

# CRyptography And Groups (CRAG)

## 1.5

Generated by Doxygen 1.6.1

Mon Sep 26 18:43:45 2011



# Contents

|          |  |           |
|----------|--|-----------|
| <b>1</b> | <b>Bug List</b>  | <b>1</b>  |
| <b>2</b> | <b>Directory Hierarchy</b>                               | <b>3</b>  |
| 2.1      | Directories . . . . .                                    | 3         |
| <b>3</b> | <b>Namespace Index</b>                                   | <b>5</b>  |
| 3.1      | Namespace List . . . . .                                 | 5         |
| <b>4</b> | <b>Class Index</b>                                       | <b>7</b>  |
| 4.1      | Class Hierarchy . . . . .                                | 7         |
| <b>5</b> | <b>Class Index</b>                                       | <b>13</b> |
| 5.1      | Class List . . . . .                                     | 13        |
| <b>6</b> | <b>File Index</b>  | <b>19</b> |
| 6.1      | File List . . . . .                                      | 19        |
| <b>7</b> | <b>Directory Documentation</b>                           | <b>23</b> |
| 7.1      | Alphabet/ Directory Reference . . . . .                  | 23        |
| 7.2      | BraidGroup/ Directory Reference . . . . .                | 24        |
| 7.3      | CryptoAAG/ Directory Reference . . . . .                 | 25        |
| 7.4      | CryptoAE/ Directory Reference . . . . .                  | 26        |
| 7.5      | CryptoKL/ Directory Reference . . . . .                  | 27        |
| 7.6      | CryptoShftConj/ Directory Reference . . . . .            | 28        |
| 7.7      | CryptoTripleDecomposition/ Directory Reference . . . . . | 29        |

|      |  |    |
|------|--|----|
| 7.8  | Elt/ Directory Reference . . . . .                               | 30 |
| 7.9  | Equation/ Directory Reference . . . . .                          | 31 |
| 7.10 | Experiments/ Directory Reference . . . . .                       | 32 |
| 7.11 | FreeGroup/ Directory Reference . . . . .                         | 33 |
| 7.12 | FreeMetabelianGroup/ Directory Reference . . . . .               | 34 |
| 7.13 | general/ Directory Reference . . . . .                           | 35 |
| 7.14 | Graph/ Directory Reference . . . . .                             | 36 |
| 7.15 | Graphics/ Directory Reference . . . . .                          | 37 |
| 7.16 | Group/ Directory Reference . . . . .                             | 38 |
| 7.17 | HigmanGroup/ Directory Reference . . . . .                       | 39 |
| 7.18 | ThompsonGroup/include/ Directory Reference . . . . .             | 40 |
| 7.19 | TheGrigorchukGroup/include/ Directory Reference . . . . .        | 41 |
| 7.20 | StringSimilarity/include/ Directory Reference . . . . .          | 42 |
| 7.21 | ranlib/include/ Directory Reference . . . . .                    | 43 |
| 7.22 | Maps/include/ Directory Reference . . . . .                      | 44 |
| 7.23 | HigmanGroup/include/ Directory Reference . . . . .               | 45 |
| 7.24 | Group/include/ Directory Reference . . . . .                     | 46 |
| 7.25 | Graphics/include/ Directory Reference . . . . .                  | 47 |
| 7.26 | Graph/include/ Directory Reference . . . . .                     | 48 |
| 7.27 | general/include/ Directory Reference . . . . .                   | 49 |
| 7.28 | FreeMetabelianGroup/include/ Directory Reference . . . . .       | 50 |
| 7.29 | SbgpFG/include/ Directory Reference . . . . .                    | 51 |
| 7.30 | FreeGroup/include/ Directory Reference . . . . .                 | 52 |
| 7.31 | Experiments/include/ Directory Reference . . . . .               | 53 |
| 7.32 | Equation/include/ Directory Reference . . . . .                  | 54 |
| 7.33 | Elt/include/ Directory Reference . . . . .                       | 55 |
| 7.34 | CryptoTripleDecomposition/include/ Directory Reference . . . . . | 56 |
| 7.35 | CryptoShftConj/include/ Directory Reference . . . . .            | 57 |
| 7.36 | CryptoKL/include/ Directory Reference . . . . .                  | 58 |
| 7.37 | CryptoAE/include/ Directory Reference . . . . .                  | 59 |
| 7.38 | CryptoAAG/include/ Directory Reference . . . . .                 | 60 |

|          |   |           |
|----------|---|-----------|
| 7.39     | BraidGroup/include/ Directory Reference . . . . . | 61        |
| 7.40     | Alphabet/include/ Directory Reference . . . . .   | 62        |
| 7.41     | Maps/ Directory Reference . . . . .               | 63        |
| 7.42     | ranlib/ Directory Reference . . . . .             | 64        |
| 7.43     | SbgpFG/ Directory Reference . . . . .             | 65        |
| 7.44     | StringSimilarity/ Directory Reference . . . . .   | 66        |
| 7.45     | TheGrigorchukGroup/ Directory Reference . . . . . | 67        |
| 7.46     | ThompsonGroup/ Directory Reference . . . . .      | 68        |
| <b>8</b> | <b>Namespace Documentation</b>                    | <b>69</b> |
| 8.1      | __gnu_cxx Namespace Reference . . . . .           | 69        |
| 8.2      | AAGChallenge Namespace Reference . . . . .        | 70        |
| 8.2.1    | Function Documentation . . . . .                  | 71        |
| 8.2.1.1  | generateKeyDecomp . . . . .                       | 71        |
| 8.2.1.2  | generateSubgroup . . . . .                        | 71        |
| 8.2.1.3  | getDelta . . . . .                                | 71        |
| 8.2.1.4  | getDpSubgroup . . . . .                           | 71        |
| 8.2.1.5  | getSgGenComponentsRandom . . . . .                | 71        |
| 8.2.1.6  | getSgGenComponentsSquares . . . . .               | 71        |
| 8.2.1.7  | randomSubgroupWord . . . . .                      | 71        |
| 8.2.1.8  | specialSubgroupWord . . . . .                     | 71        |
| 8.2.1.9  | specialSubgroupWordRandom . . . . .               | 71        |
| 8.3      | BG Namespace Reference . . . . .                  | 72        |
| 8.3.1    | Function Documentation . . . . .                  | 72        |
| 8.3.1.1  | solveWPinBG . . . . .                             | 72        |
| 8.3.1.2  | solveWPinBG . . . . .                             | 72        |
| 8.4      | Graphs Namespace Reference . . . . .              | 73        |
| 8.4.1    | Detailed Description . . . . .                    | 74        |
| 8.4.2    | Function Documentation . . . . .                  | 75        |
| 8.4.2.1  | fold . . . . .                                    | 75        |
| 8.4.2.2  | fold . . . . .                                    | 75        |

|          |                                    |    |
|----------|------------------------------------|----|
| 8.4.2.3  | fold . . . . .                     | 75 |
| 8.4.2.4  | getDistances_in . . . . .          | 75 |
| 8.4.2.5  | getDistances_out . . . . .         | 75 |
| 8.4.2.6  | getGeodesicTree_in . . . . .       | 76 |
| 8.4.2.7  | getGeodesicTree_out . . . . .      | 76 |
| 8.4.2.8  | liftup . . . . .                   | 76 |
| 8.4.2.9  | operator<< . . . . .               | 76 |
| 8.4.2.10 | readoffGeodesicTree . . . . .      | 76 |
| 8.4.2.11 | reduce_path . . . . .              | 77 |
| 8.4.2.12 | trace . . . . .                    | 77 |
| 8.4.2.13 | trace_path . . . . .               | 77 |
| 8.4.2.14 | unfold . . . . .                   | 77 |
| 8.5      | msgs Namespace Reference . . . . . | 78 |
| 8.5.1    | Detailed Description . . . . .     | 78 |
| 8.5.2    | Function Documentation . . . . .   | 78 |
| 8.5.2.1  | error . . . . .                    | 78 |
| 8.5.2.2  | error . . . . .                    | 78 |
| 8.5.2.3  | error . . . . .                    | 79 |
| 8.5.2.4  | warn . . . . .                     | 79 |
| 8.6      | PC Namespace Reference . . . . .   | 80 |
| 8.6.1    | Detailed Description . . . . .     | 80 |
| 8.6.2    | Typedef Documentation . . . . .    | 82 |
| 8.6.2.1  | Sign . . . . .                     | 82 |
| 8.6.3    | Function Documentation . . . . .   | 82 |
| 8.6.3.1  | addSigns . . . . .                 | 82 |
| 8.6.3.2  | compareSigns . . . . .             | 82 |
| 8.6.3.3  | negateSign . . . . .               | 82 |
| 8.6.3.4  | signToInt . . . . .                | 82 |
| 8.6.3.5  | signToString . . . . .             | 82 |
| 8.6.4    | Variable Documentation . . . . .   | 82 |
| 8.6.4.1  | MINUS . . . . .                    | 82 |

|          |  |           |
|----------|--|-----------|
| 8.6.4.2  | PLUS . . . . .                                   | 82        |
| 8.6.4.3  | UNDEFINED_COL . . . . .                          | 82        |
| 8.6.4.4  | UNDEFINED_ROW . . . . .                          | 82        |
| 8.6.4.5  | ZERO . . . . .                                   | 83        |
| 8.7      | RMap Namespace Reference . . . . .               | 84        |
| 8.7.1    | Detailed Description . . . . .                   | 84        |
| 8.7.2    | Function Documentation . . . . .                 | 84        |
| 8.7.2.1  | getRandomAuto . . . . .                          | 84        |
| 8.7.2.2  | getRandomWhiteheadAuto . . . . .                 | 84        |
| 8.8      | WhiteheadAutoSet Namespace Reference . . . . .   | 85        |
| 8.8.1    | Function Documentation . . . . .                 | 85        |
| 8.8.1.1  | reduceBy . . . . .                               | 85        |
| <b>9</b> | <b>Class Documentation</b>                       | <b>87</b> |
| 9.1      | AAGKeyPairGenerator Class Reference . . . . .    | 87        |
| 9.1.1    | Detailed Description . . . . .                   | 88        |
| 9.1.2    | Constructor & Destructor Documentation . . . . . | 88        |
| 9.1.2.1  | AAGKeyPairGenerator . . . . .                    | 88        |
| 9.1.3    | Member Function Documentation . . . . .          | 88        |
| 9.1.3.1  | getFalsePair . . . . .                           | 88        |
| 9.1.3.2  | getTruePair . . . . .                            | 88        |
| 9.1.4    | Member Data Documentation . . . . .              | 89        |
| 9.1.4.1  | AliceDecompositionLength . . . . .               | 89        |
| 9.1.4.2  | BobDecompositionLength . . . . .                 | 89        |
| 9.1.4.3  | maxLen . . . . .                                 | 89        |
| 9.1.4.4  | minLen . . . . .                                 | 89        |
| 9.1.4.5  | nGens . . . . .                                  | 89        |
| 9.1.4.6  | nSgGens . . . . .                                | 89        |
| 9.2      | AAGProtocolInstance Class Reference . . . . .    | 90        |
| 9.2.1    | Detailed Description . . . . .                   | 91        |
| 9.2.2    | Constructor & Destructor Documentation . . . . . | 91        |

