_for_edge 9.1.3.18 get_rcfs_for_node 9.1.3.19 get_relevant_cycle_prototypes 9.1.3.20 get_relevant_cycles 9.1.3.21 get_relevant_cycles_for_rcf 9.1.3.22 get_relevant_cycles_for_urf 9.1.3.23 get_sssr 9.1.3.24 get_urfs_for_edge 9.1.3.25 get_urfs_for_node 9.1.3.26 get_weight_for_rcf 9.1.3.27 get_weight_for_urf 9.1.3.28 is_calculated 9.1.3.29 set_graph 9.1.4 Member Data Documentation 9.1.4.1 rcfs 9.1.4.2 ringsystems 9.1.4.3 urfs 9.2 py_rdl.wrapper.Cycle.Cycle Class Reference | 24 24 24 25 25 25 25 |
| 9.1.3.16 get_nof_urf 9.1.3.17 get_rcfs_for_edge 9.1.3.18 get_rcfs_for_node 9.1.3.19 get_relevant_cycle_prototypes 9.1.3.20 get_relevant_cycles_for_rcf 9.1.3.21 get_relevant_cycles_for_urf 9.1.3.22 get_relevant_cycles_for_urf 9.1.3.23 get_sssr 9.1.3.24 get_urfs_for_edge 9.1.3.25 get_urfs_for_node 9.1.3.26 get_weight_for_rcf 9.1.3.27 get_weight_for_urf 9.1.3.28 is_calculated 9.1.3.29 set_graph 9.1.4 Member Data Documentation 9.1.4.1 rcfs 9.1.4.2 ringsystems 9.1.4.3 urfs | 24 24 25 25 25 25 |
| 9.1.3.17 get_rcfs_for_edge 9.1.3.18 get_rcfs_for_node 9.1.3.19 get_relevant_cycle_prototypes 9.1.3.20 get_relevant_cycles 9.1.3.21 get_relevant_cycles_for_rcf 9.1.3.22 get_relevant_cycles_for_urf 9.1.3.23 get_sssr 9.1.3.24 get_urfs_for_edge 9.1.3.25 get_urfs_for_node 9.1.3.26 get_weight_for_rcf 9.1.3.27 get_weight_for_urf 9.1.3.28 is_calculated 9.1.3.29 set_graph 9.1.4 Member Data Documentation 9.1.4.1 rcfs 9.1.4.2 ringsystems 9.1.4.3 urfs 9.2 py_rdl.wrapper.Cycle.Cycle Class Reference | 24 24 25 25 25 25 |
| 9.1.3.18 get_rcfs_for_node 9.1.3.19 get_relevant_cycle_prototypes 9.1.3.20 get_relevant_cycles 9.1.3.21 get_relevant_cycles_for_rcf 9.1.3.22 get_relevant_cycles_for_urf 9.1.3.23 get_sssr 9.1.3.24 get_urfs_for_edge 9.1.3.25 get_urfs_for_node 9.1.3.26 get_weight_for_rcf 9.1.3.27 get_weight_for_urf 9.1.3.28 is_calculated 9.1.3.29 set_graph 9.1.4 Member Data Documentation 9.1.4.1 rcfs 9.1.4.2 ringsystems 9.1.4.3 urfs 9.2 py_rdl.wrapper.Cycle.Cycle Class Reference | 24 25 25 25 25 |
| 9.1.3.19 get_relevant_cycle_prototypes 9.1.3.20 get_relevant_cycles 9.1.3.21 get_relevant_cycles_for_rcf 9.1.3.22 get_relevant_cycles_for_urf 9.1.3.23 get_sssr 9.1.3.24 get_urfs_for_edge 9.1.3.25 get_urfs_for_node 9.1.3.26 get_weight_for_rcf 9.1.3.27 get_weight_for_urf 9.1.3.28 is_calculated 9.1.3.29 set_graph 9.1.4 Member Data Documentation 9.1.4.1 rcfs 9.1.4.2 ringsystems 9.1.4.3 urfs 9.2 py_rdl.wrapper.Cycle.Cycle Class Reference | 25 25 25 25 |
| 9.1.3.20 get_relevant_cycles 9.1.3.21 get_relevant_cycles_for_rcf 9.1.3.22 get_relevant_cycles_for_urf 9.1.3.23 get_sssr 9.1.3.24 get_urfs_for_edge 9.1.3.25 get_urfs_for_node 9.1.3.26 get_weight_for_rcf 9.1.3.27 get_weight_for_urf 9.1.3.28 is_calculated 9.1.3.29 set_graph 9.1.4 Member Data Documentation 9.1.4.1 rcfs 9.1.4.2 ringsystems 9.1.4.3 urfs 9.2 py_rdl.wrapper.Cycle.Cycle Class Reference | 25 25 25 |
| 9.1.3.21 get_relevant_cycles_for_rcf 9.1.3.22 get_relevant_cycles_for_urf 9.1.3.23 get_sssr 9.1.3.24 get_urfs_for_edge 9.1.3.25 get_urfs_for_node 9.1.3.26 get_weight_for_rcf 9.1.3.27 get_weight_for_urf 9.1.3.28 is_calculated 9.1.3.29 set_graph 9.1.4 Member Data Documentation 9.1.4.1 rcfs 9.1.4.2 ringsystems 9.1.4.3 urfs 9.2 py_rdl.wrapper.Cycle.Cycle Class Reference | 25 25 |
| 9.1.3.22 get_relevant_cycles_for_urf 9.1.3.23 get_sssr 9.1.3.24 get_urfs_for_edge 9.1.3.25 get_urfs_for_node 9.1.3.26 get_weight_for_rcf 9.1.3.27 get_weight_for_urf 9.1.3.28 is_calculated 9.1.3.29 set_graph 9.1.4 Member Data Documentation 9.1.4.1 rcfs 9.1.4.2 ringsystems 9.1.4.3 urfs 9.2 py_rdl.wrapper.Cycle.Cycle Class Reference | 25 |
| 9.1.3.23 get_sssr 9.1.3.24 get_urfs_for_edge 9.1.3.25 get_urfs_for_node 9.1.3.26 get_weight_for_rcf 9.1.3.27 get_weight_for_urf 9.1.3.28 is_calculated 9.1.3.29 set_graph 9.1.4.1 rcfs 9.1.4.2 ringsystems 9.1.4.3 urfs 9.2 py_rdl.wrapper.Cycle.Cycle Class Reference | |
| 9.1.3.24 get_urfs_for_edge 9.1.3.25 get_urfs_for_node 9.1.3.26 get_weight_for_rcf 9.1.3.27 get_weight_for_urf 9.1.3.28 is_calculated 9.1.3.29 set_graph 9.1.4 Member Data Documentation 9.1.4.1 rcfs 9.1.4.2 ringsystems 9.1.4.3 urfs 9.2 py_rdl.wrapper.Cycle.Cycle Class Reference | 25 |
| 9.1.3.25 get_urfs_for_node 9.1.3.26 get_weight_for_rcf 9.1.3.27 get_weight_for_urf 9.1.3.28 is_calculated 9.1.3.29 set_graph 9.1.3.29 set_graph 9.1.4.1 rcfs 9.1.4.2 ringsystems 9.1.4.2 ringsystems 9.1.4.3 urfs 9.2 py_rdl.wrapper.Cycle.Cycle Class Reference 9.1.4.3 Reference | |
| 9.1.3.26 get_weight_for_rcf 9.1.3.27 get_weight_for_urf 9.1.3.28 is_calculated 9.1.3.29 set_graph 9.1.4 Member Data Documentation 9.1.4.1 rcfs 9.1.4.2 ringsystems 9.1.4.3 urfs 9.2 py_rdl.wrapper.Cycle.Cycle Class Reference | 25 |
| 9.1.3.27 get_weight_for_urf 9.1.3.28 is_calculated 9.1.3.29 set_graph 9.1.4 Member Data Documentation 9.1.4.1 rcfs 9.1.4.2 ringsystems 9.1.4.3 urfs 9.2 py_rdl.wrapper.Cycle.Cycle Class Reference | 25 |
| 9.1.3.28 is_calculated | 25 |
| 9.1.3.29 set_graph | 25 |
| 9.1.4 Member Data Documentation 9.1.4.1 rcfs 9.1.4.2 ringsystems 9.1.4.3 urfs 9.2 py_rdl.wrapper.Cycle.Cycle Class Reference | 26 |
| 9.1.4.1 rcfs 9.1.4.2 ringsystems 9.1.4.3 urfs 9.2 py_rdl.wrapper.Cycle.Cycle Class Reference | 26 |
| 9.1.4.2 ringsystems | 26 |
| 9.1.4.3 urfs | 26 |
| 9.2 py_rdl.wrapper.Cycle.Cycle Class Reference | 26 |
| | 26 |
| 9.2.1 Detailed Description | 26 |
| | 27 |
| 9.2.2 Constructor & Destructor Documentation | 27 |
| 9.2.2.1init | 27 |
| 9.2.3 Member Function Documentation | 27 |
| 9.2.3.1 weight | 27 |
| 9.3 py_rdl.CycleFamily.CycleFamily Class Reference | 27 |
| 9.3.1 Detailed Description | 28 |
| 9.3.2 Constructor & Destructor Documentation | 28 |
| 9.3.2.1init | 28 |
| 9.4 py_rdl.Graph.Graph Class Reference | 28 |
| 9.4.1 Detailed Description | 28 |
| 9.4.2 Constructor & Destructor Documentation | 28 |
| 9.4.2.1init | 28 |
| 9.4.3 Member Function Documentation | 28 |
| 9.4.3.1 add_edge | 28 |

CONTENTS

| | | 9.4.3.2 from_edges | 29 |
|---------|---------|--|----|
| | | 9.4.3.3 get_edge_for_indices | 29 |
| | | 9.4.3.4 get_edges | 29 |
| | | 9.4.3.5 get_index_for_node | 29 |
| | | 9.4.3.6 get_indices_for_edge | 29 |
| | | 9.4.3.7 get_node_for_index | 30 |
| | | 9.4.3.8 get_nof_nodes | 30 |
| 9.5 | py_rdl. | CycleFamily.RCF Class Reference | 30 |
| | 9.5.1 | Detailed Description | 30 |
| | 9.5.2 | Constructor & Destructor Documentation | 31 |
| | | 9.5.2.1init | 31 |
| | 9.5.3 | Member Function Documentation | 31 |
| | | 9.5.3.1str | 31 |
| 9.6 | RDL_d | cycle Struct Reference | 31 |
| | 9.6.1 | Detailed Description | 31 |
| | 9.6.2 | Member Data Documentation | 31 |
| | | 9.6.2.1 edges | 31 |
| | | 9.6.2.2 rcf | 31 |
| | | 9.6.2.3 urf | 31 |
| | | 9.6.2.4 weight | 32 |
| 9.7 | py_rdl. | CycleFamily.ResultCollection Class Reference | 32 |
| | 9.7.1 | Detailed Description | 32 |
| | 9.7.2 | Constructor & Destructor Documentation | 32 |
| | | 9.7.2.1init | 32 |
| | 9.7.3 | Member Function Documentation | 32 |
| | | 9.7.3.1repr | 32 |
| | | 9.7.3.2str | 33 |
| 9.8 | py_rdl. | CycleFamily.Ringsystem Class Reference | 33 |
| | 9.8.1 | Detailed Description | 33 |
| | 9.8.2 | Constructor & Destructor Documentation | 33 |
| | | 9.8.2.1init | 33 |
| 9.9 | py_rdl. | CycleFamily.URF Class Reference | 33 |
| | 9.9.1 | Detailed Description | 34 |
| | 9.9.2 | Constructor & Destructor Documentation | 34 |
| | | 9.9.2.1init | 34 |
| | 9.9.3 | Member Function Documentation | 34 |
| | | 9.9.3.1str | 34 |
| 10 File | Docum | entation | 35 |
| | | nimalExample/MinimalExample c File Reference | 35 |

vi CONTENTS

| | 10.1.1 | Detailed Description | 35 |
|------|---------|---|----|
| 10.2 | src/Rin | gDecomposerLib/RingDecomposerLib.h File Reference | 35 |
| | 10.2.1 | Detailed Description | 38 |
| | 10.2.2 | Typedef Documentation | 38 |
| | | 10.2.2.1 RDL_cycle | 38 |
| | | 10.2.2.2 RDL_cycleIterator | 39 |
| | | 10.2.2.3 RDL_data | 39 |
| | | 10.2.2.4 RDL_graph | 39 |
| | 10.2.3 | Function Documentation | 39 |
| | | 10.2.3.1 RDL_addUEdge | 39 |
| | | 10.2.3.2 RDL_calculate | 39 |
| | | 10.2.3.3 RDL_cycleIteratorAtEnd | 40 |
| | | 10.2.3.4 RDL_cycleIteratorGetCycle | 40 |
| | | 10.2.3.5 RDL_cycleIteratorNext | 40 |
| | | 10.2.3.6 RDL_deleteCycle | 41 |
| | | 10.2.3.7 RDL_deleteCycleIterator | 42 |
| | | 10.2.3.8 RDL_deleteCycles | 42 |
| | | 10.2.3.9 RDL_deleteData | 42 |
| | | 10.2.3.10 RDL_deleteEdgeldxArray | 42 |
| | | 10.2.3.11 RDL_deleteGraph | 42 |
| | | 10.2.3.12 RDL_getEdgeArray | 43 |
| | | 10.2.3.13 RDL_getEdgeld | 43 |
| | | 10.2.3.14 RDL_getEdgesForRCF | 43 |
| | | 10.2.3.15 RDL_getEdgesForRingsystem | 44 |
| | | 10.2.3.16 RDL_getEdgesForURF | 45 |
| | | 10.2.3.17 RDL_getNodesForRCF | 45 |
| | | 10.2.3.18 RDL_getNodesForRingsystem | 46 |
| | | 10.2.3.19 RDL_getNodesForURF | 46 |
| | | 10.2.3.20 RDL_getNofEdges | 46 |
| | | 10.2.3.21 RDL_getNofEdgesForRingsystem | 47 |
| | | 10.2.3.22 RDL_getNofNodesForRingsystem | 47 |
| | | 10.2.3.23 RDL_getNofRC | 47 |
| | | 10.2.3.24 RDL_getNofRCF | 47 |
| | | 10.2.3.25 RDL_getNofRCFContainingEdge | 48 |
| | | 10.2.3.26 RDL_getNofRCFContainingNode | 48 |
| | | 10.2.3.27 RDL_getNofRCForRCF | 48 |
| | | 10.2.3.28 RDL_getNofRCForURF | 49 |
| | | 10.2.3.29 RDL_getNofRingsystems | 49 |
| | | 10.2.3.30 RDL_getNofURF | 49 |
| | | 10.2.3.31 RDL_getNofURFContainingEdge | 49 |