|          |  |    |
|----------|--|----|
| 9.2.2.1  | AAGProtocolInstance . . . . .                    | 91 |
| 9.2.3    | Member Function Documentation . . . . .          | 91 |
| 9.2.3.1  | challenge . . . . .                              | 91 |
| 9.2.3.2  | generateHardProductOfGenerators . . . . .        | 91 |
| 9.2.3.3  | getAliceConjSbgp . . . . .                       | 91 |
| 9.2.3.4  | getAliceKey . . . . .                            | 91 |
| 9.2.3.5  | getAlicePublicSbgp . . . . .                     | 91 |
| 9.2.3.6  | getBobConjSbgp . . . . .                         | 92 |
| 9.2.3.7  | getBobKey . . . . .                              | 92 |
| 9.2.3.8  | getBobPublicSbgp . . . . .                       | 92 |
| 9.2.3.9  | getSharedKey . . . . .                           | 92 |
| 9.2.3.10 | printStats . . . . .                             | 92 |
| 9.2.3.11 | random . . . . .                                 | 92 |
| 9.2.4    | Member Data Documentation . . . . .              | 92 |
| 9.2.4.1  | AliceConjugatedSbgp . . . . .                    | 92 |
| 9.2.4.2  | AliceKey . . . . .                               | 92 |
| 9.2.4.3  | AliceKeyDecomposition . . . . .                  | 93 |
| 9.2.4.4  | AlicePublicSbgp . . . . .                        | 93 |
| 9.2.4.5  | BobConjugatedSbgp . . . . .                      | 93 |
| 9.2.4.6  | BobKey . . . . .                                 | 93 |
| 9.2.4.7  | BobKeyDecomposition . . . . .                    | 93 |
| 9.2.4.8  | BobPublicSbgp . . . . .                          | 93 |
| 9.2.4.9  | theSharedKey . . . . .                           | 93 |
| 9.3      | AdvDehnAlgorithm Class Reference . . . . .       | 94 |
| 9.3.1    | Detailed Description . . . . .                   | 94 |
| 9.3.2    | Constructor & Destructor Documentation . . . . . | 95 |
| 9.3.2.1  | AdvDehnAlgorithm . . . . .                       | 95 |
| 9.3.2.2  | AdvDehnAlgorithm . . . . .                       | 95 |
| 9.3.3    | Member Function Documentation . . . . .          | 95 |
| 9.3.3.1  | addCycle . . . . .                               | 95 |
| 9.3.3.2  | builtup . . . . .                                | 95 |



|         |  |     |
|---------|--|-----|
| 9.3.3.3 | getFSA . . . . .                                 | 95  |
| 9.3.3.4 | isLoop . . . . .                                 | 95  |
| 9.3.4   | Member Data Documentation . . . . .              | 95  |
| 9.3.4.1 | checkedStates . . . . .                          | 95  |
| 9.3.4.2 | theFSA . . . . .                                 | 95  |
| 9.3.4.3 | theGroup . . . . .                               | 95  |
| 9.3.4.4 | theWord . . . . .                                | 96  |
| 9.4     | AEKeyExchange Class Reference . . . . .          | 97  |
| 9.4.1   | Detailed Description . . . . .                   | 97  |
| 9.4.2   | Constructor & Destructor Documentation . . . . . | 97  |
| 9.4.2.1 | AEKeyExchange . . . . .                          | 97  |
| 9.4.3   | Member Function Documentation . . . . .          | 98  |
| 9.4.3.1 | alicePublicKey . . . . .                         | 98  |
| 9.4.3.2 | bobPublicKey . . . . .                           | 98  |
| 9.4.3.3 | generatePublicKey . . . . .                      | 98  |
| 9.4.3.4 | generateTuples . . . . .                         | 98  |
| 9.4.3.5 | starMult . . . . .                               | 98  |
| 9.4.4   | Member Data Documentation . . . . .              | 98  |
| 9.4.4.1 | M0 . . . . .                                     | 98  |
| 9.4.4.2 | the_n . . . . .                                  | 98  |
| 9.4.4.3 | the_p . . . . .                                  | 98  |
| 9.4.4.4 | theTTPTuple . . . . .                            | 98  |
| 9.5     | AImage Class Reference . . . . .                 | 100 |
| 9.5.1   | Detailed Description . . . . .                   | 100 |
| 9.5.2   | Constructor & Destructor Documentation . . . . . | 100 |
| 9.5.2.1 | AImage . . . . .                                 | 100 |
| 9.5.2.2 | AImage . . . . .                                 | 101 |
| 9.5.2.3 | AImage . . . . .                                 | 101 |
| 9.5.2.4 | AImage . . . . .                                 | 101 |
| 9.5.2.5 | AImage . . . . .                                 | 101 |
| 9.5.3   | Member Function Documentation . . . . .          | 101 |

|         |  |     |
|---------|--|-----|
| 9.5.3.1 | getHeight . . . . .                              | 101 |
| 9.5.3.2 | getSize . . . . .                                | 101 |
| 9.5.3.3 | getType . . . . .                                | 101 |
| 9.5.3.4 | getWidth . . . . .                               | 101 |
| 9.5.3.5 | saveTo . . . . .                                 | 102 |
| 9.5.4   | Member Data Documentation . . . . .              | 102 |
| 9.5.4.1 | height . . . . .                                 | 102 |
| 9.5.4.2 | size . . . . .                                   | 102 |
| 9.5.4.3 | width . . . . .                                  | 102 |
| 9.6     | Alphabet Class Reference . . . . .               | 103 |
| 9.6.1   | Detailed Description . . . . .                   | 103 |
| 9.6.2   | Member Function Documentation . . . . .          | 104 |
| 9.6.2.1 | getLetter . . . . .                              | 104 |
| 9.6.2.2 | getNum . . . . .                                 | 104 |
| 9.6.2.3 | printVector . . . . .                            | 104 |
| 9.6.2.4 | printWord . . . . .                              | 104 |
| 9.6.2.5 | readVector . . . . .                             | 104 |
| 9.6.2.6 | readWord . . . . .                               | 104 |
| 9.7     | AParser Class Reference . . . . .                | 105 |
| 9.7.1   | Detailed Description . . . . .                   | 105 |
| 9.7.2   | Constructor & Destructor Documentation . . . . . | 105 |
| 9.7.2.1 | AParser . . . . .                                | 105 |
| 9.7.2.2 | ~AParser . . . . .                               | 105 |
| 9.7.3   | Member Function Documentation . . . . .          | 105 |
| 9.7.3.1 | getAlphabet . . . . .                            | 105 |
| 9.7.3.2 | getType . . . . .                                | 105 |
| 9.7.3.3 | parse . . . . .                                  | 105 |
| 9.7.4   | Member Data Documentation . . . . .              | 105 |
| 9.7.4.1 | localFlexLexer . . . . .                         | 105 |
| 9.7.4.2 | theAlphabet . . . . .                            | 106 |
| 9.7.4.3 | theType . . . . .                                | 106 |

|          |   |     |
|----------|---|-----|
| 9.8      | StraightLineProgramWord::Assertion Struct Reference . . . . . | 107 |
| 9.8.1    | Detailed Description . . . . .                                | 107 |
| 9.8.2    | Constructor & Destructor Documentation . . . . .              | 108 |
| 9.8.2.1  | Assertion . . . . .   | 108 |
| 9.8.3    | Member Function Documentation . . . . .                       | 108 |
| 9.8.3.1  | operator< . . . . .   | 108 |
| 9.8.3.2  | similar . . . . .   | 108 |
| 9.8.4    | Friends And Related Function Documentation . . . . .          | 108 |
| 9.8.4.1  | operator<< . . . . .  | 108 |
| 9.8.5    | Member Data Documentation . . . . .                           | 108 |
| 9.8.5.1  | theBase1 . . . . .  | 108 |
| 9.8.5.2  | theLength . . . . .   | 108 |
| 9.8.5.3  | theVertex1 . . . . .  | 109 |
| 9.8.5.4  | theVertex2 . . . . .  | 109 |
| 9.9      | AutoSet Class Reference . . . . .                             | 110 |
| 9.9.1    | Detailed Description . . . . .                                | 110 |
| 9.9.2    | Member Function Documentation . . . . .                       | 110 |
| 9.9.2.1  | getSet . . . . .  | 110 |
| 9.10     | BalancedTree< Obj > Class Template Reference . . . . .        | 111 |
| 9.10.1   | Detailed Description . . . . .                                | 112 |
| 9.10.2   | Constructor & Destructor Documentation . . . . .              | 112 |
| 9.10.2.1 | BalancedTree . . . . .  | 112 |
| 9.10.2.2 | BalancedTree . . . . .  | 112 |
| 9.10.2.3 | ~BalancedTree . . . . .                                       | 113 |
| 9.10.3   | Member Function Documentation . . . . .                       | 113 |
| 9.10.3.1 | getList . . . . .   | 113 |
| 9.10.3.2 | getList . . . . .   | 113 |
| 9.10.3.3 | insert . . . . .  | 113 |
| 9.10.3.4 | insert . . . . .  | 114 |
| 9.10.3.5 | insert . . . . .  | 114 |
| 9.10.3.6 | rotateLeft . . . . .  | 114 |

|          |  |     |
|----------|--|-----|
| 9.10.3.7 | rotateRight . . . . .                            | 114 |
| 9.10.3.8 | size . . . . .                                   | 115 |
| 9.10.4   | Member Data Documentation . . . . .              | 115 |
| 9.10.4.1 | theRoot . . . . .                                | 115 |
| 9.11     | BG::BGMonomial Struct Reference . . . . .        | 116 |
| 9.11.1   | Detailed Description . . . . .                   | 116 |
| 9.11.2   | Member Enumeration Documentation . . . . .       | 116 |
| 9.11.2.1 | "@0 . . . . .                                    | 116 |
| 9.11.3   | Member Data Documentation . . . . .              | 117 |
| 9.11.3.1 | expb . . . . .                                   | 117 |
| 9.11.3.2 | K . . . . .                                      | 117 |
| 9.11.3.3 | type . . . . .                                   | 117 |
| 9.11.3.4 | U . . . . .                                      | 117 |
| 9.11.3.5 | X . . . . .                                      | 117 |
| 9.12     | BKLeftNormalForm Class Reference . . . . .       | 118 |
| 9.12.1   | Detailed Description . . . . .                   | 121 |
| 9.12.2   | Member Typedef Documentation . . . . .           | 121 |
| 9.12.2.1 | NF . . . . .                                     | 121 |
| 9.12.3   | Member Enumeration Documentation . . . . .       | 121 |
| 9.12.3.1 | transformationResult . . . . .                   | 121 |
| 9.12.4   | Constructor & Destructor Documentation . . . . . | 121 |
| 9.12.4.1 | BKLeftNormalForm . . . . .                       | 121 |
| 9.12.4.2 | BKLeftNormalForm . . . . .                       | 122 |
| 9.12.4.3 | BKLeftNormalForm . . . . .                       | 122 |
| 9.12.4.4 | BKLeftNormalForm . . . . .                       | 122 |
| 9.12.4.5 | BKLeftNormalForm . . . . .                       | 122 |
| 9.12.5   | Member Function Documentation . . . . .          | 122 |
| 9.12.5.1 | adjustDecomposition . . . . .                    | 122 |
| 9.12.5.2 | areConjugate . . . . .                           | 122 |
| 9.12.5.3 | cycle . . . . .                                  | 123 |
| 9.12.5.4 | decycle . . . . .                                | 123 |

|           |  |     |
|-----------|--|-----|
| 9.12.5.5  | findSSSRepresentative . . . . .              | 123 |
| 9.12.5.6  | getDecomposition . . . . .                   | 123 |
| 9.12.5.7  | getPower . . . . .                           | 123 |
| 9.12.5.8  | getSimpleConjugators . . . . .               | 123 |
| 9.12.5.9  | getSimpleSummitConjugators . . . . .         | 124 |
| 9.12.5.10 | getTinyTwistPermutation . . . . .            | 124 |
| 9.12.5.11 | getWord . . . . .                            | 124 |
| 9.12.5.12 | inverse . . . . .                            | 124 |
| 9.12.5.13 | isTrivial . . . . .                          | 124 |
| 9.12.5.14 | multiply . . . . .                           | 124 |
| 9.12.5.15 | operator NF . . . . .                        | 124 |
| 9.12.5.16 | operator!= . . . . .                         | 125 |
| 9.12.5.17 | operator* . . . . .                          | 125 |
| 9.12.5.18 | operator*= . . . . .                         | 125 |
| 9.12.5.19 | operator- . . . . .                          | 125 |
| 9.12.5.20 | operator< . . . . .                          | 125 |
| 9.12.5.21 | operator= . . . . .                          | 125 |
| 9.12.5.22 | operator= . . . . .                          | 125 |
| 9.12.5.23 | operator== . . . . .                         | 125 |
| 9.12.5.24 | setDecomposition . . . . .                   | 125 |
| 9.12.5.25 | setPower . . . . .                           | 126 |
| 9.12.5.26 | transform . . . . .                          | 126 |
| 9.12.6    | Member Data Documentation . . . . .          | 126 |
| 9.12.6.1  | theDecomposition . . . . .                   | 126 |
| 9.12.6.2  | theOmegaPower . . . . .                      | 126 |
| 9.12.6.3  | theRank . . . . .                            | 126 |
| 9.13      | BKLRightNormalForm Class Reference . . . . . | 127 |
| 9.13.1    | Detailed Description . . . . .               | 129 |
| 9.13.2    | Member Typedef Documentation . . . . .       | 130 |
| 9.13.2.1  | NF . . . . .                                 | 130 |
| 9.13.3    | Member Enumeration Documentation . . . . .   | 130 |

|           |  |     |
|-----------|--|-----|
| 9.13.3.1  | transformationResult . . . . .                   | 130 |
| 9.13.4    | Constructor & Destructor Documentation . . . . . | 130 |
| 9.13.4.1  | BKLRightNormalForm . . . . .                     | 130 |
| 9.13.4.2  | BKLRightNormalForm . . . . .                     | 130 |
| 9.13.4.3  | BKLRightNormalForm . . . . .                     | 130 |
| 9.13.4.4  | BKLRightNormalForm . . . . .                     | 131 |
| 9.13.5    | Member Function Documentation . . . . .          | 131 |
| 9.13.5.1  | adjustDecomposition . . . . .                    | 131 |
| 9.13.5.2  | getDecomposition . . . . .                       | 131 |
| 9.13.5.3  | getPower . . . . .                               | 131 |
| 9.13.5.4  | getTinyTwistPermutation . . . . .                | 131 |
| 9.13.5.5  | getWord . . . . .                                | 131 |
| 9.13.5.6  | inverse . . . . .                                | 132 |
| 9.13.5.7  | isTrivial . . . . .                              | 132 |
| 9.13.5.8  | multiply . . . . .                               | 132 |
| 9.13.5.9  | operator NF . . . . .                            | 132 |
| 9.13.5.10 | operator!= . . . . .                             | 132 |
| 9.13.5.11 | operator* . . . . .                              | 132 |
| 9.13.5.12 | operator- . . . . .                              | 132 |
| 9.13.5.13 | operator< . . . . .                              | 133 |
| 9.13.5.14 | operator= . . . . .                              | 133 |
| 9.13.5.15 | operator= . . . . .                              | 133 |
| 9.13.5.16 | operator== . . . . .                             | 133 |
| 9.13.5.17 | setDecomposition . . . . .                       | 133 |
| 9.13.5.18 | setPower . . . . .                               | 133 |
| 9.13.5.19 | transform . . . . .                              | 134 |
| 9.13.6    | Member Data Documentation . . . . .              | 134 |
| 9.13.6.1  | theDecomposition . . . . .                       | 134 |
| 9.13.6.2  | theOmegaPower . . . . .                          | 134 |
| 9.13.6.3  | theRank . . . . .                                | 134 |
| 9.14      | BraidDrawPDF Class Reference . . . . .           | 135 |

---

|           |  |     |
|-----------|--|-----|
| 9.14.1    | Detailed Description . . . . .                   | 136 |
| 9.14.2    | Constructor & Destructor Documentation . . . . . | 136 |
| 9.14.2.1  | BraidDrawPDF . . . . .                           | 136 |
| 9.14.2.2  | ~BraidDrawPDF . . . . .                          | 136 |
| 9.14.3    | Member Function Documentation . . . . .          | 136 |
| 9.14.3.1  | draw . . . . .                                   | 136 |
| 9.14.3.2  | drawCompressedBraid . . . . .                    | 136 |
| 9.14.3.3  | drawGenerator . . . . .                          | 137 |
| 9.14.3.4  | drawGrid . . . . .                               | 137 |
| 9.14.3.5  | getPageStrip . . . . .                           | 137 |
| 9.14.3.6  | posInPage . . . . .                              | 137 |
| 9.14.3.7  | save . . . . .                                   | 137 |
| 9.14.3.8  | setSS . . . . .                                  | 138 |
| 9.14.3.9  | stripesInPage . . . . .                          | 138 |
| 9.14.3.10 | stripPosition . . . . .                          | 138 |
| 9.14.3.11 | tableWidth . . . . .                             | 138 |
| 9.14.4    | Member Data Documentation . . . . .              | 138 |
| 9.14.4.1  | betBraids . . . . .                              | 138 |
| 9.14.4.2  | N . . . . .                                      | 138 |
| 9.14.4.3  | pdf_out . . . . .                                | 139 |
| 9.14.4.4  | ss . . . . .                                     | 139 |
| 9.14.4.5  | theLength . . . . .                              | 139 |
| 9.14.4.6  | theTitle . . . . .                               | 139 |
| 9.14.4.7  | useCircle . . . . .                              | 139 |
| 9.15      | BraidGroup Class Reference . . . . .             | 140 |
| 9.15.1    | Detailed Description . . . . .                   | 140 |
| 9.15.2    | Constructor & Destructor Documentation . . . . . | 140 |
| 9.15.2.1  | BraidGroup . . . . .                             | 140 |
| 9.15.3    | Member Function Documentation . . . . .          | 141 |
| 9.15.3.1  | getBraidRelators . . . . .                       | 141 |
| 9.15.3.2  | getRank . . . . .                                | 141 |

|           |  |     |
|-----------|--|-----|
| 9.15.3.3  | twist . . . . .                                      | 141 |
| 9.15.4    | Member Data Documentation . . . . .                  | 141 |
| 9.15.4.1  | theRank . . . . .                                    | 141 |
| 9.16      | BraidNode Struct Reference . . . . .                 | 142 |
| 9.16.1    | Detailed Description . . . . .                       | 143 |
| 9.16.2    | Constructor & Destructor Documentation . . . . .     | 143 |
| 9.16.2.1  | BraidNode . . . . .                                  | 143 |
| 9.16.2.2  | BraidNode . . . . .                                  | 143 |
| 9.16.3    | Member Function Documentation . . . . .              | 144 |
| 9.16.3.1  | isHandle . . . . .                                   | 144 |
| 9.16.3.2  | removeHandle . . . . .                               | 144 |
| 9.16.3.3  | removeNodeTerminalClosure . . . . .                  | 144 |
| 9.16.3.4  | removeTerminalNode . . . . .                         | 144 |
| 9.16.4    | Member Data Documentation . . . . .                  | 144 |
| 9.16.4.1  | ahead . . . . .                                      | 144 |
| 9.16.4.2  | back . . . . .                                       | 144 |
| 9.16.4.3  | back_left . . . . .                                  | 144 |
| 9.16.4.4  | back_right . . . . .                                 | 144 |
| 9.16.4.5  | left . . . . .                                       | 145 |
| 9.16.4.6  | link . . . . .                                       | 145 |
| 9.16.4.7  | marked . . . . .                                     | 145 |
| 9.16.4.8  | right . . . . .                                      | 145 |
| 9.16.4.9  | theNumber . . . . .                                  | 145 |
| 9.16.4.10 | type . . . . .                                       | 145 |
| 9.16.4.11 | weight . . . . .                                     | 145 |
| 9.17      | BSets Class Reference . . . . .                      | 147 |
| 9.17.1    | Detailed Description . . . . .                       | 147 |
| 9.17.2    | Member Function Documentation . . . . .              | 147 |
| 9.17.2.1  | generateEqual . . . . .                              | 147 |
| 9.17.2.2  | generateRandom . . . . .                             | 148 |
| 9.17.3    | Friends And Related Function Documentation . . . . . | 148 |