CONTENTS vii

| 10.2.3.32 RDL_g | etNofURFContainingNode . | | 50 |
|----------------------------------|---------------------------|------|--------|
| 10.2.3.33 RDL_g | etRCFsContainingEdge | | 50 |
| 10.2.3.34 RDL_g | etRCFsContainingNode | | 5 |
| 10.2.3.35 RDL_g | etRCPrototypes | | 5 |
| 10.2.3.36 RDL_g | etRCycles | | 5 |
| 10.2.3.37 RDL_g | etRCyclesForRCF | | 52 |
| 10.2.3.38 RDL_g | etRCyclesForRCFIterator . | | 52 |
| 10.2.3.39 RDL_g | etRCyclesForURF | | 52 |
| 10.2.3.40 RDL_g | etRCyclesForURFIterator . | | 53 |
| 10.2.3.41 RDL_g | etRCyclesIterator | | 53 |
| 10.2.3.42 RDL_g | etRingsystemForEdge | | 53 |
| 10.2.3.43 RDL_g | etSSSR | | 53 |
| 10.2.3.44 RDL_g | etURFsContainingEdge | | 54 |
| 10.2.3.45 RDL_g | etURFsContainingNode | | 54 |
| 10.2.3.46 RDL_g | etWeightForRCF | | 54 |
| 10.2.3.47 RDL_g | etWeightForURF | | 55 |
| 10.2.3.48 RDL_ii | nitNewGraph | | 55 |
| 10.2.3.49 RDL_s | etOutputFunction | | 55 |
| 10.2.3.50 RDL_t | anslateCycArray | | 55 |
| 10.3 src/Test/Test.c File Refere | ence | | 56 |
| 10.3.1 Detailed Descript | ion | | 56 |
| | | | |
| | | | |

58

Index

Main Page

The RingDecomposerLib is a library for calculation of the unique ring families and related ring topology descriptions.

It is developed at the University of Hamburg, ZBH - Center for Bioinformatics by Niek Andresen, Florian Flachsenberg and Matthias Rarey.

The library is distributed under BSD New license, see the file LICENSE and BSD New.

Please cite:

Kolodzik, A.; Urbaczek, S.; Rarey, M. Unique Ring Families: A Chemically Meaningful Description of Molecular Ring Topologies. J. Chem. Inf. Model., 2012, 52, pp 2013-2021

Flachsenberg, F.; Andresen, N.; Rarey, M. RingDecomposerLib: An Open-Source Implementation of Unique Ring Families and other Cycle Bases. SUBMITTED

This package contains a C-library as well as an optional Python wrapper for the library.

These pages are the documentation of the RingDecomposerLib library and the Python wrapper.

For installation instructions see here: Installation instructions

Core library RingDecomposerLib

The documentation of the C library is available here: RingDecomposerLib.h

The testing util for the library documented is here: Test.c

A minimal example can be found here: Minimal Example

Python wrapper py_urf

The documentation of the Python wrapper can be found here: py_rdl.Calculator.Calculator, py_rdl.Graph.Graph

Example scripts that show how to use the Python wrapper together with RDKit can be found here: Minimal RDKit Example, RDKit Example

A minimal example im Python looks like this:

```
import py_rdl
data = py_rdl.Calculator.get_calculated_result([(0,1),(1,2),(2,3),(3,4),(4,5),(5,0)])
print(data.get_nof_urf())
```

2 Main Page

Minimal Example

A minimal example for the C library is:

RDL_deleteData(data);

```
* This file is part of the RingDecomposerLib, licensed * under BSD New license (see LICENSE in the root directory).
 * Copyright (c) 2016
 * University of Hamburg, ZBH - Center for Bioinformatics
 \star Niek Andresen, Florian Flachsenberg, Matthias Rarey
* Kolodzik, A.; Urbaczek, S.; Rarey, M.
* Unique Ring Families: A Chemically Meaningful Description
 * of Molecular Ring Topologies.
 * J. Chem. Inf. Model., 2012, 52, pp 2013-2021
* Flachsenberg, F.; Andresen, N.; Rarey, M. * RingDecomposerLib: An Open-Source Implementation of
 * Unique Ring Families and other Cycle Bases.
 * SUBMITTED
#include <assert.h>
#include <stdio.h>
#include "RingDecomposerLib.h"
int main()
 RDL_graph* graph;
 RDL_data* data;
  unsigned nof_urf, i;
 double nof_rc;
 const unsigned nof_edges=10, nof_nodes=8;
 graph = RDL_initNewGraph(nof_nodes);
  for (i = 0; i < nof_edges; ++i) {
   RDL_addUEdge(graph, edges[i][0], edges[i][1]);
  /* step 2: calculate URFs, RCFs */
  data = RDL_calculate(graph);
  /* check that calculation was successful */
  assert (data != NULL);
 /* step 3: how many URFs and RCs does the graph have? */
nof_urf = RDL_getNofURF(data);
 nof_rc = RDL_getNofRC(data);
  RDL_cycleIterator *it = RDL_getRCyclesIterator(data);
  while(!RDL_cycleIteratorAtEnd(it)) {
    cycle = RDL_cycleIteratorGetCycle(it);
    /* <do something with the cycle> */
    RDL_deleteCycle(cycle);
    RDL_cycleIteratorNext(it);
  RDL_deleteCycleIterator(it);
  /* delete data and graph */
```

4 Minimal Example

```
printf("URFs: %d\n RCs: %.0f\n", nof_urf, nof_rc);
   return 0;
}
```

Minimal RDKit Example

A minimal example for using the RingDecomposerLib together with RDKit is:

```
1 # This file is part of the RingDecomposerLib, licensed
2 # under BSD New license (see LICENSE in the root directory).
3 # Copyright (c) 2016
4 # University of Hamburg, ZBH - Center for Bioinformatics
5 # Niek Andresen, Florian Flachsenberg, Matthias Rarey
  # Please cite:
9 # Kolodzik, A.; Urbaczek, S.; Rarey, M.
10 # Unique Ring Families: A Chemically Meaningful Description
11 # of Molecular Ring Topologies.
12 # J. Chem. Inf. Model., 2012, 52, pp 2013-2021
13 #
14 # Flachsenberg, F.; Andresen, N.; Rarey, M.
15 # RingDecomposerLib: An Open-Source Implementation of
16 # Unique Ring Families and other Cycle Bases.
17 # SUBMITTED
18
19 from __future__ import print_function
20 import py_rdl
21 import rdkit.Chem
23 # p-cyclophane as URF example
24 molecule = rdkit.Chem.MolFromSmiles('Clc2ccc(cc2)CCc2ccc(cc2)Cl')
26 # calculation step
27 data = py_rdl.Calculator.get_calculated_result(
           molecule.GetBonds(), # pass edges
rdkit.Chem.Bond.GetBeginAtom, # get first atom
                rdkit.Chem.Bond.GetEndAtom, # get second atom
rdkit.Chem.Atom.GetIdx, # identify atom
rdkit.Chem.Bond.GetIdx # identify edge
31
32
          )
33
35 # iterate over URFs
36 for urf in data.urfs:
37
        # print URF
      print(urf)
# URF can be used as an index, get RCs
rcs = data.get_relevant_cycles_for_urf(urf)
# print RCs
for rc in rcs:
38
39
40
42
          print(rc)
```

| 6 | Minimal RDKit Example |
|---|-----------------------|
| - | |

RDKit Example

A more complex examplefor using the RingDecomposerLib together with RDKit is:

```
1 # This file is part of the RingDecomposerLib, licensed
2 # under BSD New license (see LICENSE in the root directory).
3 # Copyright (c) 2016
4 # University of Hamburg, ZBH - Center for Bioinformatics
5 # Niek Andresen, Florian Flachsenberg, Matthias Rarey
7 # Please cite:
8 #
9 # Kolodzik, A.; Urbaczek, S.; Rarey, M.
10 # Unique Ring Families: A Chemically Meaningful Description
11 # of Molecular Ring Topologies.
12 # J. Chem. Inf. Model., 2012, 52, pp 2013-2021
13 #
14 # Flachsenberg, F.; Andresen, N.; Rarey, M.
15 # RingDecomposerLib: An Open-Source Implementation of
16 # Unique Ring Families and other Cycle Bases.
17 # SUBMITTED
19 ## @file rdkit_demo.py
20 \#\# @brief This file containes simple demo for using the
21 ## Python wrapper
22 ## of the RingDecomposerLib together with RDKit.
25 ##
26 ##
           rdkit_demo.py --input <input_file> [--output]
27 ##
28 ## The program takes as input a molecule file
29 ## (.smi or .sdf) and reads in every molecule and
30 ## calculates the ring topology and prints it to stdout.
32 ## The parameter '--output' is optional. If specified,
33 ## the script will write a number of SVG images,
34 ## where RCs are marked according to family membership.
36 from __future__ import absolute_import, print_function
38 import argparse
39 import os
40 import random
41 import sys
43 import py_rdl
44
45 import rdkit.Chem
46 import rdkit.Chem.AllChem
47 import rdkit.Chem.Draw
48
49
50 def analyze_molecule(molecule):
51
     data = py_rdl.Calculator.get_calculated_result(
                                                     # pass edges
                  molecule.GetBonds(),
52
                   rdkit.Chem.Bond.GetBeginAtom,
53
                                                          # get first atom
                   rdkit.Chem.Bond.GetEndAtom, # get second atom rdkit.Chem.Atom.GetIdx, # identify atom
                                                          # identify edge
                    rdkit.Chem.Bond.GetIdx
57
               )
58
     print('URFs')
59
       for urf in data:
60
         print(urf)
```