---

|          |  |     |
|----------|--|-----|
| 9.17.3.1 | operator<<                                   | 148 |
| 9.17.4   | Member Data Documentation                    | 148 |
| 9.17.4.1 | BL   | 148 |
| 9.17.4.2 | BR   | 148 |
| 9.18     | BalancedTree< Obj >::BTNode Struct Reference | 149 |
| 9.18.1   | Detailed Description                         | 149 |
| 9.18.2   | Constructor & Destructor Documentation       | 149 |
| 9.18.2.1 | BTNode                                       | 149 |
| 9.18.2.2 | BTNode                                       | 149 |
| 9.18.2.3 | ~BTNode                                      | 149 |
| 9.18.3   | Member Data Documentation                    | 150 |
| 9.18.3.1 | height1                                      | 150 |
| 9.18.3.2 | height2                                      | 150 |
| 9.18.3.3 | subtree1                                     | 150 |
| 9.18.3.4 | subtree2                                     | 150 |
| 9.18.3.5 | theObject                                    | 150 |
| 9.18.3.6 | weight1                                      | 151 |
| 9.18.3.7 | weight2                                      | 151 |
| 9.19     | CImage Class Reference                       | 152 |
| 9.19.1   | Detailed Description                         | 153 |
| 9.19.2   | Constructor & Destructor Documentation       | 153 |
| 9.19.2.1 | CImage                                       | 153 |
| 9.19.2.2 | CImage                                       | 153 |
| 9.19.2.3 | CImage                                       | 153 |
| 9.19.2.4 | CImage                                       | 153 |
| 9.19.2.5 | CImage                                       | 153 |
| 9.19.3   | Member Function Documentation                | 154 |
| 9.19.3.1 | getBlueImage                                 | 154 |
| 9.19.3.2 | getBluePixel                                 | 154 |
| 9.19.3.3 | getBluePixel                                 | 154 |
| 9.19.3.4 | getGreenImage                                | 154 |

|           |  |     |
|-----------|--|-----|
| 9.19.3.5  | getGreenPixel . . . . .                              | 154 |
| 9.19.3.6  | getGreenPixel . . . . .                              | 154 |
| 9.19.3.7  | getRedImage . . . . .                                | 155 |
| 9.19.3.8  | getRedPixel . . . . .                                | 155 |
| 9.19.3.9  | getRedPixel . . . . .                                | 155 |
| 9.19.3.10 | getType . . . . .                                    | 155 |
| 9.19.3.11 | operator= . . . . .                                  | 156 |
| 9.19.3.12 | printOnPPM . . . . .                                 | 156 |
| 9.19.3.13 | readFromPPM . . . . .                                | 156 |
| 9.19.3.14 | saveTo . . . . .                                     | 156 |
| 9.19.3.15 | setBluePixel . . . . .                               | 156 |
| 9.19.3.16 | setBluePixel . . . . .                               | 156 |
| 9.19.3.17 | setBluePixelCliped . . . . .                         | 156 |
| 9.19.3.18 | setBluePixelCliped . . . . .                         | 156 |
| 9.19.3.19 | setGreenPixel . . . . .                              | 156 |
| 9.19.3.20 | setGreenPixel . . . . .                              | 157 |
| 9.19.3.21 | setGreenPixelCliped . . . . .                        | 157 |
| 9.19.3.22 | setGreenPixelCliped . . . . .                        | 157 |
| 9.19.3.23 | setRedPixel . . . . .                                | 157 |
| 9.19.3.24 | setRedPixel . . . . .                                | 157 |
| 9.19.3.25 | setRedPixelCliped . . . . .                          | 157 |
| 9.19.3.26 | setRedPixelCliped . . . . .                          | 157 |
| 9.19.4    | Friends And Related Function Documentation . . . . . | 158 |
| 9.19.4.1  | CLUTImage . . . . .                                  | 158 |
| 9.19.5    | Member Data Documentation . . . . .                  | 158 |
| 9.19.5.1  | blueImage . . . . .                                  | 158 |
| 9.19.5.2  | greenImage . . . . .                                 | 158 |
| 9.19.5.3  | redImage . . . . .                                   | 158 |
| 9.20      | CLUTImage Class Reference . . . . .                  | 159 |
| 9.20.1    | Detailed Description . . . . .                       | 159 |
| 9.20.2    | Constructor & Destructor Documentation . . . . .     | 159 |

|          |   |     |
|----------|---|-----|
| 9.20.2.1 | CLUTImage . . . . .   | 159 |
| 9.20.2.2 | CLUTImage . . . . .   | 159 |
| 9.20.2.3 | CLUTImage . . . . .   | 160 |
| 9.20.2.4 | CLUTImage . . . . .   | 160 |
| 9.20.3   | Member Function Documentation . . . . .                       | 160 |
| 9.20.3.1 | getBluePixel . . . . .  | 160 |
| 9.20.3.2 | getBluePixel . . . . .  | 160 |
| 9.20.3.3 | getGreenPixel . . . . .                                       | 160 |
| 9.20.3.4 | getGreenPixel . . . . .                                       | 160 |
| 9.20.3.5 | getRedPixel . . . . .   | 161 |
| 9.20.3.6 | getRedPixel . . . . .   | 161 |
| 9.21     | PC::Col Struct Reference . . . . .                            | 162 |
| 9.21.1   | Detailed Description . . . . .                                | 162 |
| 9.21.2   | Constructor & Destructor Documentation . . . . .              | 162 |
| 9.21.2.1 | Col . . . . .   | 162 |
| 9.21.3   | Member Data Documentation . . . . .                           | 162 |
| 9.21.3.1 | colid . . . . .   | 162 |
| 9.21.3.2 | UNDEFINED . . . . .   | 163 |
| 9.22     | PC::PowerCircuitCompMatrix::ColHdr Struct Reference . . . . . | 164 |
| 9.22.1   | Detailed Description . . . . .                                | 164 |
| 9.22.2   | Member Data Documentation . . . . .                           | 164 |
| 9.22.2.1 | markings . . . . .  | 164 |
| 9.23     | CommDivider Struct Reference . . . . .                        | 165 |
| 9.23.1   | Detailed Description . . . . .                                | 165 |
| 9.23.2   | Friends And Related Function Documentation . . . . .          | 165 |
| 9.23.2.1 | operator<< . . . . .  | 165 |
| 9.23.3   | Member Data Documentation . . . . .                           | 165 |
| 9.23.3.1 | begin . . . . .   | 165 |
| 9.23.3.2 | end . . . . .   | 165 |
| 9.23.3.3 | gen . . . . .   | 166 |
| 9.23.3.4 | length . . . . .  | 166 |

|   |     |
|---|-----|
| 9.24 compMaps Struct Reference . . . . .                    | 167 |
| 9.24.1 Detailed Description . . . . .                       | 167 |
| 9.24.2 Member Function Documentation . . . . .              | 167 |
| 9.24.2.1 operator() . . . . .                               | 167 |
| 9.25 ConfigFile Class Reference . . . . .                   | 168 |
| 9.25.1 Detailed Description . . . . .                       | 169 |
| 9.25.2 Constructor & Destructor Documentation . . . . .     | 169 |
| 9.25.2.1 ConfigFile . . . . .                               | 169 |
| 9.25.2.2 ConfigFile . . . . .                               | 169 |
| 9.25.3 Member Function Documentation . . . . .              | 169 |
| 9.25.3.1 getValue . . . . .                                 | 169 |
| 9.25.3.2 ltrim . . . . .                                    | 170 |
| 9.25.3.3 printOn . . . . .                                  | 170 |
| 9.25.3.4 readFrom . . . . .                                 | 170 |
| 9.25.3.5 rtrim . . . . .                                    | 170 |
| 9.25.3.6 setVariable . . . . .                              | 170 |
| 9.25.3.7 trim . . . . .                                     | 170 |
| 9.25.4 Friends And Related Function Documentation . . . . . | 170 |
| 9.25.4.1 operator<< . . . . .                               | 170 |
| 9.25.5 Member Data Documentation . . . . .                  | 171 |
| 9.25.5.1 parameters . . . . .                               | 171 |
| 9.26 ConstWordIterator Class Reference . . . . .            | 172 |
| 9.26.1 Detailed Description . . . . .                       | 173 |
| 9.26.2 Member Typedef Documentation . . . . .               | 173 |
| 9.26.2.1 PII . . . . .                                      | 173 |
| 9.26.2.2 PII . . . . .                                      | 173 |
| 9.26.3 Constructor & Destructor Documentation . . . . .     | 174 |
| 9.26.3.1 ConstWordIterator . . . . .                        | 174 |
| 9.26.3.2 ConstWordIterator . . . . .                        | 174 |
| 9.26.3.3 ConstWordIterator . . . . .                        | 174 |
| 9.26.3.4 ConstWordIterator . . . . .                        | 174 |

---

|           |  |     |
|-----------|--|-----|
| 9.26.3.5  | ConstWordIterator . . . . .                          | 174 |
| 9.26.3.6  | ConstWordIterator . . . . .                          | 174 |
| 9.26.4    | Member Function Documentation . . . . .              | 174 |
| 9.26.4.1  | operator!= . . . . .                                 | 174 |
| 9.26.4.2  | operator!= . . . . .                                 | 174 |
| 9.26.4.3  | operator* . . . . .                                  | 174 |
| 9.26.4.4  | operator* . . . . .                                  | 174 |
| 9.26.4.5  | operator++ . . . . .                                 | 174 |
| 9.26.4.6  | operator++ . . . . .                                 | 174 |
| 9.26.4.7  | operator++ . . . . .                                 | 174 |
| 9.26.4.8  | operator++ . . . . .                                 | 174 |
| 9.26.4.9  | operator-- . . . . .                                 | 174 |
| 9.26.4.10 | operator-- . . . . .                                 | 174 |
| 9.26.4.11 | operator-- . . . . .                                 | 174 |
| 9.26.4.12 | operator-- . . . . .                                 | 174 |
| 9.26.4.13 | operator= . . . . .                                  | 174 |
| 9.26.4.14 | operator= . . . . .                                  | 174 |
| 9.26.4.15 | operator== . . . . .                                 | 174 |
| 9.26.4.16 | operator== . . . . .                                 | 174 |
| 9.26.5    | Friends And Related Function Documentation . . . . . | 174 |
| 9.26.5.1  | WordRep . . . . .                                    | 174 |
| 9.26.6    | Member Data Documentation . . . . .                  | 175 |
| 9.26.6.1  | theIterator . . . . .                                | 175 |
| 9.26.6.2  | theIterator . . . . .                                | 175 |
| 9.26.6.3  | theList . . . . .                                    | 175 |
| 9.26.6.4  | theOffset . . . . .                                  | 175 |
| 9.26.6.5  | theWord . . . . .                                    | 175 |
| 9.27      | CutVertices Class Reference . . . . .                | 176 |
| 9.27.1    | Detailed Description . . . . .                       | 177 |
| 9.27.2    | Constructor & Destructor Documentation . . . . .     | 177 |
| 9.27.2.1  | CutVertices . . . . .                                | 177 |