8 RDKit Example

```
62
          print(urf.weight)
           print(urf.nodes)
64
           print (urf.edges)
6.5
           print(data.get_relevant_cycles_for_urf(urf))
66
      print('URF for each atom')
67
       for a in molecule.GetAtoms():
68
69
           print(a.GetIdx(), data.get_urfs_for_node(a.GetIdx()))
70
71
       print('URF for each bond')
72
       for b in molecule.GetBonds():
73
         print(b.GetIdx(), data.get_urfs_for_edge(b.GetIdx()))
74
75
       print('MCB')
76
       print(data.get_sssr())
77
78
       return data
79
80
81 def get_rgb_from_hsv(h, s, v, a):
       def clamp(v, minv, maxv):
82
83
           return max(minv, min(v, maxv))
84
      8.5
86
88
89
      i = int(h / 60.0)
      r = (h % 60.0) / 60.0
90
      p = v * (1.0 - s)

q = v * (1.0 - s * r)
91
92
93
       t = v * (1.0 - s * (1.0 - r))
94
9.5
       if i == 0 or i == 6:
       return (v, t, p, a)
elif i == 1:
96
97
98
           return (q, v, p, a)
       elif i == 2:
99
100
            return (p, v, t, a)
101
       elif i == 3:
102
       return (p, q, v, a)
elif i == 4:
103
104
           return (t, p, v, a)
       elif i == 5:
105
106
           return (v, p, q, a)
107
108
        raise ValueError("Internal color error")
109
110
111 def get_distinct_colors(nof_colors, h=0.0, s=0.8, v=0.4, a=1.0, variation=0.2):
112
        colors = []
113
114
        for i in range(nof_colors):
           h_ = (h + i * 360.0 / nof_colors) % 360.0

s_ = s + variation * random.random()

v_ = v + variation + random.random()
115
116
117
118
            colors.append(get_rgb_from_hsv(h_, s_, v_, a)[:-1])
119
120
        return colors
121
122
123 def draw molecule (mol, filename, higlights):
124
        rdkit.Chem.AllChem.Compute2DCoords(mol)
125
126
        drawer = rdkit.Chem.Draw.rdMolDraw2D.MolDraw2DSVG(200, 200)
127
        drawer.DrawMolecule(mol, highlightAtoms=None, highlightBonds=highlights,
128
            highlightAtomColors=None, highlightBondColors=highlights)
129
130
        drawer.FinishDrawing()
131
        svg = drawer.GetDrawingText().replace('svg:', '')
132
        with open(filename, 'w') as outfile:
133
134
            outfile.write(svg)
135
136
137 if __name__ == '__main__':
138
        parser = argparse.ArgumentParser('RDKit ring family example')
        139
140
        141
142
143
144
        args = parser.parse_args()
145
146
        ext = os.path.splitext(args.input)[1]
147
148
        if ext == '.smi':
```

```
supplier = rdkit.Chem.SmilesMolSupplier(args.input, titleLine=False)
150
        elif ext == '.sdf':
151
            supplier = rdkit.Chem.SDMolSupplier(args.input)
        else:
152
            raise ValueError('invalid extension: %s' % ext)
153
154
155
        for i, mol in enumerate(supplier):
156
            print("Molecule", i)
157
             if mol is None:
158
                 print("ERROR")
159
            if mol.HasProp("_Name"):
160
                name = mol.GetProp("_Name")
161
162
163
                 name = "NO_NAME"
164
             print(name)
165
             data = analyze_molecule(mol)
            if args.output:
166
167
                rcps = data.get_relevant_cycle_prototypes()
168
                 rcf_colors = get_distinct_colors(len(rcps))
169
170
                 urf_colors = get_distinct_colors(data.get_nof_urf())
                for j, r in enumerate(rcps):
   outfile_name = args.input + '_{}_rcp_{}.svg'.format(i, j)
   highlights = {b: rcf_colors[j] for b in r.edges}
171
172
173
174
                     draw_molecule(mol, outfile_name, highlights)
175
                 for j, r in enumerate(data.get_relevant_cycles()):
176
                     outfile_name = args.input + '_{}_rc_{}_rcf.svg'.format(i, j)
                     highlights = {b: rcf_colors[r.rcf.index] for b in r.edges}
177
                     draw_molecule(mol, outfile_name, highlights)
178
179
180
                     outfile_name = args.input + '_{}_rc_{}_urf.svg'.format(i, j)
181
                     highlights = {b: urf_colors[r.urf.index] for b in r.edges}
182
                     draw_molecule(mol, outfile_name, highlights)
```

10 **RDKit Example**

Installation instructions

This document describes the installation procedure for the RingDecomposerLib library.

This package uses CMake for the build process. See the CMake documentation for general configuration options.

The most useful option is -DCMAKE_INSTALL_PREFIX=<your/target/path> for changing the install directory.

Core library

Prerequisites

For the core library the build process needs CMake (>= 3.1) and an ANSI-C compiler (for example gcc, clang or MSVC).

Installation Unix and macOS

Installation on Unix using make and the system's default C compiler.

```
mkdir build
cd build
cmake .. -DCMAKE_BUILD_TYPE=Release
make && make install
```

Test

Use the following command to test the build of the core library. Testing is strongly recommended.

```
make test
```

The program will compare the relevant cycles to the cycles present in the input file. Furthermore the URFs will be validated, i.e. calculated with an independent exponential algorithm (definition of the URFs). The SSSR will also be validated (it is checked if it's a cycle base). A number of consistency tests is executed to ensure integrity and robustness. See the file test/README for a descriptions of the test files.

Warning

All tests should succeed! If any test fails, check the output. On exceptionally slow machines the failure could result from timeouts.

12 Installation instructions

Installation Windows

Installation on Windows using the Visual Studio C compiler and MSBuild.exe from the command line (cmd.exe). Replace Visual Studio 14 2015 by your Version.

```
mkdir build
cd build
cmake.exe -G"Visual Studio 14 2015" ..
MSBuild.exe ALL_BUILD.vcxproj /property:Configuration=Release
MSBuild.exe INSTALL.vcxproj /property:Configuration=Release
```

Use the following command to test the build of the core library.

```
MSBuild.exe RUN_TESTS.vcxproj /property:Configuration=Release
```

Python wrapper

The Python wrapper is optional.

Prerequisites

The Python wrapper needs additionally Python (≥ 2.7 or ≥ 3.3), and Cython.

Installation Unix and macOS

On Unix use the same commands as for installation of the core library with the CMake option -DBUILD_PYTHON-WRAPPER=ON

```
\verb|cmake| ... - \verb|DCMAKE_BUILD_TYPE=Release| - \verb|DBUILD_PYTHON_WRAPPER=ON| \\
```

Depending on your system configuration you may have to specify additional options.

If your Python executable isn't in your current \$PATH – or you want to use another – specify with:

cmake .. -DCMAKE_BUILD_TYPE=Release -DBUILD_PYTHON_WRAPPER=ON -DPYTHON_EXECUTABLE=<path/to/your/python/executa

Use <code>-DPYTHON_FLAGS</code> to modify the installation behaviour of the python wrapper If you can't or don't want to install the Python wrapper into the default directory, specify the target with:

or

```
cmake .. -DCMAKE_BUILD_TYPE=Release -DBUILD_PYTHON_WRAPPER=ON -DPYTHON_FLAGS="--prefix=<python/library/install
```

Use the following command to test the core library and the Python wrapper. Testing is strongly recommended.

make test

Warning

The Python wrapper will already be installed during make and not make install. If you don't want the wrapper to be installed, set -DPYTHON_FLAGS="--prefix=/some/temp/folder". In that case the Python tests will fail.

Installation Windows

On Windows use

```
cmake.exe -G"Visual Studio 14 2015" .. -DBUILD_PYTHON_WRAPPER=ON
```

Warning

Before compiling with MSBuild.exe you may have to set one environment variable. This is neccessary because Python2.7 on Windows usually is built using Visual Studio 9 2008. Replace VS140COM-NTOOLS by your version.

```
SET VS90COMNTOOLS=%VS140COMNTOOLS%
```

The other options are the same as for Unix.

Warning

The Python wrapper will already be installed during the build step and not the install step. If you don't want the wrapper to be installed, set $-DPYTHON_FLAGS="--prefix=/some/temp/folder"$. In that case the Python tests will fail.

C++ RDKit Benchmark

The C++ RDKit Benchmark is optional. It demonstrates the usage with RDKit and is the tool used for benchmarking the library in the paper.

Prerequisites

The C++ RDKit Benchmark needs RDKit (>= 2016.03.01) to be installed and an C++ compiler (g++, clang++, MSCV). Additionally the RDKit header files (for example librdkit-dev package for Debian) and the boost header files (for example libboost-dev for Debian) are needed (>= 1.45).

Installation (Unix and macOS)

Use the same commands as for Installation of the core library with the CMake option -DBUILD_RDKIT_BENCHM-ARK=ON

```
cmake .. -DCMAKE_BUILD_TYPE=Release -DBUILD_RDKIT_BENCHMARK=ON
```

Depending on your system configuration you may have to specify additional options.

If the boost libraries are not in the default search path, change with -DBOOST_ROOT (see FindBoost for details). If the RDKit headers are not in a default search path, specify with:

```
cmake .. -DCMAKE_BUILD_TYPE=Release -DBUILD_RDKIT_BENCHMARK=ON -DRDKIT_INCLUDE_DIR=<path/to/your/rdkit/headers
```

Warning

Because of the complex dependencies the RDKit Benchmark is currently untested on Windows.

14 Installation instructions

Hierarchical Index

6.1 Class Hierarchy

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

| py_rdl.Calculator.Calculator | 21 |
|-------------------------------------|----|
| py_rdl.wrapper.Cycle.Cycle | 26 |
| py_rdl.Graph.Graph | 28 |
| RDL_cycle | 31 |
| py_rdl.CycleFamily.ResultCollection | 32 |
| py_rdl.CycleFamily.CycleFamily | 27 |
| py_rdl.CycleFamily.RCF | 30 |
| py_rdl.CycleFamily.URF | 33 |
| py_rdl.CycleFamily.Ringsystem | 33 |

16 **Hierarchical Index**

Class Index

7.1 Class List

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

| py_rdl.Calculator.Calculator | 21 |
|-------------------------------------|----|
| py_rdl.wrapper.Cycle.Cycle | 26 |
| py_rdl.CycleFamily.CycleFamily | 27 |
| py_rdl.Graph.Graph | 28 |
| py_rdl.CycleFamily.RCF (| 30 |
| RDL_cycle | 31 |
| py_rdl.CycleFamily.ResultCollection | 32 |
| py_rdl.CycleFamily.Ringsystem | 33 |
| py_rdl.CycleFamily.URF | 33 |

18 Class Index

File Index

8.1 File List

| Here is a list of al | documented f | iles with b | rief descriptions |
|----------------------|--------------|-------------|-------------------|
|----------------------|--------------|-------------|-------------------|

| src/MinimalExample/MinimalExample.c | |
|---|----|
| Minimal example for the RingDecomposerLib library | 35 |
| src/RingDecomposerLib/RingDecomposerLib.h | |
| This file contains the API of the RingDecomposerLib library | 35 |
| src/Test/Test.c | |
| Demo and validation tool for the RingDecomposerLib library | 56 |

20 File Index

Class Documentation

9.1 py_rdl.Calculator.Calculator Class Reference

Public Member Functions

- def __init__
- def get_calculated_result_for_graph
- · def get_calculated_result
- · def set graph
- · def calculate
- · def is_calculated
- · def get_nof_urf
- def get_nof_rcf
- def get_nof_ringsystems
- def len
- def get_nodes_for_urf
- · def get_nodes_for_rcf
- def get_nodes_for_ringsystem
- def get_edges_for_urf
- def get_edges_for_rcf
- def get_edges_for_ringsystem
- def get_weight_for_urf
- · def get_weight_for_rcf
- def get_relevant_cycles_for_urf
- def get_relevant_cycles_for_rcf
- def get_urfs_for_node
- def get_urfs_for_edge
- · def get_rcfs_for_node
- def get_rcfs_for_edge
- def get_sssr
- def get_relevant_cycles
- def get_nof_relevant_cycles
- def get_nof_relevant_cycles_for_urf
- def get_relevant_cycle_prototypes
- def __getitem__

22 Class Documentation

Public Attributes

• urfs

List of calculated URFs.

· rcfs

List of calculated RCFs.

· ringsystems

List of calculated ring systems.

9.1.1 Detailed Description

Object for representing a calculation result.

9.1.2 Constructor & Destructor Documentation

9.1.2.1 def py_rdl.Calculator.Calculator.__init__ (self, graph)

Initialize a new Calculator object with a Graph.

Use @classmethod provided instead (get_calculated_result_for_graph, get_calculated_result).

You can access URFs, RCFs and ring systems with the public attributes 'urfs', 'rcfs' and 'ringsystems'. The objects can be uses as indices for all functions below, that operate on the respective ring descriptions.

9.1.3 Member Function Documentation

9.1.3.1 def py_rdl.Calculator.Calculator.__getitem__ (self, item)

Get the item-th URF. item: The index of the URF raises RDLError if calculation wasn't successful raises IndexError if the index is out of range returns the item-th URF.

9.1.3.2 def py_rdl.Calculator.Calculator.__len__ (self)

Builtin for getting the length (number of URFs).

9.1.3.3 def py_rdl.Calculator.Calculator.calculate (self)

Calculate results for the graph datastructure. raises RDLError, if the calculation fails

When creating a Calculator object with the @classmethod provided (get_calculated_result_for_graph, get_calculated_result) calling this function is not necessary.

9.1.3.4 def py_rdl.Calculator.Calculator.get_calculated_result (cls, edge_iterable, get_node_1 = operator.itemgetter(0), get_node_2 = operator.itemgetter(1), get_node_id = lambda x: x, get_edge_id = lambda x: x) Calculate the ring topologies and return the results. edge_iterable: an iterable containing the edges get_node_1: function retrieving the first node of an edge get_node_2: function retrieving the second node of an edge get_node_id: function for retrieving node identifier get_edge_id: function for retrieving edge identifier returns the calculated Calculator This @classmethod creates a new Graph. The edge_iterable can be any iterable (list, generator etc.). An edge can be as simple as a pair of nodes. The get_node function must return the respective nodes adjacent to an edge. The node must be hashable. If it's not, use some unique ID instead. The edge must be hashable. If it's not, use some unique ID instead. 9.1.3.5 def py_rdl.Calculator.Calculator.get_calculated_result_for_graph (cls, graph) Calculate the ring topologies for a Graph and return the results. 9.1.3.6 def py_rdl.Calculator.Calculator.get_edges_for_rcf (self, rcf_index) Get the edges in this RCF. raises RDLError if calculation wasn't successful returns list of edges in this RCF 9.1.3.7 def py_rdl.Calculator.Calculator.get_edges_for_ringsystem (self, rs_index) Get the edges in this Ringsystem. A ring system is a 2-connected component of the graph. raises RDLError if calculation wasn't successful returns list of edges in this Ringsystem 9.1.3.8 def py_rdl.Calculator.Calculator.get_edges_for_urf (self, urf_index) Get the edges in this URF. raises RDLError if calculation wasn't successful returns list of edges in this URF

Generated on Wed Nov 2 2016 18:05:59 for RingDecomposerLib 1.0 by Doxygen

raises RDLError if calculation wasn't successful

Get the nodes in this RCF.

returns list of nodes in this RCF

9.1.3.9 def py_rdl.Calculator.Calculator.get_nodes_for_rcf (self, rcf_index)

24 Class Documentation

9.1.3.10 def py_rdl.Calculator.Calculator.get_nodes_for_ringsystem (self, rs_index)

Get the nodes in this Ringsystem.
A ring system is a 2-connected component of the graph.
raises RDLError if calculation wasn't successful
returns list of nodes in this Ringsystem

9.1.3.11 def py_rdl.Calculator.Calculator.get_nodes_for_urf (self, urf_index)

Get the nodes in this URF. raises RDLError if calculation wasn't successful returns list of nodes in this URF

9.1.3.12 def py_rdl.Calculator.Calculator.get_nof_rcf (self)

Get the number of RCFs. raises RDLError if calculation wasn't successful

9.1.3.13 def py_rdl.Calculator.Calculator.get_nof_relevant_cycles (self)

Get the number of relevant cycles. raises RDLError if calculation wasn't successful returns number of RCs

9.1.3.14 def py_rdl.Calculator.Calculator.get_nof_relevant_cycles_for_urf (self, urf_index)

Get the number of relevant cycles for given URF. raises RDLError if calculation wasn't successful returns number of RCs $\,$

9.1.3.15 def py_rdl.Calculator.Calculator.get_nof_ringsystems (self)

Get the number of Ringsystems. A ring system is a 2-connected component of the graph. raises RDLError if calculation wasn't successful

9.1.3.16 def py_rdl.Calculator.Calculator.get_nof_urf (self)

Get the number of URFs. raises RDLError if calculation wasn't successful

9.1.3.17 def py_rdl.Calculator.Calculator.get_rcfs_for_edge (self, edge)

Get the RCFs this edge is part of. raises RDLError if calculation wasn't successful returns list of RCFs $\,$

9.1.3.18 def py_rdl.Calculator.Calculator.get_rcfs_for_node (self, node)

Get the RCFs this node is part of. raises RDLError if calculation wasn't successful returns a list of RCFs

9.1.3.19 def py_rdl.Calculator.Calculator.get_relevant_cycle_prototypes (self)

Get relevant cycle prototypes (one for each RCF). raises RDLError if calculation wasn't successful returns list of relevant cycle prototypes

9.1.3.20 def py_rdl.Calculator.Calculator.get_relevant_cycles (self)

Get relevant cycles.
raises RDLError if calculation wasn't successful
returns generator for enumerating relevant cycles

9.1.3.21 def py_rdl.Calculator.Calculator.get_relevant_cycles_for_rcf (self, rcf_index)

Get the cycles in this of RCF. raises RDLError if calculation wasn't successful returns generator for enumerating relevant cycles in this RCF

9.1.3.22 def py_rdl.Calculator.Calculator.get_relevant_cycles_for_urf (self, urf_index)

Get the cycles in this of URF. raises RDLError if calculation wasn't successful returns generator for enumerating relevant cycles in this URF

9.1.3.23 def py_rdl.Calculator.Calculator.get_sssr (self)

Get a minimal cycle base. raises RDLError if calculation wasn't successful returns list of cycles

9.1.3.24 def py_rdl.Calculator.Calculator.get_urfs_for_edge (self, edge)

Get the URFs this edge is part of. raises RDLError if calculation wasn't successful returns list of URFs

9.1.3.25 def py_rdl.Calculator.Calculator.get_urfs_for_node (self, node)

Get the URFs this node is part of. raises RDLError if calculation wasn't successful returns a list of URFs

9.1.3.26 def py_rdl.Calculator.Calculator.get_weight_for_rcf (self, rcf_index)

Get weight of this RCF. raises RDLError if calculation wasn't successful return weight for given RCF $\,$

9.1.3.27 def py_rdl.Calculator.Calculator.get_weight_for_urf (self, urf_index)

Get weight of this URF. raises RDLError if calculation wasn't successful return weight for given URF

26 Class Documentation

9.1.3.28 def py_rdl.Calculator.Calculator.is_calculated (self)

```
Check calculation status. returns True if calculation was successful, False otherwise.
```

9.1.3.29 def py_rdl.Calculator.Calculator.set_graph (self, graph)

```
Set the graph data structure.

This has to be a Graph object. See Graph for a generic interface.
```

9.1.4 Member Data Documentation

9.1.4.1 py_rdl.Calculator.Calculator.rcfs

List of calculated RCFs.

You can access individual RCFs using this list and use the RCF objects as index to all functions below, that take an RCF as argument.

9.1.4.2 py_rdl.Calculator.Calculator.ringsystems

List of calculated ring systems.

You can access individual ring systems using this list and use the Ringsystem objects as index to all functions below, that take an ring system as argument.

9.1.4.3 py_rdl.Calculator.Calculator.urfs

List of calculated URFs.

You can access individual URFs using this list and use the URF objects as index to all functions below, that take an URF as argument.

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

• src/python/py_rdl/Calculator.py

9.2 py_rdl.wrapper.Cycle.Cycle Class Reference

Public Member Functions

```
def __init__
def weight
Get the weight of the cycle.
```

```
    def __str__
```

def __repr__

Public Attributes

• edges

set of edges

nodes

set of nodes

• urf

URF this cycle belongs to.

rcf

RCF this cycle belongs to.

9.2.1 Detailed Description

This class represents an cycle by it's edges, nodes, and the urf and rcf it is part of.

9.2.2 Constructor & Destructor Documentation

9.2.2.1 def py_rdl.wrapper.Cycle.Cycle.__init__ (self, edges, nodes, urf, rcf)

Initialize a cycle with edges, nodes and ring families.

9.2.3 Member Function Documentation

9.2.3.1 def py_rdl.wrapper.Cycle.Cycle.weight (self)

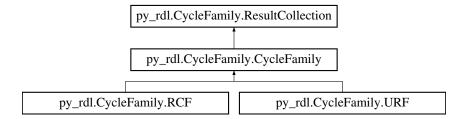
Get the weight of the cycle.

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

• src/python/py_rdl/wrapper/Cycle.py

9.3 py_rdl.CycleFamily.CycleFamily Class Reference

Inheritance diagram for py_rdl.CycleFamily.CycleFamily:



Public Member Functions

def __init__

Public Attributes

· weight

weight of the cycle family

28 Class Documentation

9.3.1 Detailed Description

```
Object for representing a cycle family.
```

9.3.2 Constructor & Destructor Documentation

9.3.2.1 def py_rdl.CycleFamily.CycleFamily.__init__ (self, index, nodes, edges, weight)

```
index: index of result
nodes: list or set of nodes
edges: list or set of edges
weight: weight of the cycles in the family
```

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

src/python/py_rdl/CycleFamily.py

9.4 py_rdl.Graph.Graph Class Reference

Public Member Functions

- def init
- · def from_edges
- · def add edge
- · def get edge for indices
- def get_indices_for_edge
- def get_node_for_index
- def get_index_for_node
- def get_nof_nodes
- def get_edges

9.4.1 Detailed Description

```
Object for holding a graph for calculations.
```

9.4.2 Constructor & Destructor Documentation

9.4.2.1 def py_rdl.Graph.Graph.__init__ (self)

```
Create a new (empty) graph.

Use this method and add individual edges or use from_edges instead.
```

9.4.3 Member Function Documentation

```
9.4.3.1 def py_rdl.Graph.Graph.add_edge ( self, edge, get\_node\_1 = operator.itemgetter(0), <math>get\_node\_2 = operator.itemgetter(1), get\_node\_id = lambda x: x, get\_edge\_id = lambda x: x)
```

```
Add an edge to this graph.
edge: edge to be added to the graph
get_node_1: function retrieving the first node of an edge
get_node_2: function retrieving the second node of an edge
get_node_id: function for retrieving node identifier
get_edge_id: function for retrieving edge identifier
```

```
returns the edge_id if successfull, None if edge already present
(and warns in that case)
This function adds an edge to the graph.
DO NOT USE AFTER CALCULATION
9.4.3.2 def py_rdl.Graph.Graph.from_edges ( cls, edge_iterable, get_node_1 = operator.itemgetter(0),
      get_node_2 = operator.itemgetter(1), get_node_id = lambda x: x, get_edge_id =
      lambda x: x)
Create a new Graph datastructure
edge_iterable: an iterable containing the edges
get_node_1: function retrieving the first node of an edge
get_node_2: function retrieving the second node of an edge
get_node_id: function for retrieven node identifier
get_edge_id: function for retrieving edge identifier
This classmethod creates a new Graph.
The edge_iterable can be any iterable (list, generator etc.).
An edge can be as simple as a pair of nodes.
```

9.4.3.3 def py_rdl.Graph.Graph.get_edge_for_indices (self, node_index1, node_index2)

The get_node_ function must return the respective nodes adjacent to an edge (default: first and second element of the edge). The node must be hashable. If it's not, use some unique ID instead.

The edge must be hashable. If it's not, use some unique ID

Returns the edge which is formed by the two nodes (as indices).
node_index1: index of the first node
node_index2: index of the second node
returns edge adjacent by two nodes
raises KeyError if the edge does not exist

This function is used internally by Calculator, you probably won't need this function.

9.4.3.4 def py_rdl.Graph.Graph.get_edges (self)

instead (provided by get_edge_id).

Get the edges in the graph.

This function is used internally by Calculator, you won't need this function.

9.4.3.5 def py_rdl.Graph.Graph.get_index_for_node (self, node)

Returns the internal index for the node. node: node object returns the index for the node object raises KeyError if node does not exist

This function is used internally by Calculator, you probably won't need this function.

9.4.3.6 def py_rdl.Graph.Graph.get_indices_for_edge (self, edge)

Returns the edge which is formed by the two nodes $% \left(1\right) =\left(1\right) \left(1\right) \left($

30 Class Documentation

```
(as indices).
edge: edge
returns edge adjacent by two nodes
raises KeyError if the edge does not exist
This function is used internally by Calculator, you
probably won't need this function.
```

9.4.3.7 def py_rdl.Graph.Graph.get_node_for_index (self, node_index)

```
Returns the node with the internal index. node_index: index of the node returns the node object with this index raises IndexError if node does not exist
```

This function is used internally by Calculator, you probably won't need this function.

9.4.3.8 def py_rdl.Graph.Graph.get_nof_nodes (self)

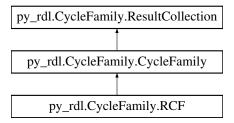
Get the number of nodes in the graph.

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

· src/python/py_rdl/Graph.py

9.5 py_rdl.CycleFamily.RCF Class Reference

Inheritance diagram for py_rdl.CycleFamily.RCF:



Public Member Functions

- def __str__
- def __init__

Additional Inherited Members

9.5.1 Detailed Description

Object for representing a relevant cycle family.

9.5.2 Constructor & Destructor Documentation

9.5.2.1 def py_rdl.CycleFamily.RCF.__init__ (self, index, nodes, edges, weight)

```
index: index of result
nodes: list or set of nodes
edges: list or set of edges
weight: weight of the cycles in the family
```

9.5.3 Member Function Documentation

```
9.5.3.1 def py_rdl.CycleFamily.RCF.__str__( self )
to string.
```

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

• src/python/py_rdl/CycleFamily.py

9.6 RDL_cycle Struct Reference

```
#include <RingDecomposerLib.h>
```

Public Attributes

- RDL edge * edges
- · unsigned weight
- · unsigned urf
- · unsigned rcf

9.6.1 Detailed Description

This structure holds a cycle (or ring) in the graph. It is essentially an array of edges which are represented by pairs of nodes.