|          |  |     |
|----------|--|-----|
| 9.27.2.2 | ~CutVertices                           | 178 |
| 9.27.3   | Member Function Documentation          | 178 |
| 9.27.3.1 | ArticulationPoints                     | 178 |
| 9.27.3.2 | compute                                | 178 |
| 9.27.3.3 | computeBruteForce                      | 178 |
| 9.27.3.4 | DepthFirstSearch                       | 178 |
| 9.27.3.5 | getCutVertices                         | 178 |
| 9.27.3.6 | init                                   | 179 |
| 9.27.3.7 | numberOfComponents                     | 179 |
| 9.27.3.8 | RDFS_Compute_Low                       | 179 |
| 9.27.3.9 | RecursiveDepthFirstSearch              | 179 |
| 9.27.4   | Member Data Documentation              | 179 |
| 9.27.4.1 | articulation_point                     | 179 |
| 9.27.4.2 | discover                               | 179 |
| 9.27.4.3 | Low                                    | 179 |
| 9.27.4.4 | N                                      | 180 |
| 9.27.4.5 | pred                                   | 180 |
| 9.27.4.6 | theGraph                               | 180 |
| 9.27.4.7 | time                                   | 180 |
| 9.27.4.8 | visit                                  | 180 |
| 9.28     | DCBraidReduction Class Reference       | 181 |
| 9.28.1   | Detailed Description                   | 181 |
| 9.28.2   | Constructor & Destructor Documentation | 181 |
| 9.28.2.1 | DCBraidReduction                       | 181 |
| 9.28.3   | Member Function Documentation          | 181 |
| 9.28.3.1 | shorten                                | 181 |
| 9.28.4   | Member Data Documentation              | 181 |
| 9.28.4.1 | N                                      | 181 |
| 9.28.4.2 | thePartition                           | 182 |
| 9.28.4.3 | thePerturb                             | 182 |
| 9.29     | DDL Struct Reference                   | 183 |

---

|          |  |     |
|----------|--|-----|
| 9.29.1   | Detailed Description . . . . .                   | 183 |
| 9.29.2   | Constructor & Destructor Documentation . . . . . | 183 |
| 9.29.2.1 | DDL . . . . .                                    | 183 |
| 9.29.2.2 | DDL . . . . .                                    | 184 |
| 9.29.2.3 | ~DDL . . . . .                                   | 184 |
| 9.29.3   | Member Function Documentation . . . . .          | 184 |
| 9.29.3.1 | append . . . . .                                 | 184 |
| 9.29.3.2 | delcur . . . . .                                 | 184 |
| 9.29.3.3 | delend . . . . .                                 | 184 |
| 9.29.3.4 | insertA . . . . .                                | 184 |
| 9.29.3.5 | insertA . . . . .                                | 184 |
| 9.29.3.6 | toWord . . . . .                                 | 184 |
| 9.29.4   | Member Data Documentation . . . . .              | 184 |
| 9.29.4.1 | first . . . . .                                  | 184 |
| 9.29.4.2 | last . . . . .                                   | 184 |
| 9.29.4.3 | len . . . . .                                    | 184 |
| 9.30     | DDLNode Struct Reference . . . . .               | 185 |
| 9.30.1   | Detailed Description . . . . .                   | 185 |
| 9.30.2   | Constructor & Destructor Documentation . . . . . | 185 |
| 9.30.2.1 | DDLNode . . . . .                                | 185 |
| 9.30.3   | Member Data Documentation . . . . .              | 185 |
| 9.30.3.1 | left . . . . .                                   | 185 |
| 9.30.3.2 | right . . . . .                                  | 185 |
| 9.30.3.3 | value . . . . .                                  | 185 |
| 9.31     | DehornoyForm Class Reference . . . . .           | 187 |
| 9.31.1   | Detailed Description . . . . .                   | 187 |
| 9.31.2   | Constructor & Destructor Documentation . . . . . | 188 |
| 9.31.2.1 | DehornoyForm . . . . .                           | 188 |
| 9.31.2.2 | DehornoyForm . . . . .                           | 188 |
| 9.31.3   | Member Function Documentation . . . . .          | 188 |
| 9.31.3.1 | computeDehornoyForm . . . . .                    | 188 |

|          |  |     |
|----------|--|-----|
| 9.31.3.2 | computeDehornoyForm . . . . .                        | 188 |
| 9.31.3.3 | getDehornoyForm . . . . .                            | 188 |
| 9.31.3.4 | getDehornoyForm . . . . .                            | 188 |
| 9.31.4   | Member Data Documentation . . . . .                  | 188 |
| 9.31.4.1 | theDehornoyForm . . . . .                            | 188 |
| 9.31.4.2 | theIndex . . . . .                                   | 189 |
| 9.31.4.3 | theLinkedStructure . . . . .                         | 189 |
| 9.32     | dump< T > Class Template Reference . . . . .         | 190 |
| 9.32.1   | Detailed Description . . . . .                       | 190 |
| 9.32.2   | Member Typedef Documentation . . . . .               | 190 |
| 9.32.2.1 | const_iterator . . . . .                             | 190 |
| 9.32.3   | Constructor & Destructor Documentation . . . . .     | 190 |
| 9.32.3.1 | dump . . . . .                                       | 190 |
| 9.32.4   | Member Function Documentation . . . . .              | 191 |
| 9.32.4.1 | printOn . . . . .                                    | 191 |
| 9.32.5   | Friends And Related Function Documentation . . . . . | 191 |
| 9.32.5.1 | operator<< . . . . .                                 | 191 |
| 9.32.6   | Member Data Documentation . . . . .                  | 191 |
| 9.32.6.1 | theCont . . . . .                                    | 191 |
| 9.33     | Dump Class Reference . . . . .                       | 192 |
| 9.33.1   | Detailed Description . . . . .                       | 192 |
| 9.33.2   | Member Data Documentation . . . . .                  | 192 |
| 9.33.2.1 | dump_out . . . . .                                   | 192 |
| 9.34     | EditingDistance Class Reference . . . . .            | 193 |
| 9.34.1   | Detailed Description . . . . .                       | 193 |
| 9.34.2   | Constructor & Destructor Documentation . . . . .     | 193 |
| 9.34.2.1 | EditingDistance . . . . .                            | 193 |
| 9.34.3   | Member Function Documentation . . . . .              | 193 |
| 9.34.3.1 | measure . . . . .                                    | 193 |
| 9.35     | Equation Class Reference . . . . .                   | 195 |
| 9.35.1   | Detailed Description . . . . .                       | 196 |



|          |  |     |
|----------|--|-----|
| 9.35.2   | Constructor & Destructor Documentation . . . . .     | 196 |
| 9.35.2.1 | Equation . . . . .                                   | 196 |
| 9.35.3   | Member Function Documentation . . . . .              | 196 |
| 9.35.3.1 | getTheEquation . . . . .                             | 196 |
| 9.35.3.2 | getTheNumberOfGenerators . . . . .                   | 196 |
| 9.35.3.3 | getTheNumberOfVariables . . . . .                    | 196 |
| 9.35.3.4 | isGenerator . . . . .                                | 197 |
| 9.35.3.5 | isQuadratic . . . . .                                | 197 |
| 9.35.3.6 | isVariable . . . . .                                 | 197 |
| 9.35.3.7 | randomQuadraticEquation . . . . .                    | 197 |
| 9.35.3.8 | trivialSolution . . . . .                            | 197 |
| 9.35.4   | Friends And Related Function Documentation . . . . . | 197 |
| 9.35.4.1 | operator<< . . . . .                                 | 197 |
| 9.35.5   | Member Data Documentation . . . . .                  | 197 |
| 9.35.5.1 | theEquation . . . . .                                | 197 |
| 9.35.5.2 | theNumberOfGenerators . . . . .                      | 198 |
| 9.35.5.3 | theNumberOfVariables . . . . .                       | 198 |
| 9.36     | FiniteAlphabet Class Reference . . . . .             | 199 |
| 9.36.1   | Detailed Description . . . . .                       | 199 |
| 9.36.2   | Constructor & Destructor Documentation . . . . .     | 199 |
| 9.36.2.1 | FiniteAlphabet . . . . .                             | 199 |
| 9.37     | FoldDetails Struct Reference . . . . .               | 200 |
| 9.37.1   | Detailed Description . . . . .                       | 200 |
| 9.37.2   | Constructor & Destructor Documentation . . . . .     | 200 |
| 9.37.2.1 | FoldDetails . . . . .                                | 200 |
| 9.37.3   | Member Data Documentation . . . . .                  | 200 |
| 9.37.3.1 | base . . . . .                                       | 200 |
| 9.37.3.2 | label . . . . .                                      | 200 |
| 9.37.3.3 | state1 . . . . .                                     | 200 |
| 9.37.3.4 | state2 . . . . .                                     | 201 |
| 9.37.3.5 | state_num . . . . .                                  | 201 |

|           |   |     |
|-----------|---|-----|
| 9.38      | Graphs::FoldDetails< VertexType, EdgeType > Struct Template Reference . . . . . | 202 |
| 9.38.1    | Detailed Description . . . . .  | 202 |
| 9.38.2    | Constructor & Destructor Documentation . . . . .                                | 202 |
| 9.38.2.1  | FoldDetails . . . . .   | 202 |
| 9.38.3    | Member Data Documentation . . . . .   | 203 |
| 9.38.3.1  | theEdge1 . . . . .  | 203 |
| 9.38.3.2  | theEdge2 . . . . .  | 203 |
| 9.38.3.3  | theOrigin . . . . .   | 203 |
| 9.38.3.4  | theVertex1 . . . . .  | 203 |
| 9.38.3.5  | theVertex2 . . . . .  | 203 |
| 9.39      | FPGroup Class Reference . . . . .   | 204 |
| 9.39.1    | Detailed Description . . . . .  | 205 |
| 9.39.2    | Constructor & Destructor Documentation . . . . .                                | 205 |
| 9.39.2.1  | FPGroup . . . . .   | 205 |
| 9.39.2.2  | FPGroup . . . . .   | 205 |
| 9.39.2.3  | FPGroup . . . . .   | 205 |
| 9.39.2.4  | FPGroup . . . . .   | 205 |
| 9.39.3    | Member Function Documentation . . . . .   | 205 |
| 9.39.3.1  | getAlphabet . . . . .   | 205 |
| 9.39.3.2  | getGeneratorsNames . . . . .  | 205 |
| 9.39.3.3  | initializeGenNames . . . . .  | 206 |
| 9.39.3.4  | numberOfGenerators . . . . .  | 206 |
| 9.39.3.5  | randomEqWord_Baltimore . . . . .  | 206 |
| 9.39.3.6  | randomIdentity_Baltimore . . . . .  | 206 |
| 9.39.3.7  | randomIdentity_Classic . . . . .  | 206 |
| 9.39.3.8  | randomIdentity_Stack . . . . .  | 206 |
| 9.39.3.9  | relators . . . . .  | 207 |
| 9.39.3.10 | triangulatePresentation . . . . .   | 207 |
| 9.39.4    | Friends And Related Function Documentation . . . . .                            | 207 |
| 9.39.4.1  | operator<< . . . . .  | 207 |

---

|          |  |     |
|----------|--|-----|
| 9.39.4.2 | operator>>                                 | 207 |
| 9.39.5   | Member Data Documentation                  | 207 |
| 9.39.5.1 | numOfGenerators                            | 207 |
| 9.39.5.2 | theAlphabet                                | 207 |
| 9.39.5.3 | theRelators                                | 207 |
| 9.39.5.4 | useDefaultAlphabet                         | 208 |
| 9.40     | FRDFIT Class Reference                     | 209 |
| 9.40.1   | Detailed Description                       | 209 |
| 9.40.2   | Constructor & Destructor Documentation     | 209 |
| 9.40.2.1 | FRDFIT                                     | 209 |
| 9.40.3   | Member Function Documentation              | 209 |
| 9.40.3.1 | compute                                    | 209 |
| 9.40.3.2 | pp   | 209 |
| 9.40.4   | Member Data Documentation                  | 209 |
| 9.40.4.1 | theGroupGens                               | 209 |
| 9.41     | FreeGroup Class Reference                  | 211 |
| 9.41.1   | Detailed Description                       | 211 |
| 9.41.2   | Constructor & Destructor Documentation     | 212 |
| 9.41.2.1 | FreeGroup                                  | 212 |
| 9.41.2.2 | FreeGroup                                  | 212 |
| 9.41.3   | Member Function Documentation              | 212 |
| 9.41.3.1 | doesContain                                | 212 |
| 9.41.3.2 | getAlphabet                                | 212 |
| 9.41.3.3 | isAlmostPrimitive                          | 212 |
| 9.41.3.4 | isPrimitive                                | 212 |
| 9.41.4   | Friends And Related Function Documentation | 212 |
| 9.41.4.1 | operator<<                                 | 212 |
| 9.41.4.2 | operator>>                                 | 212 |
| 9.41.5   | Member Data Documentation                  | 213 |
| 9.41.5.1 | theAlphabet                                | 213 |
| 9.41.5.2 | theRank                                    | 213 |

|           |   |     |
|-----------|---|-----|
| 9.41.5.3  | useDefaultAlphabet . . . . .                            | 213 |
| 9.42      | FreeMetabelianGroupAlgorithms Class Reference . . . . . | 214 |
| 9.42.1    | Detailed Description . . . . .                          | 214 |
| 9.42.2    | Constructor & Destructor Documentation . . . . .        | 214 |
| 9.42.2.1  | FreeMetabelianGroupAlgorithms . . . . .                 | 214 |
| 9.42.3    | Member Function Documentation . . . . .                 | 215 |
| 9.42.3.1  | conjugate . . . . .                                     | 215 |
| 9.42.3.2  | getWordFromEdgeMap . . . . .                            | 215 |
| 9.42.3.3  | trivial . . . . .                                       | 215 |
| 9.43      | FSA Class Reference . . . . .                           | 216 |
| 9.43.1    | Detailed Description . . . . .                          | 217 |
| 9.43.2    | Member Typedef Documentation . . . . .                  | 217 |
| 9.43.2.1  | edge_type . . . . .                                     | 217 |
| 9.43.2.2  | state_type . . . . .                                    | 217 |
| 9.43.3    | Constructor & Destructor Documentation . . . . .        | 217 |
| 9.43.3.1  | FSA . . . . .   | 217 |
| 9.43.3.2  | FSA . . . . .   | 217 |
| 9.43.4    | Member Function Documentation . . . . .                 | 217 |
| 9.43.4.1  | addFSA . . . . .  | 217 |
| 9.43.4.2  | addLoop . . . . .                                       | 218 |
| 9.43.4.3  | addRay . . . . .  | 218 |
| 9.43.4.4  | deterministic . . . . .                                 | 218 |
| 9.43.4.5  | eraseEdge . . . . .                                     | 218 |
| 9.43.4.6  | eraseState . . . . .                                    | 218 |
| 9.43.4.7  | fold . . . . .  | 218 |
| 9.43.4.8  | getInitStates . . . . .                                 | 218 |
| 9.43.4.9  | getStates . . . . .                                     | 219 |
| 9.43.4.10 | getStates . . . . .                                     | 219 |
| 9.43.4.11 | getTermStates . . . . .                                 | 219 |
| 9.43.4.12 | isDeterministic . . . . .                               | 219 |
| 9.43.4.13 | liftup . . . . .  | 219 |

---

|           |  |     |
|-----------|--|-----|
| 9.43.4.14 | makeInitial . . . . .                            | 219 |
| 9.43.4.15 | makeNonInitial . . . . .                         | 219 |
| 9.43.4.16 | makeNonTerminal . . . . .                        | 220 |
| 9.43.4.17 | makeTerminal . . . . .                           | 220 |
| 9.43.4.18 | newEdge . . . . .                                | 220 |
| 9.43.4.19 | newState . . . . .                               | 220 |
| 9.43.4.20 | operator* . . . . .                              | 220 |
| 9.43.4.21 | operator== . . . . .                             | 220 |
| 9.43.4.22 | pinch . . . . .                                  | 220 |
| 9.43.4.23 | unfold . . . . .                                 | 220 |
| 9.44      | FSAEdge Struct Reference . . . . .               | 222 |
| 9.44.1    | Detailed Description . . . . .                   | 222 |
| 9.44.2    | Constructor & Destructor Documentation . . . . . | 222 |
| 9.44.2.1  | FSAEdge . . . . .                                | 222 |
| 9.44.2.2  | FSAEdge . . . . .                                | 222 |
| 9.44.3    | Member Function Documentation . . . . .          | 222 |
| 9.44.3.1  | operator< . . . . .                              | 222 |
| 9.44.3.2  | operator== . . . . .                             | 223 |
| 9.44.4    | Member Data Documentation . . . . .              | 223 |
| 9.44.4.1  | label . . . . .                                  | 223 |
| 9.44.4.2  | target . . . . .                                 | 223 |
| 9.45      | FSARep Class Reference . . . . .                 | 224 |
| 9.45.1    | Detailed Description . . . . .                   | 226 |
| 9.45.2    | Member Typedef Documentation . . . . .           | 226 |
| 9.45.2.1  | edge_type . . . . .                              | 226 |
| 9.45.3    | Constructor & Destructor Documentation . . . . . | 226 |
| 9.45.3.1  | FSARep . . . . .                                 | 226 |
| 9.45.4    | Member Function Documentation . . . . .          | 226 |
| 9.45.4.1  | addFSA . . . . .                                 | 226 |
| 9.45.4.2  | addLoop . . . . .                                | 226 |
| 9.45.4.3  | addRay . . . . .                                 | 226 |

|           |  |     |
|-----------|--|-----|
| 9.45.4.4  | clone . . . . .                                      | 227 |
| 9.45.4.5  | eraseEdge . . . . .                                  | 227 |
| 9.45.4.6  | eraseState . . . . .                                 | 227 |
| 9.45.4.7  | fold . . . . .                                       | 227 |
| 9.45.4.8  | getInitStates . . . . .                              | 227 |
| 9.45.4.9  | getStates . . . . .                                  | 227 |
| 9.45.4.10 | getStates . . . . .                                  | 227 |
| 9.45.4.11 | getTermStates . . . . .                              | 228 |
| 9.45.4.12 | liftup . . . . .                                     | 228 |
| 9.45.4.13 | makeInitial . . . . .                                | 228 |
| 9.45.4.14 | makeNonInitial . . . . .                             | 228 |
| 9.45.4.15 | makeNonTerminal . . . . .                            | 228 |
| 9.45.4.16 | makeTerminal . . . . .                               | 228 |
| 9.45.4.17 | newEdge . . . . .                                    | 228 |
| 9.45.4.18 | newState . . . . .                                   | 229 |
| 9.45.4.19 | pinch . . . . .                                      | 229 |
| 9.45.4.20 | unfold . . . . .                                     | 229 |
| 9.45.5    | Friends And Related Function Documentation . . . . . | 229 |
| 9.45.5.1  | FSA . . . . .  | 229 |
| 9.45.6    | Member Data Documentation . . . . .                  | 229 |
| 9.45.6.1  | initStates . . . . .                                 | 229 |
| 9.45.6.2  | maxState . . . . .                                   | 229 |
| 9.45.6.3  | termStates . . . . .                                 | 229 |
| 9.45.6.4  | theStates . . . . .                                  | 229 |
| 9.46      | FSAStruct Reference . . . . .                        | 231 |
| 9.46.1    | Detailed Description . . . . .                       | 231 |
| 9.46.2    | Member Typedef Documentation . . . . .               | 231 |
| 9.46.2.1  | edge_type . . . . .                                  | 231 |
| 9.46.3    | Constructor & Destructor Documentation . . . . .     | 231 |
| 9.46.3.1  | FSAStruct . . . . .                                  | 231 |
| 9.46.3.2  | FSAStruct . . . . .                                  | 231 |