9.6.2 Member Data Documentation

```
9.6.2.1 \quad \textbf{RDL\_edge} * \textbf{RDL\_cycle} :: \textbf{edges}
```

array of RDL_edge in the cycle

9.6.2.2 unsigned RDL_cycle::rcf

RCF of the cycle

9.6.2.3 unsigned RDL_cycle::urf

URF of the cycle

32 Class Documentation

9.6.2.4 unsigned RDL_cycle::weight

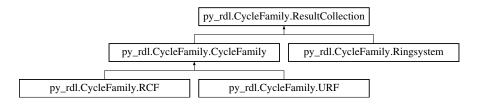
weight of the cycle (length of the array)

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

• src/RingDecomposerLib/RingDecomposerLib.h

9.7 py_rdl.CycleFamily.ResultCollection Class Reference

Inheritance diagram for py_rdl.CycleFamily.ResultCollection:



Public Member Functions

- def __init__
- def str
- def __repr__

Public Attributes

• index

index of the result

• nodes

set of nodes of the result

· edges

set of edges of the result

9.7.1 Detailed Description

```
Object for representing a result.
```

9.7.2 Constructor & Destructor Documentation

$9.7.2.1 \quad def \ py_rdl. Cycle Family. Result Collection. _init_ (\ \textit{self, index, nodes, edges} \)$

```
index: index of result
nodes: list or set of nodes
edges: list or set of edges
```

9.7.3 Member Function Documentation

9.7.3.1 def py_rdl.CycleFamily.ResultCollection.__repr__ (self)

Representation

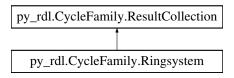
```
9.7.3.2 def py_rdl.CycleFamily.ResultCollection.__str__( self )
to string.
```

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

• src/python/py_rdl/CycleFamily.py

9.8 py_rdl.CycleFamily.Ringsystem Class Reference

Inheritance diagram for py_rdl.CycleFamily.Ringsystem:



Public Member Functions

def __init__

Additional Inherited Members

9.8.1 Detailed Description

```
Object for representing a ringsystem.

A ring system is a 2-connected component of the graph.
```

9.8.2 Constructor & Destructor Documentation

9.8.2.1 def py_rdl.CycleFamily.Ringsystem.__init__ (self, index, nodes, edges)

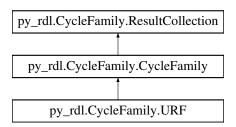
```
index: index of result
nodes: list or set of nodes
edges: list or set of edges
```

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

src/python/py_rdl/CycleFamily.py

9.9 py_rdl.CycleFamily.URF Class Reference

Inheritance diagram for py_rdl.CycleFamily.URF:



34 Class Documentation

Public Member Functions

```
def __str__def __init__
```

Additional Inherited Members

9.9.1 Detailed Description

```
Object for representing a unique ring family.
```

9.9.2 Constructor & Destructor Documentation

```
9.9.2.1 def py_rdl.CycleFamily.URF.__init__ ( self, index, nodes, edges, weight )
```

```
index: index of result
nodes: list or set of nodes
edges: list or set of edges
weight: weight of the cycles in the family
```

9.9.3 Member Function Documentation

```
9.9.3.1 def py_rdl.CycleFamily.URF.__str__( self )
to string.
```

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

• src/python/py_rdl/CycleFamily.py

Chapter 10

File Documentation

10.1 src/MinimalExample/MinimalExample.c File Reference

Minimal example for the RingDecomposerLib library.

```
#include <assert.h>
#include <stdio.h>
#include "RingDecomposerLib.h"
```

10.1.1 Detailed Description

Minimal example for the RingDecomposerLib library.

10.2 src/RingDecomposerLib/RingDecomposerLib.h File Reference

This file contains the API of the RingDecomposerLib library.

Classes

• struct RDL_cycle

Typedefs

```
    typedef struct RDL_data RDL_data
```

Structure representing a calculation result.

• typedef struct RDL_graph RDL_graph

Datastructure representing a graph for calculations.

typedef unsigned RDL_node

A node is represented by its index in the graph (0 to |V|-1).

typedef RDL_node RDL_edge [2]

An edge is represented by an array of size two containing the adjacent nodes.

• typedef enum RDL_ERROR_LEVEL RDL_ERROR_LEVEL

error levels for custom logging functions

- typedef void(* RDL_outputFunction)(RDL_ERROR_LEVEL level, const char *m,...)
- typedef struct RDL cycle RDL cycle
- typedef struct RDL_cycleIterator RDL_cycleIterator

Iterator for relevant cycles.

Enumerations

enum RDL_ERROR_LEVEL { RDL_DEBUG, RDL_WARNING, RDL_ERROR, RDL_INITIALIZE_OUTPUT
 }

error levels for custom logging functions

Functions

RDL API void RDL setOutputFunction (RDL outputFunction func)

Set the output function for warning and error messages.

• RDL_API void RDL_writeToStderr (RDL_ERROR_LEVEL level, const char *fmt,...)

An output function for writing everything to stderr.

RDL_API void RDL_writeNothing (RDL_ERROR_LEVEL level, const char *fmt,...)

No output function.

• RDL API RDL graph * RDL initNewGraph (unsigned nof nodes)

Initializes a new RDL_graph.

• RDL_API void RDL_deleteGraph (RDL_graph *graph)

Delete RDL graph.

• RDL API unsigned RDL addUEdge (RDL graph *graph, RDL node node1, RDL node node2)

Adds an undirected edge to the graph.

RDL_API RDL_data * RDL_calculate (RDL_graph *input_graph)

Calculates the RDL data structure of the given graph and returns it.

• RDL_API void RDL_deleteData (RDL_data *data)

Deletes RDL_data from memory, including the RDL_graph.

• RDL_API unsigned RDL_getNofURF (const RDL_data *data)

Returns the number of URFs.

• RDL_API unsigned RDL_getNofRCF (const RDL_data *data)

Returns the number of RCFs.

• RDL_API unsigned RDL_getWeightForURF (const RDL_data *data, unsigned index)

Returns the weight of each cycle in the URF identified by its index.

• RDL_API unsigned RDL_getWeightForRCF (const RDL_data *data, unsigned index)

Returns the weight of each cycle in the RCF identified by its index.

RDL_API unsigned RDL_getNodesForURF (const RDL_data *data, unsigned index, RDL_node **RDL_node array_ptr)

Gives the nodes of an URF identified with its index in an array of RDL_node.

RDL_API unsigned RDL_getEdgesForURF (const RDL_data *data, unsigned index, RDL_edge **RDL_edge array_ptr)

Gives the edges of an URF identified with its index.

RDL_API unsigned RDL_getNodesForRCF (const RDL_data *data, unsigned index, RDL_node **RDL_node array ptr)

Gives the nodes of an RCF identified with its index in an array of RDL_node.

RDL_API unsigned RDL_getEdgesForRCF (const RDL_data *data, unsigned index, RDL_edge **RDL_edge array_ptr)

Gives the edges of an RCF identified with its index.

• RDL_API unsigned RDL_getNofURFContainingNode (const RDL_data *data, RDL_node node)

Returns the number of URFs that contain the given node.

RDL_API unsigned RDL_getNofURFContainingEdge (const RDL_data *data, RDL_node node1, RDL_node node2)

Returns the number of URFs that contain the RDL_edge defined by the two given nodes.

• RDL_API unsigned RDL_getNofRCFContainingNode (const RDL_data *data, RDL_node node)

Returns the number of RCFs that contain the given node.

RDL_API unsigned RDL_getNofRCFContainingEdge (const RDL_data *data, RDL_node node1, RDL_node node2)

Returns the number of RCFs that contain the RDL_edge defined by the two given nodes.

RDL_API unsigned RDL_getURFsContainingNode (const RDL_data *data, RDL_node node, unsigned **R-DL ids ptr)

Gives all URFs containing the node.

RDL_API unsigned RDL_getURFsContainingEdge (const RDL_data *data, RDL_node node1, RDL_node node2, unsigned **RDL_ids_ptr)

Gives all URFs containing the edge.

RDL_API unsigned RDL_getRCFsContainingNode (const RDL_data *data, RDL_node node, unsigned **R-DL ids ptr)

Gives all RCFs containing the node.

RDL_API unsigned RDL_getRCFsContainingEdge (const RDL_data *data, RDL_node node1, RDL_node node2, unsigned **RDL_ids_ptr)

Gives all RCFs containing the edge.

• RDL API void RDL deleteCycle (RDL cycle *cycle)

Free memory of RDL_cycle.

RDL API RDL cycleIterator * RDL cycleIteratorNext (RDL cycleIterator *it)

Advance the cycle iterator by one.

RDL_API RDL_cycle * RDL_cycleIteratorGetCycle (RDL_cycleIterator *it)

Get the cycle as RDL_cycle.

RDL_API int RDL_cycleIteratorAtEnd (RDL_cycleIterator *it)

Check if iterator is at end (invalid)

RDL_API void RDL_deleteCycleIterator (RDL_cycleIterator *it)

Free memory of the cycle iterator.

RDL_API unsigned RDL_getRCyclesForURF (const RDL_data *data, unsigned index, RDL_cycle ***RDL_cycle_array_ptr)

Gives all relevant cycles of the URF with the given index.

• RDL_API RDL_cycleIterator * RDL_getRCyclesForURFIterator (const RDL_data *data, unsigned index)

Get iterator for all relevant cycles of the URF with the given index.

RDL_API unsigned RDL_getRCyclesForRCF (const RDL_data *data, unsigned index, RDL_cycle ***RDL_cycle_array_ptr)

Gives all relevant cycles of the RCF with the given index.

• RDL_API RDL_cycleIterator * RDL_getRCyclesForRCFIterator (const RDL_data *data, unsigned index)

Get iterator for all relevant cycles of the RCF with the given index.

• RDL_API unsigned RDL_getRCycles (const RDL_data *data, RDL_cycle ***RDL_cycle_array_ptr)

Gives a list of all relevant cycles.

RDL_API RDL_cycleIterator * RDL_getRCyclesIterator (const RDL_data *data)

Get iterator for all relevant cycles.

• RDL_API double RDL_getNofRCForURF (const RDL_data *data, unsigned index)

Gives the number of relevant cycles in this URF.

• RDL API double RDL getNofRCForRCF (const RDL data *data, unsigned index)

Gives the number of relevant cycles in this RCF.

RDL_API double RDL_getNofRC (const RDL_data *data)

Gives the number of relevant cycles.

RDL_API unsigned RDL_getSSSR (const RDL_data *data, RDL_cycle ***RDL_cycle_array_ptr)
 Gives a set of cycles that forms a Minimal Cycle Basis of the graph.

• RDL_API unsigned RDL_getRCPrototypes (const RDL_data *data, RDL_cycle ***RDL_cycle_array_ptr)

Gives a list of relevant cycle prototypes (one for each RCF).

• RDL API void RDL deleteCycles (RDL cycle **cycles, unsigned number)

Deallocates the structure given by RDL_getRCycles(), RDL_getSSSR(), RDL_getRCPrototypes() and RDL_getR-CyclesForURF(), if called on its result and return value (the number of cycles)

RDL_API unsigned RDL_translateCycArray (const RDL_data *data, RDL_cycle **old_array, unsigned number, char ***RDL_cycle_array_ptr)

Translates the results of RDL_getRCycles(), RDL_getSSSR(), RDL_getRCPrototypes() and RDL_getRCyclesForU-RF() (arrays of RDL_cycle) into an array of cycles represented by arrays of $\{0, 1\}^{\wedge}|E|$ (bitsets).

RDL API void RDL deleteEdgeIdxArray (char **cycles, unsigned number)

Deallocates the structure given by RDL_translateCycArray(), if called on its result and return value (the number of cycles).

- RDL_API unsigned RDL_getEdgeArray (const RDL_data *data, RDL_edge **RDL_edge_array_ptr)
 Gives the edges of the graph.
- RDL_API unsigned RDL_getNofEdges (const RDL_data *data)

Get the number of edges in the graph.

RDL_API unsigned RDL_getEdgeId (const RDL_data *data, unsigned from, unsigned to)
 Get the id of the edge.

• RDL_API unsigned RDL_getNofRingsystems (const RDL_data *data)

Get the number of ring systems in the graph.

 $\bullet \ \ RDL_API \ unsigned \ RDL_getNofNodesForRingsystem \ (const \ RDL_data \ *data, \ unsigned \ idx)$

Get the number of nodes in the ring system.

• RDL_API unsigned RDL_getNofEdgesForRingsystem (const RDL_data *data, unsigned idx)

Get the number of edges in the ring system.

RDL_API unsigned RDL_getEdgesForRingsystem (const RDL_data *data, unsigned idx, RDL_edge **edges)

Get the edges in the ring system.

RDL_API unsigned RDL_getNodesForRingsystem (const RDL_data *data, unsigned idx, RDL_node **nodes)

Get the nodes in the ring system.

• RDL_API unsigned RDL_getRingsystemForEdge (const RDL_data *data, unsigned from, unsigned to) Get the ring system id for given edge.

Variables

• RDL_outputFunction RDL_outputFunc

the output function for warnings and errors

• const unsigned RDL_INVALID_RESULT

Invalid result indicator.

const unsigned RDL_DUPLICATE_EDGE

Duplicate edge indicator.

const unsigned RDL NO RINGSYSTEM

No ringsystem indicator.

const double RDL_INVALID_RC_COUNT

Invalid number of RCs.

10.2.1 Detailed Description

This file contains the API of the RingDecomposerLib library.

10.2.2 Typedef Documentation

10.2.2.1 typedef struct RDL cycle RDL cycle

This structure holds a cycle (or ring) in the graph. It is essentially an array of edges which are represented by pairs of nodes.

10.2.2.2 typedef struct RDL_cycleIterator RDL_cycleIterator

Iterator for relevant cycles.

Always check if the iterator is at end BEFORE incrementing or accessing it.

Example usage:

```
RDL_cycle *cycle;
RDL_cycleIterator *it = RDL_getRCyclesIterator(data);
while(!RDL_cycleIteratorAtEnd(it)) {
    cycle = RDL_cycleIteratorGetCycle(it);
    <do something with the cycle>
    RDL_deleteCycle(cycle);
    RDL_cycleIteratorNext(it);
}
RDL_deleteCycleIterator(it);
```

10.2.2.3 typedef struct RDL data RDL data

Structure representing a calculation result.

This is the central structure, that is used to store the calculation results. Almost all functions in this header work on it.

10.2.2.4 typedef struct RDL_graph RDL_graph

Datastructure representing a graph for calculations.

Build graph data structure that is used for the calculation. The graph is stored as an adjacency list. The vertices are numbered from 0 to |V|-1. Call the following functions:

RDL_initNewGraph(unsigned V) to initialize a new graph with V vertices (returns RDL_graph *)

RDL_addUEdge(RDL_graph *, RDL_node from, RDL_node to) to add a new (undirected) edge from the vertex with index "from" to the vertex with index "to".

Then RDL calculate can be called on it.