|          |  |     |
|----------|--|-----|
| 9.46.4   | Member Function Documentation . . . . .  | 232 |
| 9.46.4.1 | operator< . . . . .  | 232 |
| 9.46.4.2 | operator== . . . . .   | 232 |
| 9.46.5   | Member Data Documentation . . . . .  | 232 |
| 9.46.5.1 | in . . . . .   | 232 |
| 9.46.5.2 | out . . . . .  | 232 |
| 9.46.5.3 | theState . . . . .   | 232 |
| 9.47     | Graph Class Reference . . . . .  | 233 |
| 9.47.1   | Detailed Description . . . . .   | 233 |
| 9.47.2   | Member Typedef Documentation . . . . .   | 233 |
| 9.47.2.1 | edge_type . . . . .  | 233 |
| 9.47.2.2 | state_type . . . . .   | 233 |
| 9.47.3   | Constructor & Destructor Documentation . . . . .                                   | 234 |
| 9.47.3.1 | Graph . . . . .  | 234 |
| 9.47.4   | Member Function Documentation . . . . .  | 234 |
| 9.47.4.1 | clear . . . . .  | 234 |
| 9.47.4.2 | getStates . . . . .  | 234 |
| 9.47.4.3 | newEdge . . . . .  | 234 |
| 9.47.4.4 | newState . . . . .   | 234 |
| 9.48     | Graphs::GraphConcept< VertexType, EdgeType > Class Template<br>Reference . . . . . | 235 |
| 9.48.1   | Detailed Description . . . . .   | 236 |
| 9.48.2   | Member Typedef Documentation . . . . .   | 236 |
| 9.48.2.1 | edge_type . . . . .  | 236 |
| 9.48.2.2 | presentation_type . . . . .  | 236 |
| 9.48.2.3 | vertex_type . . . . .  | 237 |
| 9.48.3   | Constructor & Destructor Documentation . . . . .                                   | 237 |
| 9.48.3.1 | GraphConcept . . . . .   | 237 |
| 9.48.4   | Member Function Documentation . . . . .  | 237 |
| 9.48.4.1 | clear . . . . .  | 237 |
| 9.48.4.2 | eraseVertex . . . . .  | 237 |

|           |   |     |
|-----------|---|-----|
| 9.48.4.3  | getVertices . . . . .   | 237 |
| 9.48.4.4  | newEdge . . . . .   | 238 |
| 9.48.4.5  | newVertex . . . . .   | 238 |
| 9.48.4.6  | newVertex . . . . .   | 238 |
| 9.48.4.7  | pinch . . . . .   | 238 |
| 9.48.4.8  | replaceVertex . . . . .   | 239 |
| 9.49      | Graphs::GraphConceptRep< VertexType, EdgeType > Class Template<br>Reference . . . . . | 240 |
| 9.49.1    | Detailed Description . . . . .  | 241 |
| 9.49.2    | Member Typedef Documentation . . . . .  | 241 |
| 9.49.2.1  | edge_type . . . . .   | 241 |
| 9.49.2.2  | vertex_type . . . . .   | 242 |
| 9.49.3    | Constructor & Destructor Documentation . . . . .                                      | 242 |
| 9.49.3.1  | GraphConceptRep . . . . .   | 242 |
| 9.49.4    | Member Function Documentation . . . . .   | 242 |
| 9.49.4.1  | _newVertex . . . . .  | 242 |
| 9.49.4.2  | clear . . . . .   | 242 |
| 9.49.4.3  | clone . . . . .   | 243 |
| 9.49.4.4  | eraseVertex . . . . .   | 243 |
| 9.49.4.5  | eraseVertex . . . . .   | 243 |
| 9.49.4.6  | getVertices . . . . .   | 243 |
| 9.49.4.7  | newEdge . . . . .   | 243 |
| 9.49.4.8  | newVertex . . . . .   | 244 |
| 9.49.4.9  | newVertex . . . . .   | 244 |
| 9.49.4.10 | pinch . . . . .   | 244 |
| 9.49.4.11 | replaceVertex . . . . .   | 244 |
| 9.49.5    | Friends And Related Function Documentation . . . . .                                  | 245 |
| 9.49.5.1  | GraphConcept< vertex_type, edge_type > . . . . .                                      | 245 |
| 9.49.6    | Member Data Documentation . . . . .   | 245 |
| 9.49.6.1  | maxVertex . . . . .   | 245 |
| 9.49.6.2  | theVertices . . . . .   | 245 |



|          |  |     |
|----------|--|-----|
| 9.50     | GraphDrawingAttributes Class Reference . . . . . | 246 |
| 9.50.1   | Detailed Description . . . . .                   | 247 |
| 9.50.2   | Member Typedef Documentation . . . . .           | 247 |
| 9.50.2.1 | COLOR . . . . .                                  | 247 |
| 9.50.3   | Member Enumeration Documentation . . . . .       | 247 |
| 9.50.3.1 | NODESHAPE . . . . .                              | 247 |
| 9.50.4   | Constructor & Destructor Documentation . . . . . | 248 |
| 9.50.4.1 | GraphDrawingAttributes . . . . .                 | 248 |
| 9.50.5   | Member Function Documentation . . . . .          | 248 |
| 9.50.5.1 | getNodeColor . . . . .                           | 248 |
| 9.50.5.2 | getNodeShape . . . . .                           | 249 |
| 9.50.5.3 | getNodeShapeNames . . . . .                      | 249 |
| 9.50.5.4 | initializeNodeShapeNames . . . . .               | 249 |
| 9.50.5.5 | setDefaultNodeShape . . . . .                    | 249 |
| 9.50.5.6 | setNodeColor . . . . .                           | 249 |
| 9.50.5.7 | setNodeShape . . . . .                           | 249 |
| 9.50.6   | Member Data Documentation . . . . .              | 250 |
| 9.50.6.1 | nodeColor . . . . .                              | 250 |
| 9.50.6.2 | nodeShape . . . . .                              | 250 |
| 9.50.6.3 | nodeShapeNames . . . . .                         | 250 |
| 9.50.6.4 | theDefaultNodeColor . . . . .                    | 250 |
| 9.50.6.5 | theDefaultNodeShape . . . . .                    | 250 |
| 9.51     | GraphEdge Struct Reference . . . . .             | 251 |
| 9.51.1   | Detailed Description . . . . .                   | 252 |
| 9.51.2   | Constructor & Destructor Documentation . . . . . | 252 |
| 9.51.2.1 | GraphEdge . . . . .                              | 252 |
| 9.51.2.2 | GraphEdge . . . . .                              | 252 |
| 9.51.2.3 | GraphEdge . . . . .                              | 252 |
| 9.51.2.4 | GraphEdge . . . . .                              | 252 |
| 9.51.3   | Member Function Documentation . . . . .          | 253 |
| 9.51.3.1 | inverse . . . . .                                | 253 |

|          |  |     |
|----------|--|-----|
| 9.51.3.2 | operator!=                                 | 253 |
| 9.51.3.3 | operator<                                  | 253 |
| 9.51.3.4 | operator<                                  | 253 |
| 9.51.3.5 | operator==                                 | 254 |
| 9.51.3.6 | operator==                                 | 254 |
| 9.51.4   | Member Data Documentation                  | 254 |
| 9.51.4.1 | target                                     | 254 |
| 9.51.4.2 | theTarget                                  | 254 |
| 9.52     | GraphRep Class Reference                   | 255 |
| 9.52.1   | Detailed Description                       | 256 |
| 9.52.2   | Member Typedef Documentation               | 256 |
| 9.52.2.1 | edge_type                                  | 256 |
| 9.52.2.2 | state_type                                 | 256 |
| 9.52.3   | Constructor & Destructor Documentation     | 256 |
| 9.52.3.1 | GraphRep                                   | 256 |
| 9.52.4   | Member Function Documentation              | 256 |
| 9.52.4.1 | clear                                      | 256 |
| 9.52.4.2 | clone                                      | 256 |
| 9.52.4.3 | getStates                                  | 256 |
| 9.52.4.4 | newEdge                                    | 257 |
| 9.52.4.5 | newState                                   | 257 |
| 9.52.5   | Friends And Related Function Documentation | 257 |
| 9.52.5.1 | Graph                                      | 257 |
| 9.52.6   | Member Data Documentation                  | 257 |
| 9.52.6.1 | maxState                                   | 257 |
| 9.52.6.2 | theStates                                  | 257 |
| 9.53     | GraphState Struct Reference                | 258 |
| 9.53.1   | Detailed Description                       | 258 |
| 9.53.2   | Member Typedef Documentation               | 258 |
| 9.53.2.1 | edge_type                                  | 258 |
| 9.53.3   | Constructor & Destructor Documentation     | 258 |

|          |   |     |
|----------|---|-----|
| 9.53.3.1 | GraphState . . . . .  | 258 |
| 9.53.3.2 | GraphState . . . . .  | 258 |
| 9.53.4   | Member Function Documentation . . . . .                     | 259 |
| 9.53.4.1 | operator< . . . . .   | 259 |
| 9.53.4.2 | operator== . . . . .  | 259 |
| 9.53.5   | Member Data Documentation . . . . .                         | 259 |
| 9.53.5.1 | in . . . . .  | 259 |
| 9.53.5.2 | out . . . . .   | 259 |
| 9.53.5.3 | theState . . . . .  | 259 |
| 9.54     | GraphVertex< EdgeType > Struct Template Reference . . . . . | 260 |
| 9.54.1   | Detailed Description . . . . .                              | 260 |
| 9.54.2   | Member Typedef Documentation . . . . .                      | 260 |
| 9.54.2.1 | edge_type . . . . .   | 260 |
| 9.54.3   | Constructor & Destructor Documentation . . . . .            | 261 |
| 9.54.3.1 | GraphVertex . . . . .                                       | 261 |
| 9.54.4   | Member Data Documentation . . . . .                         | 261 |
| 9.54.4.1 | in . . . . .  | 261 |
| 9.54.4.2 | out . . . . .   | 261 |
| 9.55     | GRImage Class Reference . . . . .                           | 262 |
| 9.55.1   | Detailed Description . . . . .                              | 263 |
| 9.55.2   | Constructor & Destructor Documentation . . . . .            | 263 |
| 9.55.2.1 | GRImage . . . . .   | 263 |
| 9.55.2.2 | GRImage . . . . .   | 263 |
| 9.55.2.3 | GRImage . . . . .   | 263 |
| 9.55.2.4 | GRImage . . . . .   | 263 |
| 9.55.2.5 | GRImage . . . . .   | 263 |
| 9.55.2.6 | ~GRImage . . . . .  | 263 |
| 9.55.3   | Member Function Documentation . . . . .                     | 263 |
| 9.55.3.1 | getPixel . . . . .  | 263 |
| 9.55.3.2 | getPixel . . . . .  | 263 |
| 9.55.3.3 | getType . . . . .   | 264 |