10.2.3 Function Documentation

```
10.2.3.1 RDL_API unsigned RDL_addUEdge ( RDL_graph * graph, RDL_node node1, RDL_node node2 )
```

Adds an undirected edge to the graph.

Parameters

| graph | pointer to the RDL_graph that the edge will to be added to |
|-------------|--|
| node1,node2 | pair of RDL_node which the edge is going to connect |

Returns

internal index of edge if successful, RDL_INVALID_RESULT on failures (invalid index, loop), RDL_DUPLICA-TE_EDGE if the edge was already present

```
10.2.3.2 RDL_API RDL_data* RDL_calculate ( RDL_graph * input_graph )
```

Calculates the RDL data structure of the given graph and returns it.

Parameters

input_graph The graph for calculation.

Returns

pointer to the RDL_data containing the calculation result, NULL on failure

Note

The RDL_graph has to be an undirected graph.

Takes ownership of the RDL_graph, if calculation is successful!

10.2.3.3 RDL_API int RDL_cycleIteratorAtEnd (RDL_cycleIterator * it)

Check if iterator is at end (invalid)

Parameters

it RDL_cycleIterator

Returns

0 if not at end, a non-zero value otherwise

10.2.3.4 RDL_API RDL_cycle* RDL_cycleIteratorGetCycle (RDL_cycleIterator * it)

Get the cycle as RDL_cycle.

Parameters

it RDL_cycleIterator

Returns

RDL_cycle holding the cycle, NULL on error

Note

The RDL_cycle is constructed and returned as a pointer. You have to delete it using RDL_deleteCycle. DO NOT call without checking RDL_cycleIteratorAtEnd

10.2.3.5 RDL_API RDL_cycleIterator* RDL_cycleIteratorNext (RDL_cycleIterator* it)

Advance the cycle iterator by one.

Parameters

it RDL_cycleIterator

Returns

returns the iterator it itself, if successful; NULL on error

Note

DO NOT call without checking RDL_cycleIteratorAtEnd

10.2.3.6 RDL_API void RDL_deleteCycle (RDL_cycle * cycle)

Free memory of RDL_cycle.

Parameters

| cycle | The RDL_cycle to delete. |
|-------|--------------------------|
|-------|--------------------------|

10.2.3.7 RDL_API void RDL_deleteCycleIterator (RDL_cycleIterator * it)

Free memory of the cycle iterator.

Parameters

| it | RDL_cycleIterator |
|----|-------------------|

10.2.3.8 RDL_API void RDL_deleteCycles (RDL_cycle ** cycles, unsigned number)

Deallocates the structure given by RDL_getRCycles(), RDL_getSSSR(), RDL_getRCPrototypes() and RDL_getR-CyclesForURF(), if called on its result and return value (the number of cycles)

Parameters

| cycles | The array of cycles. |
|--------|-----------------------------------|
| number | The number of cycles in the array |

10.2.3.9 RDL_API void RDL_deleteData (RDL_data * data)

Deletes RDL_data from memory, including the RDL_graph.

Parameters

| _ | | |
|---|------|---|
| | data | pointer to the RDL_data that is going to be deleted |

Note

also deletes the associated RDL graph

10.2.3.10 RDL_API void RDL_deleteEdgeldxArray (char ** cycles, unsigned number)

Deallocates the structure given by RDL_translateCycArray(), if called on its result and return value (the number of cycles).

Parameters

| cycles | The translated array of cycles. |
|--------|------------------------------------|
| number | The number of cycles in the array. |

10.2.3.11 RDL_API void RDL_deleteGraph (RDL_graph * graph)

Delete RDL_graph.

Parameters

| graph | pointer to RDL_graph to delete |
|-------|--------------------------------|

Note

You don't have to delete the graph after successful calculation!

10.2.3.12 RDL_API unsigned RDL_getEdgeArray (const RDL_data * data, RDL_edge ** RDL_edge_array_ptr)

Gives the edges of the graph.

Returns

the number of edges in the graph, RDL_INVALID_RESULT on error

Parameters

| data | pointer to the RDL_data holding the calculation results. |
|-----------|--|
| RDL_edge | pointer that points to the result array (declare RDL_edge * and give address as parameter) |
| array_ptr | |

Note

Result has to be deallocated using free (*RDL_edge_array_ptr)

10.2.3.13 RDL API unsigned RDL getEdgeld (const RDL data * data, unsigned from, unsigned to)

Get the id of the edge.

Parameters

| data | pointer to the RDL_data holding the calculation results. |
|------|--|
| from | RDL_node starting at this edge |
| to | RDL_node starting at this edge |

Returns

edge id, RDL_INVALID_RESULT, if not present

Note

As the graph is stored as an adjacency list, this function has linear runtime in the node degree.

10.2.3.14 RDL_API unsigned RDL_getEdgesForRCF (const RDL_data * data, unsigned index, RDL_edge ** RDL_edge_array_ptr)

Gives the edges of an RCF identified with its index.

Returns

the number of edges in this RCF, RDL_INVALID_RESULT on failure

Parameters

| data | pointer to the RDL_data holding the calculation results. |
|-----------|--|
| index | the index of the RCF |
| RDL_edge | pointer that points to the result array (declare RDL_edge * and give address as parameter) |
| array_ptr | |

Note

Result has to be deallocated using free (*RDL_edge_array_ptr) Gives an array of edges where an edge is represented by two RDL_node that it connects.

10.2.3.15 RDL_API unsigned RDL_getEdgesForRingsystem (const RDL_data * data, unsigned idx, RDL_edge ** edges)

Get the edges in the ring system.

Parameters

| data | pointer to the RDL_data holding the calculation results. |
|-------|--|
| idx | index of the ring system |
| edges | pointer that points to the result array (declare RDL_edge * and give address as parameter) |

Note

Result has to be deallocated using free (*edges)

Returns

number of edges in the ring system, RDL_INVALID_RESULT on error

Note

A ring system is a 2-connected component of the graph.

10.2.3.16 RDL_API unsigned RDL_getEdgesForURF (const RDL_data * data, unsigned index, RDL_edge ** RDL_edge_array_ptr)

Gives the edges of an URF identified with its index.

Returns

the number of edges in this URF, RDL_INVALID_RESULT on failure

Parameters

| data | pointer to the RDL_data holding the calculation results. |
|-----------|--|
| index | the index of the URF |
| RDL_edge | pointer that points to the result array (declare RDL_edge * and give address as parameter) |
| array_ptr | |

Note

Result has to be deallocated using free (*RDL_edge_array_ptr) Gives an array of edges where an edge is represented by two RDL_node that it connects.

10.2.3.17 RDL_API unsigned RDL_getNodesForRCF (const RDL_data * data, unsigned index, RDL_node ** RDL_node_array_ptr)

Gives the nodes of an RCF identified with its index in an array of RDL_node.

Returns

the number of nodes in the RCF, RDL_INVALID_RESULT on failure

Parameters

| data | pointer to the RDL_data holding the calculation results. |
|------|--|

| index | the index of the RCF |
|-----------|--|
| RDL_node | pointer that points to the result array (declare RDL_node * and give address as parameter) |
| array_ptr | |

Note

result has to be deallocated using free (*RDL_node_array_ptr).

10.2.3.18 RDL_API unsigned RDL_getNodesForRingsystem (const RDL_data * data, unsigned idx, RDL_node ** nodes)

Get the nodes in the ring system.

Parameters

| data | pointer to the RDL_data holding the calculation results. |
|-------|--|
| idx | index of the ring system |
| nodes | pointer that points to the result array (declare RDL_edge * and give address as parameter) |

Note

Result has to be deallocated using free (*nodes)

Returns

number of nodes in the ring system, RDL_INVALID_RESULT on error

Note

A ring system is a 2-connected component of the graph.

10.2.3.19 RDL_API unsigned RDL_getNodesForURF (const RDL_data * data, unsigned index, RDL_node ** RDL_node_array_ptr)

Gives the nodes of an URF identified with its index in an array of RDL_node.

Returns

the number of nodes in the URF, RDL INVALID RESULT on failure

Parameters

| data | pointer to the RDL data holding the calculation results. |
|------------|--|
| index | the index of the URF |
| RDL node - | pointer that points to the result array (declare RDL_node * and give address as parameter) |
| array ptr | |

Note

result has to be deallocated using free (*RDL_node_array_ptr).

10.2.3.20 RDL_API unsigned RDL_getNofEdges (const RDL_data * data)

Get the number of edges in the graph.

Parameters

| data | pointer to the RDL_data holding the calculation results. |
|------|--|
|------|--|

Returns

number of edges in the graph, RDL_INVALID_RESULT on error

10.2.3.21 RDL_API unsigned RDL_getNofEdgesForRingsystem (const RDL_data * data, unsigned idx)

Get the number of edges in the ring system.

Parameters

| data | pointer to the RDL_data holding the calculation results. |
|------|--|
| idx | index of the ring system |

Returns

number of edges in the ring system, RDL_INVALID_RESULT on error

Note

A ring system is a 2-connected component of the graph.

10.2.3.22 RDL_API unsigned RDL_getNofNodesForRingsystem (const RDL_data * data, unsigned idx)

Get the number of nodes in the ring system.

Parameters

| data | pointer to the RDL_data holding the calculation results. |
|------|--|
| idx | index of the ring system |

Returns

number of nodes in the ring system, RDL_INVALID_RESULT on error

Note

A ring system is a 2-connected component of the graph.

10.2.3.23 RDL_API double RDL_getNofRC (const RDL_data * data)

Gives the number of relevant cycles.

Parameters

| data | pointer to the RDL_data holding the calculation results |
|------|---|

Returns

the number of relevant cycles RDL_INVALID_RC_COUNT if the number is too large to calculate or on failure

10.2.3.24 RDL_API unsigned RDL_getNofRCF (const RDL_data * data)

Returns the number of RCFs.

Parameters

| data | Pointer to the RDL_data of which the number of RCF is requested |
|------|---|
|------|---|

Returns

The number of RCFs, RDL_INVALID_RESULT on failure

10.2.3.25 RDL_API unsigned RDL_getNofRCFContainingEdge (const RDL_data * data, RDL_node node1, RDL_node node2)

Returns the number of RCFs that contain the RDL_edge defined by the two given nodes.

Parameters

| data | Pointer to the RDL_data storing the calculation results. |
|-------------|--|
| node1,node2 | pair of RDL_node connected by the edge |

Returns

number of RCFs containing the edge, RDL_INVALID_RESULT on failure

Note

This functions internally enumerates all edges for each RCF. For repeated checks with different RDL_node it is recommended to enumerate them with RDL_getEdgesForRCF and check against this array.

10.2.3.26 RDL_API unsigned RDL_getNofRCFContainingNode (const RDL_data * data, RDL_node node)

Returns the number of RCFs that contain the given node.

Parameters

| data | pointer to the RDL_data holding the calculation results. |
|------|--|
| node | the RDL_node to look for in the RCFs |

Returns

number of RCFs containing the node, RDL_INVALID_RESULT on failure

Note

This functions internally enumerates all nodes for each RCF. For repeated checks with different RDL_node it is recommended to enumerate them with RDL_getNodesForRCF and check against this array.

10.2.3.27 RDL_API double RDL_getNofRCForRCF (const RDL_data * data, unsigned index)

Gives the number of relevant cycles in this RCF.

Parameters

| data | pointer to the RDL_data holding the calculation results. |
|-------|--|
| index | id of the RCF |

Returns

the number of relevant cycles in the RCF or RDL_INVALID_RC_COUNT if the number is too large to calculate or on failure

10.2.3.28 RDL_API double RDL_getNofRCForURF (const RDL_data * data, unsigned index)

Gives the number of relevant cycles in this URF.

Parameters

| data | pointer to the RDL_data holding the calculation results |
|-------|---|
| index | the index of the URF |

Returns

the number of relevant cycles in the URF or RDL_INVALID_RC_COUNT if the number is too large to calculate or on failure

10.2.3.29 RDL_API unsigned RDL_getNofRingsystems (const RDL_data * data)

Get the number of ring systems in the graph.

Parameters

| data | pointer to the RDL_data holding the calculation results. |
|------|--|

Returns

number of ring systems in the graph, RDL INVALID RESULT on error

10.2.3.30 RDL_API unsigned RDL_getNofURF (const RDL_data * data)

Returns the number of URFs.

Parameters

| data Folitier to the HDL_data of which the humber of Ohi- is requested | data | Pointer to the RDL_data of which the number of URF is requested |
|--|------|---|
|--|------|---|

Returns

The number of URFs, RDL_INVALID_RESULT on failure

10.2.3.31 RDL_API unsigned RDL_getNofURFContainingEdge (const RDL_data * data, RDL_node node1, RDL_node node2)

Returns the number of URFs that contain the RDL_edge defined by the two given nodes.

Parameters

| data | Pointer to the RDL_data storing the calculation results. |
|-------------|--|
| node1,node2 | pair of RDL_node connected by the edge |

Returns

number of URFs containing the edge, RDL_INVALID_RESULT on failure

Note

This functions internally enumerates all edges for each URF. For repeated checks with different RDL_node it is recommended to enumerate them with RDL_getEdgesForURF and check against this array.

10.2.3.32 RDL_API unsigned RDL_getNofURFContainingNode (const RDL data * data, RDL node node)

Returns the number of URFs that contain the given node.

Parameters

| data | pointer to the RDL_data holding the calculation results. |
|------|--|
| node | the RDL_node to look for in the URFs |

Returns

number of URFs containing the node, RDL_INVALID_RESULT on failure

Note

This functions internally enumerates all nodes for each URF. For repeated checks with different RDL_node it is recommended to enumerate them with RDL_getNodesForURF and check against this array.

10.2.3.33 RDL_API unsigned RDL_getRCFsContainingEdge (const RDL_data * data, RDL_node node1, RDL_node node2, unsigned ** RDL_ids_ptr)

Gives all RCFs containing the edge.

Returns

the number of RCFs containing the edge, RDL_INVALID_RESULT on failure

Parameters

| data | pointer to the RDL_data holding the calculation results |
|-------------|---|
| node1 | the first RDL_node of the edge |
| node2 | the the second RDL_node of the edge |
| RDL_ids_ptr | pointer that points to the result array of integers containing all indices of RCFs containing the |
| | edge. (declare unsigned * and give address as parameter) |

Note

The array RDL_ids_ptr has to be to be deallocated with free (*RDL_ids_ptr)

This functions internally enumerates all edges for each RCF. For repeated queries with different RDL_node it is recommended to enumerate them with RDL_getEdgesForRCF once and use this array.

10.2.3.34 RDL_API unsigned RDL_getRCFsContainingNode (const RDL_data * data, RDL_node node, unsigned ** RDL_ids_ptr)

Gives all RCFs containing the node.

Returns

the number of RCFs containing the node, RDL_INVALID_RESULT on failure

Parameters

| data | pointer to the RDL_data holding the calculation results |
|-------------|---|
| node | the RDL_node |
| RDL_ids_ptr | pointer that points to the result array of integers containing all indices of URFs containing the |
| | node. (declare unsigned * and give address as parameter) |

Note

The array RDL_ids_ptr has to be to be deallocated with free (*RDL_ids_ptr)

This functions internally enumerates all nodes for each RCF. For repeated queries with different RDL_node it is recommended to enumerate them with RDL_getNodesForRCF once and use this array.

10.2.3.35 RDL_API unsigned RDL_getRCPrototypes (const RDL_data * data, RDL_cycle *** RDL_cycle_array_ptr)

Gives a list of relevant cycle prototypes (one for each RCF).

Returns

the number of prototypes, RDL_INVALID_RESULT on error

Parameters

| data | pointer to the RDL_data holding the calculation results |
|-----------|---|
| RDL_cycle | pointer to the result array (declare RDL_cycle ** and give address as parameter) The result |
| array_ptr | is an array of cycles. |

Note

Result has to be deallocated using RDL_deleteCycles(RDL_cycle **cycles, unsigned number)

10.2.3.36 RDL_API unsigned RDL_getRCycles (const RDL_data * data, RDL_cycle *** RDL_cycle_array_ptr)

Gives a list of all relevant cycles.

Returns

the number of cycles, RDL INVALID RESULT on error

Parameters

| data | pointer to the RDL_data holding the calculation results |
|-----------|---|
| RDL_cycle | pointer to the result array (declare RDL_cycle ** and give address as parameter) The result |
| array_ptr | is an array of cycles. |

Note

Result has to be deallocated using RDL_deleteCycles(RDL_cycle **cycles, unsigned number) Consider using RDL_getRCyclesIterator instead

10.2.3.37 RDL_API unsigned RDL_getRCyclesForRCF (const RDL_data * data, unsigned index, RDL_cycle ***

RDL_cycle_array_ptr)

Gives all relevant cycles of the RCF with the given index.

Returns

the number of cycles found, RDL_INVALID_RESULT on failure

Parameters

| data | pointer to the RDL_data holding the calculation results |
|-----------|---|
| index | the index of the RCF |
| RDL_cycle | pointer that points to the result array of cycles (declare RDL_cycle ** and give address as |
| array_ptr | parameter) |

Note

RDL_cycle_array_ptr has to be deallocated using RDL_deleteCycles(RDL_cycle **cycles, unsigned number)
Consider using RDL_getRCyclesForRCFIterator instead

10.2.3.38 RDL_API RDL_cycleIterator * RDL_getRCyclesForRCFIterator (const RDL_data * data, unsigned index)

Get iterator for all relevant cycles of the RCF with the given index.

Returns

RDL_cycleIterator, NULL on failure

Parameters

| data | pointer to the RDL_data holding the calculation results |
|-------|---|
| index | the index of the RCF |

Note

See RDL_cycleIterator for an example.

10.2.3.39 RDL_API unsigned RDL_getRCyclesForURF (const RDL_data * data, unsigned index, RDL_cycle ***

RDL_cycle_array_ptr)

Gives all relevant cycles of the URF with the given index.

Returns

the number of cycles found, RDL_INVALID_RESULT on failure

Parameters

| data | pointer to the RDL_data holding the calculation results |
|-----------|---|
| index | the index of the URF |
| RDL_cycle | pointer that points to the result array of cycles (declare RDL_cycle ** and give address as |
| array_ptr | parameter) |

Note

RDL_cycle_array_ptr has to be deallocated using RDL_deleteCycles(RDL_cycle **cycles, unsigned number) Consider using RDL_getRCyclesForURFIterator instead

10.2.3.40 RDL_API RDL_cycleIterator * RDL_getRCyclesForURFIterator (const RDL_data * data, unsigned index)

Get iterator for all relevant cycles of the URF with the given index.

Returns

RDL_cycleIterator, NULL on failure

Parameters

| data | pointer to the RDL_data holding the calculation results |
|-------|---|
| index | the index of the URF |

Note

See RDL_cycleIterator for an example.

10.2.3.41 RDL_API RDL_cycleIterator* RDL_getRCyclesIterator (const RDL_data * data)

Get iterator for all relevant cycles.

Returns

RDL_cycleIterator, NULL on failure

Parameters

| data | pointer to the RDL | data holding the calculation results |
|------|--------------------|--------------------------------------|
| | · · | _ |

Note

See RDL_cycleIterator for an example.

10.2.3.42 RDL_API unsigned RDL_getRingsystemForEdge (const RDL_data * data, unsigned from, unsigned to)

Get the ring system id for given edge.

Parameters

| data | pointer to the RDL_data holding the calculation results. |
|------|--|
| from | RDL_node starting at this edge |
| to | RDL_node starting at this edge |

Returns

number of edges in the ring system, RDL_INVALID_RESULT on error RDL_NO_RINGSYSTEM if edge is not part of a ring system

Note

A ring system is a 2-connected component of the graph.

10.2.3.43 RDL API unsigned RDL getSSSR (const RDL data * data, RDL cycle *** RDL cycle array ptr)

Gives a set of cycles that forms a Minimal Cycle Basis of the graph.

Returns

the number of cycles returned (|E|-|V|+1 for connected graphs), RDL_INVALID_RESULT on error

Parameters

| data | pointer to the RDL_data holding the calculation results |
|-----------|--|
| RDL_cycle | pointer that points to the result array (declare RDL_cycle ** and give address as parameter) |
| array_ptr | |

Note

Result has to be deallocated using RDL_deleteCycles(RDL_cycle **cycles, unsigned number) The result is an array of cycles.

10.2.3.44 RDL_API unsigned RDL_getURFsContainingEdge (const RDL_data * data, RDL_node node1, RDL_node node2, unsigned ** RDL_ids_ptr)

Gives all URFs containing the edge.

Returns

the number of URFs containing the edge, RDL_INVALID_RESULT on failure

Parameters

| data | pointer to the RDL_data holding the calculation results |
|-------------|---|
| node1 | the first RDL_node of the edge |
| node2 | the the second RDL_node of the edge |
| RDL_ids_ptr | pointer that points to the result array of integers containing all indices of URFs containing the |
| | edge. (declare unsigned * and give address as parameter) |

Note

The array RDL_ids_ptr has to be to be deallocated with free (*RDL_ids_ptr)

This functions internally enumerates all edges for each URF. For repeated queries with different RDL node it

is recommended to enumerate them with RDL getEdgesForURF once and use this array.

10.2.3.45 RDL_API unsigned RDL_getURFsContainingNode (const RDL_data * data, RDL_node node, unsigned ** RDL_ids_ptr)

Gives all URFs containing the node.

Returns

the number of URFs containing the node, RDL INVALID RESULT on failure

Parameters

| data | pointer to the RDL_data holding the calculation results |
|-------------|---|
| node | the RDL_node |
| RDL_ids_ptr | pointer that points to the result array of integers containing all indices of URFs containing the |
| | node. (declare unsigned * and give address as parameter) |

Note

The array RDL_ids_ptr has to be to be deallocated with free (*RDL_ids_ptr)

This functions internally enumerates all nodes for each URF. For repeated queries with different RDL_node it is recommended to enumerate them with RDL_getNodesForURF once and use this array.

10.2.3.46 RDL_API unsigned RDL_getWeightForRCF (const RDL_data * data, unsigned index)

Returns the weight of each cycle in the RCF identified by its index.

Parameters

| data | pointer to the RDL_data holding the URFs |
|-------|--|
| index | the index of the RCF |

Returns

the weight of the RCF, RDL_INVALID_RESULT on failure (if index is out of range)

10.2.3.47 RDL_API unsigned RDL_getWeightForURF (const RDL_data * data, unsigned index)

Returns the weight of each cycle in the URF identified by its index.

Parameters

| data | pointer to the RDL_data holding the URFs |
|-------|--|
| index | the index of the URF |

Returns

the weight of the URF, RDL_INVALID_RESULT on failure (if index is out of range)

10.2.3.48 RDL_API RDL_graph* RDL_initNewGraph (unsigned nof_nodes)

Initializes a new RDL graph.

Parameters

| nof_nodes | the number of nodes in the graph |
|-----------|----------------------------------|
|-----------|----------------------------------|

Returns

pointer to the new RDL_graph structure.

10.2.3.49 RDL_API void RDL_setOutputFunction (RDL_outputFunction func)

Set the output function for warning and error messages.

Parameters

| func,: | callback function for errors and warnings |
|--------|---|

Warning

This function sets a static variable. When using multiple threads, only one thread should set the output function and the output function should be thread-safe.

Default is no output at all (RDL_writeNothing). Call RDL_setOutputFunction (RDL_writeToStderr) to enable output of warning and error messages to stderr.

10.2.3.50 RDL_API unsigned RDL_translateCycArray (const RDL_data * data, RDL_cycle ** old_array, unsigned number, char *** RDL_cycle_array_ptr)

Translates the results of RDL_getRCycles(), RDL_getSSSR(), RDL_getRCPrototypes() and RDL_getRCyclesFor-URF() (arrays of RDL_cycle) into an array of cycles represented by arrays of $\{0,1\}^{\land}|E|$ (bitsets).

Parameters

| data | pointer to the RDL_data holding the calculation results. |
|-----------|---|
| old_array | The resulting structure of the functions named above |
| number | The return value of the functions named above (the number of cycles given) |
| RDL_cycle | pointer to the result array (declare char ** and give address as parameter) |
| array_ptr | |

Returns

The number of cycles given (same as the parameter 'number'), RDL INVALID RESULT on failure

Note

The initial structure still exists afterwards and still has to be deleted.

The resulting array has a 1 at position i if edge i is part of the cycle or 0 otherwise. An edge is identified by the position at which it was added to the graph starting at 0 (the return value of RDL_addUEdge()).

10.3 src/Test/Test.c File Reference

Demo and validation tool for the RingDecomposerLib library.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "RDLtesting.h"
#include "TestDemo.h"
#include "TestValidate.h"
```

10.3.1 Detailed Description

Demo and validation tool for the RingDecomposerLib library. This tool can be used for demo output and for testing/validation.

Demo output

Start the tool with

```
Test demo <filename>
```

The program will output the ring topology calculated by the library. The input file must be in DIMACS format as described below.

Validation

Start the tool with

```
Test validate <filename> [<timeout>]
```

The program will compare the relevant cycles to the cycles present in the input file. Furthermore the URFs will be validated, i.e. calculated with an independent exponential algorithm (definition of the URFs). The same procedure is applied to the SSSR (verify it's a cycle base). A number of consistency tests is executed to ensure integrity and robustness. See the file test/README for a descriptions of the test files.

The input file must be in DIMACS format as described below. A number of interesting example graphs are in the folder test.

The parameter timeout is optional and specifies a timeout in seconds for the exponential URF validation and the SSSR validation algorithm. (Timeouts are failures!)

File format

The input file format is a (modified) DIMACS graph format. See folder test for example files. Graph format specification:

First line:

```
p <number of nodes> <number of edges> [<nof cycles>]
```

with <nof cycles> optional.

Then the edges of the graph follow

```
e <node1> <node2>
```

If <nof cycles> was specified, then the relevant cycles of the graph are listed:

```
r <ring id> <ring size> <node1> <node2>
```

<ring id> and <ring size> are repeated for each edge of the graph as specified by <node1> and <node2>.

Index

| getitem | py_rdl::Graph::Graph, 30 |
|---|--|
| py_rdl::Calculator::Calculator, 22 | get_nodes_for_rcf |
| init | py_rdl::Calculator::Calculator, 23 |
| py_rdl::Calculator::Calculator, 22 | get_nodes_for_ringsystem |
| py_rdl::CycleFamily::CycleFamily, 28 | py_rdl::Calculator::Calculator, 23 |
| py_rdl::CycleFamily::RCF, 31 | get_nodes_for_urf |
| py_rdl::CycleFamily::ResultCollection, 32 | py_rdl::Calculator::Calculator, 24 |
| py_rdl::CycleFamily::Ringsystem, 33 | get_nof_nodes |
| py_rdl::CycleFamily::URF, 34 | py_rdl::Graph::Graph, 30 |
| py rdl::Graph::Graph, 28 | get_nof_rcf |
| py_rdl::wrapper::Cycle::Cycle, 27 | py_rdl::Calculator::Calculator, 24 |
| len | get_nof_relevant_cycles |
| py_rdl::Calculator::Calculator, 22 | py_rdl::Calculator::Calculator, 24 |
| repr | get_nof_relevant_cycles_for_urf |
| py_rdl::CycleFamily::ResultCollection, 32 | py_rdl::Calculator::Calculator, 24 |
| str | • • — |
| | get_nof_ringsystems |
| py_rdl::CycleFamily::RCF, 31 | py_rdl::Calculator::Calculator, 24 |
| py_rdl::CycleFamily::ResultCollection, 32 | get_nof_urf |
| py_rdl::CycleFamily::URF, 34 | py_rdl::Calculator::Calculator, 24 |
| add_edge | get_rcfs_for_edge |
| - | py_rdl::Calculator::Calculator, 24 |
| py_rdl::Graph::Graph, 28 | get_rcfs_for_node |
| calculate | py_rdl::Calculator::Calculator, 24 |
| py_rdl::Calculator::Calculator, 22 | get_relevant_cycle_prototypes |
| py_run.oaiculatoroaiculator, 22 | py_rdl::Calculator::Calculator, 24 |
| edges | get_relevant_cycles |
| RDL_cycle, 31 | py_rdl::Calculator::Calculator, 25 |
| 1122_oyo.o, 01 | get_relevant_cycles_for_rcf |
| from_edges | py_rdl::Calculator::Calculator, 25 |
| py_rdl::Graph::Graph, 29 | get_relevant_cycles_for_urf |
| 1 7 | py_rdl::Calculator::Calculator, 25 |
| get_calculated_result | get_sssr |
| py_rdl::Calculator::Calculator, 22 | py_rdl::Calculator::Calculator, 25 |
| get_calculated_result_for_graph | get_urfs_for_edge |
| py_rdl::Calculator::Calculator, 23 | py_rdl::Calculator::Calculator, 25 |
| get_edge_for_indices | get_urfs_for_node |
| py_rdl::Graph::Graph, 29 | py_rdl::Calculator::Calculator, 25 |
| get_edges | get_weight_for_rcf |
| py_rdl::Graph::Graph, 29 | py_rdl::Calculator::Calculator, 25 |
| get_edges_for_rcf | get_weight_for_urf |
| py_rdl::Calculator::Calculator, 23 | py rdl::Calculator::Calculator, 25 |
| get_edges_for_ringsystem | py_ruiGalculatorGalculator, 25 |
| py_rdl::Calculator::Calculator, 23 | is_calculated |
| get edges for urf | py_rdl::Calculator::Calculator, 25 |
| py_rdl::Calculator::Calculator, 23 | py_ruiCalculatorCalculator, 25 |
| get_index_for_node | py_rdl.Calculator.Calculator, 21 |
| py_rdl::Graph::Graph, 29 | py_rdl.CycleFamily.CycleFamily, 27 |
| get_indices_for_edge | py_rdl.CycleFamily.CycleFamily.RCF, 30 |
| py_rdl::Graph::Graph, 29 | py_rdl.CycleFamily.ResultCollection, 32 |
| aet node for index | py_rdi.CycleFamily.Ringsystem, 33 |
| AUL HUUU IUI IIIUUA | DY TUILOYOUT AITHIV.I IIIUSVSICIII. 11.1 |

INDEX 59

| py_rdl.CycleFamily.URF, 33 | get_indices_for_edge, 29 |
|---|--|
| py_rdl.Graph.Graph, 28 | get_node_for_index, 30 |
| py_rdl.wrapper.Cycle.Cycle, 26 | get_nof_nodes, 30 |
| py_rdl::Calculator::Calculator | py_rdl::wrapper::Cycle::Cycle |
| getitem, 22 | init, 27 |
| init, 22 | weight, 27 |
| len, 22 | RDL_addUEdge |
| calculate, 22 | RingDecomposerLib.h, 39 |
| get_calculated_result, 22 | RDL calculate |
| get_calculated_result_for_graph, 23 | RingDecomposerLib.h, 39 |
| get_edges_for_rcf, 23 | RDL_cycle, 31 |
| get_edges_for_ringsystem, 23 | edges, 31 |
| get_edges_for_urf, 23 | rcf, 31 |
| get_nodes_for_rcf, 23 | RingDecomposerLib.h, 38 |
| get_nodes_for_ringsystem, 23 | urf, 31 |
| get_nodes_for_urf, 24 | weight, 31 |
| get_nof_rcf, 24 | RDL_cycleIterator |
| get_nof_relevant_cycles, 24 | RingDecomposerLib.h, 38 |
| get_nof_relevant_cycles_for_urf, 24 | RDL_cycleIteratorAtEnd |
| get_nof_ringsystems, 24 | RingDecomposerLib.h, 40 |
| get_nof_urf, 24 | RDL_cycleIteratorGetCycle |
| get_rcfs_for_edge, 24 | RingDecomposerLib.h, 40 |
| get_rcfs_for_node, 24 | RDL_cycleIteratorNext |
| get_relevant_cycle_prototypes, 24 | RingDecomposerLib.h, 40 |
| get_relevant_cycles, 25 | RDL_data |
| get_relevant_cycles_for_rcf, 25 | RingDecomposerLib.h, 39 |
| get_relevant_cycles_for_urf, 25 | RDL_deleteCycle |
| get_sssr, 25 | RingDecomposerLib.h, 40 |
| get_urfs_for_edge, 25 get_urfs_for_node, 25 | RDL_deleteCycleIterator |
| get_uris_lor_node, 25 get_weight_for_rcf, 25 | RingDecomposerLib.h, 42 |
| get_weight_for_urf, 25 | RDL_deleteCycles |
| is_calculated, 25 | RingDecomposerLib.h, 42 |
| rcfs, 26 | RDL_deleteData |
| ringsystems, 26 | RingDecomposerLib.h, 42 |
| set_graph, 26 | RDL_deleteEdgeldxArray |
| urfs, 26 | RingDecomposerLib.h, 42 |
| py_rdl::CycleFamily::CycleFamily | RDL_deleteGraph |
| init, 28 | RingDecomposerLib.h, 42 |
| py_rdl::CycleFamily::RCF | RDL_getEdgeArray RingDecomposerLib.h, 42 |
| init, 31 | RDL_getEdgeId |
| str, 31 | RingDecomposerLib.h, 43 |
| py_rdl::CycleFamily::ResultCollection | RDL getEdgesForRCF |
| init, 32 | RingDecomposerLib.h, 43 |
| , <u>32</u> | RDL_getEdgesForRingsystem |
| , 32 | RingDecomposerLib.h, 43 |
| py_rdl::CycleFamily::Ringsystem | RDL_getEdgesForURF |
| init, 33 | RingDecomposerLib.h, 45 |
| py_rdl::CycleFamily::URF | RDL_getNodesForRCF |
| init, 34 | RingDecomposerLib.h, 45 |
| str, 34 | RDL_getNodesForRingsystem |
| py_rdl::Graph::Graph | RingDecomposerLib.h, 46 |
| init, 28 | RDL_getNodesForURF |
| add_edge, 28 | RingDecomposerLib.h, 46 |
| from_edges, 29 | RDL_getNofEdges |
| get_edge_for_indices, 29 | RingDecomposerLib.h, 46 |
| get_edges, 29 | RDL_getNofEdgesForRingsystem |
| get_index_for_node, 29 | RingDecomposerLib.h, 47 |

60 INDEX

| RDL_getNofNodesForRingsystem | RDL_translateCycArray |
|---|--|
| RingDecomposerLib.h, 47 | RingDecomposerLib.h, 55 |
| RDL_getNofRC | rcf |
| RingDecomposerLib.h, 47 | RDL_cycle, 31 |
| RDL_getNofRCF | rcfs |
| RingDecomposerLib.h, 47 | py_rdl::Calculator::Calculator, 26 |
| RDL_getNofRCFContainingEdge | RingDecomposerLib.h |
| RingDecomposerLib.h, 48 | RDL_addUEdge, 39 |
| RDL_getNofRCFContainingNode | RDL_calculate, 39 |
| RingDecomposerLib.h, 48 | RDL_cycle, 38 |
| RDL_getNofRCForRCF | RDL_cycleIterator, 38 |
| RingDecomposerLib.h, 48 | RDL_cycleIteratorAtEnd, 40 |
| RDL_getNofRCForURF | RDL_cycleIteratorGetCycle, 40 |
| RingDecomposerLib.h, 49 | RDL_cycleIteratorNext, 40 |
| RDL_getNofRingsystems | RDL_data, 39 |
| RingDecomposerLib.h, 49 | RDL_deleteCycle, 40 |
| RDL_getNofURF | RDL_deleteCycleIterator, 42 |
| RingDecomposerLib.h, 49 | RDL_deleteCycles, 42 |
| RDL_getNofURFContainingEdge | RDL deleteData, 42 |
| RingDecomposerLib.h, 49 | RDL_deleteEdgeldxArray, 42 |
| RDL_getNofURFContainingNode | RDL_deleteGraph, 42 |
| RingDecomposerLib.h, 50 | RDL_getEdgeArray, 42 |
| RDL_getRCFsContainingEdge | RDL_getEdgeld, 43 |
| RingDecomposerLib.h, 50 | RDL_getEdgesForRCF, 43 |
| RDL_getRCFsContainingNode | RDL_getEdgesForRingsystem, 43 |
| RingDecomposerLib.h, 50 | RDL_getEdgesForURF, 45 |
| RDL_getRCPrototypes | RDL getNodesForRCF, 45 |
| RingDecomposerLib.h, 51 | RDL_getNodesForRingsystem, 46 |
| RDL_getRCycles | RDL_getNodesForURF, 46 |
| RingDecomposerLib.h, 51 | RDL_getNofEdges, 46 |
| RDL_getRCyclesForRCF | RDL_getNofEdgesForRingsystem, 47 |
| RingDecomposerLib.h, 51 | RDL_getNofNodesForRingsystem, 47 |
| RDL_getRCyclesForRCFIterator | RDL getNofRC, 47 |
| RingDecomposerLib.h, 52 | RDL getNofRCF, 47 |
| RDL_getRCyclesForURF | RDL_getNofRCFContainingEdge, 48 |
| RingDecomposerLib.h, 52 | RDL_getNofRCFContainingNode, 48 |
| RDL getRCyclesForURFIterator | RDL_getNofRCForRCF, 48 |
| RingDecomposerLib.h, 52 | RDL getNofRCForURF, 49 |
| RDL_getRCyclesIterator | RDL_getNofRingsystems, 49 |
| RingDecomposerLib.h, 53 | RDL getNofURF, 49 |
| RDL getRingsystemForEdge | RDL getNofURFContainingEdge, 49 |
| RingDecomposerLib.h, 53 | RDL_getNofURFContainingNode, 50 |
| RDL_getSSSR | RDL_getRCFsContainingRode, 50 |
| RingDecomposerLib.h, 53 | RDL getRCFsContainingNode, 50 |
| RDL_getURFsContainingEdge | RDL_getRCPrototypes, 51 |
| RingDecomposerLib.h, 54 | RDL getRCycles, 51 |
| RDL getURFsContainingNode | RDL_getRCyclesForRCF, 51 |
| RingDecomposerLib.h, 54 | RDL_getRCyclesForRCFIterator, 52 |
| RDL_getWeightForRCF | RDL_getRCyclesForURF, 52 |
| | · |
| RingDecomposerLib.h, 54 RDL getWeightForURF | RDL_getRCyclesForURFIterator, 52 RDL_getRCyclesIterator, 53 |
| | _ · |
| RingDecomposerLib.h, 55 | RDL_getRingsystemForEdge, 53 |
| RDL_graph | RDL_getSSSR, 53 |
| RingDecomposerLib.h, 39 | RDL_getURFsContainingEdge, 54 |
| RDL_initNewGraph | RDL_getURFsContainingNode, 54 |
| RingDecomposerLib.h, 55 | RDL_getWeightForkIRF 55 |
| RDL_setOutputFunction | RDL_getWeightForURF, 55 |
| RingDecomposerLib.h, 55 | RDL_graph, 39 |

INDEX 61

```
RDL_initNewGraph, 55
     RDL_setOutputFunction, 55
     RDL_translateCycArray, 55
ringsystems
    py_rdl::Calculator::Calculator, 26
set_graph
    py_rdl::Calculator::Calculator, 26
src/MinimalExample/MinimalExample.c, 35
src/RingDecomposerLib/RingDecomposerLib.h, 35
src/Test/Test.c, 56
urf
     RDL_cycle, 31
urfs
    py_rdl::Calculator::Calculator, 26
weight
    py_rdl::wrapper::Cycle::Cycle, 27
    RDL_cycle, 31
```