|           |  |     |
|-----------|--|-----|
| 9.55.3.4  | operator= . . . . .                                  | 264 |
| 9.55.3.5  | printOnBW . . . . .                                  | 264 |
| 9.55.3.6  | printOnPGM . . . . .                                 | 264 |
| 9.55.3.7  | readFromBW . . . . .                                 | 264 |
| 9.55.3.8  | readFromPGM . . . . .                                | 264 |
| 9.55.3.9  | saveTo . . . . .                                     | 264 |
| 9.55.3.10 | setPixel . . . . .                                   | 264 |
| 9.55.3.11 | setPixel . . . . .                                   | 264 |
| 9.55.3.12 | setPixelClipped . . . . .                            | 264 |
| 9.55.3.13 | setPixelClipped . . . . .                            | 264 |
| 9.55.4    | Friends And Related Function Documentation . . . . . | 264 |
| 9.55.4.1  | GRLUTImage . . . . .                                 | 264 |
| 9.55.5    | Member Data Documentation . . . . .                  | 265 |
| 9.55.5.1  | image_string . . . . .                               | 265 |
| 9.56      | GRLUTImage Class Reference . . . . .                 | 266 |
| 9.56.1    | Detailed Description . . . . .                       | 266 |
| 9.56.2    | Constructor & Destructor Documentation . . . . .     | 266 |
| 9.56.2.1  | GRLUTImage . . . . .                                 | 266 |
| 9.56.2.2  | GRLUTImage . . . . .                                 | 267 |
| 9.56.2.3  | GRLUTImage . . . . .                                 | 267 |
| 9.56.2.4  | GRLUTImage . . . . .                                 | 267 |
| 9.56.3    | Member Function Documentation . . . . .              | 267 |
| 9.56.3.1  | getPixel . . . . .                                   | 267 |
| 9.56.3.2  | getPixel . . . . .                                   | 267 |
| 9.56.3.3  | printOn . . . . .                                    | 267 |
| 9.56.4    | Friends And Related Function Documentation . . . . . | 267 |
| 9.56.4.1  | operator<< . . . . .                                 | 267 |
| 9.57      | HammingDistance Class Reference . . . . .            | 268 |
| 9.57.1    | Detailed Description . . . . .                       | 268 |
| 9.57.2    | Constructor & Destructor Documentation . . . . .     | 268 |
| 9.57.2.1  | HammingDistance . . . . .                            | 268 |

|          |  |     |
|----------|--|-----|
| 9.57.3   | Member Function Documentation . . . . .  | 268 |
| 9.57.3.1 | measure . . . . .  | 268 |
| 9.58     | HammingDistanceCyclic Class Reference . . . . .  | 270 |
| 9.58.1   | Detailed Description . . . . .   | 270 |
| 9.58.2   | Constructor & Destructor Documentation . . . . .   | 270 |
| 9.58.2.1 | HammingDistanceCyclic . . . . .  | 270 |
| 9.58.3   | Member Function Documentation . . . . .  | 270 |
| 9.58.3.1 | measure . . . . .  | 270 |
| 9.59     | InfiniteAlphabet Class Reference . . . . .   | 272 |
| 9.59.1   | Detailed Description . . . . .   | 273 |
| 9.59.2   | Constructor & Destructor Documentation . . . . .   | 273 |
| 9.59.2.1 | InfiniteAlphabet . . . . .   | 273 |
| 9.59.3   | Member Function Documentation . . . . .  | 273 |
| 9.59.3.1 | getLetter . . . . .  | 273 |
| 9.59.3.2 | getNum . . . . .   | 273 |
| 9.59.4   | Friends And Related Function Documentation . . . . .                                     | 274 |
| 9.59.4.1 | operator<< . . . . .   | 274 |
| 9.59.5   | Member Data Documentation . . . . .  | 274 |
| 9.59.5.1 | defaultAlphabet . . . . .  | 274 |
| 9.59.5.2 | thePrefix . . . . .  | 274 |
| 9.60     | PC::SignMatrix< RowHdr, ColHdr, INTERNAL_TYPE >::intColHdr<br>Struct Reference . . . . . | 275 |
| 9.60.1   | Detailed Description . . . . .   | 275 |
| 9.60.2   | Member Data Documentation . . . . .  | 275 |
| 9.60.2.1 | ch . . . . .   | 275 |
| 9.60.2.2 | col . . . . .  | 275 |
| 9.60.2.3 | colno . . . . .  | 275 |
| 9.60.2.4 | nextUnused . . . . .   | 276 |
| 9.60.2.5 | used . . . . .   | 276 |
| 9.61     | IntLabeledEdge Struct Reference . . . . .  | 277 |
| 9.61.1   | Detailed Description . . . . .   | 278 |

|          |   |     |
|----------|---|-----|
| 9.61.2   | Constructor & Destructor Documentation . . . . .                  | 278 |
| 9.61.2.1 | IntLabeledEdge . . . . .  | 278 |
| 9.61.2.2 | IntLabeledEdge . . . . .  | 278 |
| 9.61.3   | Member Function Documentation . . . . .                           | 278 |
| 9.61.3.1 | inverse . . . . .   | 278 |
| 9.61.3.2 | operator!= . . . . .  | 278 |
| 9.61.3.3 | operator< . . . . .   | 279 |
| 9.61.3.4 | operator== . . . . .  | 279 |
| 9.61.4   | Member Data Documentation . . . . .                               | 279 |
| 9.61.4.1 | theLabel . . . . .  | 279 |
| 9.62     | PC::PowerCircuitGraph::IntMarking Struct Reference . . . . .      | 280 |
| 9.62.1   | Detailed Description . . . . .                                    | 280 |
| 9.62.2   | Member Data Documentation . . . . .                               | 280 |
| 9.62.2.1 | deleted . . . . .   | 280 |
| 9.62.2.2 | nextDeleted . . . . .   | 280 |
| 9.62.2.3 | nodes . . . . .   | 280 |
| 9.62.2.4 | reduced . . . . .   | 280 |
| 9.62.2.5 | refCount . . . . .  | 280 |
| 9.63     | PC::PowerCircuitCompMatrix::IntMarking Struct Reference . . . . . | 281 |
| 9.63.1   | Detailed Description . . . . .                                    | 281 |
| 9.63.2   | Member Data Documentation . . . . .                               | 281 |
| 9.63.2.1 | col . . . . .   | 281 |
| 9.63.2.2 | deleted . . . . .   | 281 |
| 9.63.2.3 | nextDeleted . . . . .   | 281 |
| 9.63.2.4 | node . . . . .  | 281 |
| 9.63.2.5 | pc . . . . .  | 281 |
| 9.63.2.6 | type . . . . .  | 282 |
| 9.64     | PC::PowerCircuitCompMatrix::IntNode Struct Reference . . . . .    | 283 |
| 9.64.1   | Detailed Description . . . . .                                    | 283 |
| 9.64.2   | Member Data Documentation . . . . .                               | 283 |
| 9.64.2.1 | deleted . . . . .   | 283 |

|          |  |     |
|----------|--|-----|
| 9.64.2.2 | nextDeleted . . . . .  | 283 |
| 9.64.2.3 | pc . . . . .   | 283 |
| 9.64.2.4 | row . . . . .  | 283 |
| 9.65     | PC::PowerCircuitGraph::IntNode Struct Reference . . . . .                                | 284 |
| 9.65.1   | Detailed Description . . . . .   | 284 |
| 9.65.2   | Member Data Documentation . . . . .  | 284 |
| 9.65.2.1 | BV . . . . .   | 284 |
| 9.65.2.2 | deleted . . . . .  | 284 |
| 9.65.2.3 | indexInOrder . . . . .   | 284 |
| 9.65.2.4 | nextDeleted . . . . .  | 284 |
| 9.65.2.5 | successors . . . . .   | 284 |
| 9.66     | PC::SignMatrix< RowHdr, ColHdr, INTERNAL_TYPE<br>>::intRowHdr Struct Reference . . . . . | 285 |
| 9.66.1   | Detailed Description . . . . .   | 285 |
| 9.66.2   | Member Data Documentation . . . . .  | 285 |
| 9.66.2.1 | nextUnused . . . . .   | 285 |
| 9.66.2.2 | rh . . . . .   | 285 |
| 9.66.2.3 | row . . . . .  | 285 |
| 9.66.2.4 | rowno . . . . .  | 286 |
| 9.66.2.5 | used . . . . .   | 286 |
| 9.67     | KLProtocolInstance Class Reference . . . . .   | 287 |
| 9.67.1   | Detailed Description . . . . .   | 288 |
| 9.67.2   | Constructor & Destructor Documentation . . . . .   | 288 |
| 9.67.2.1 | KLProtocolInstance . . . . .   | 288 |
| 9.67.3   | Member Function Documentation . . . . .  | 288 |
| 9.67.3.1 | getBraidRank . . . . .   | 288 |
| 9.67.3.2 | getPrivateKeyA . . . . .   | 288 |
| 9.67.3.3 | getPrivateKeyB . . . . .   | 289 |
| 9.67.3.4 | getPublicKeyA . . . . .  | 289 |
| 9.67.3.5 | getPublicKeyB . . . . .  | 289 |
| 9.67.3.6 | random . . . . .   | 289 |

|          |  |     |
|----------|--|-----|
| 9.67.4   | Member Data Documentation . . . . .              | 289 |
| 9.67.4.1 | theBase . . . . .                                | 289 |
| 9.67.4.2 | thePrivateKeyA . . . . .                         | 289 |
| 9.67.4.3 | thePrivateKeyB . . . . .                         | 290 |
| 9.67.4.4 | thePublicKeyA . . . . .                          | 290 |
| 9.67.4.5 | thePublicKeyB . . . . .                          | 290 |
| 9.67.4.6 | theRank . . . . .                                | 290 |
| 9.68     | ICommDivider Struct Reference . . . . .          | 291 |
| 9.68.1   | Detailed Description . . . . .                   | 291 |
| 9.68.2   | Member Function Documentation . . . . .          | 291 |
| 9.68.2.1 | operator() . . . . .                             | 291 |
| 9.69     | LengthAttack_A1 Class Reference . . . . .        | 292 |
| 9.69.1   | Detailed Description . . . . .                   | 292 |
| 9.69.2   | Constructor & Destructor Documentation . . . . . | 293 |
| 9.69.2.1 | LengthAttack_A1 . . . . .                        | 293 |
| 9.69.3   | Member Function Documentation . . . . .          | 293 |
| 9.69.3.1 | addNewElt . . . . .                              | 293 |
| 9.69.3.2 | check_ifVectorsEqual . . . . .                   | 293 |
| 9.69.3.3 | findKey_LengthBased . . . . .                    | 293 |
| 9.69.3.4 | sbgpGeneratorsWeight . . . . .                   | 294 |
| 9.69.3.5 | tryElt . . . . .                                 | 294 |
| 9.69.3.6 | type . . . . .                                   | 294 |
| 9.70     | LengthAttack_A2 Class Reference . . . . .        | 295 |
| 9.70.1   | Detailed Description . . . . .                   | 295 |
| 9.70.2   | Constructor & Destructor Documentation . . . . . | 296 |
| 9.70.2.1 | LengthAttack_A2 . . . . .                        | 296 |
| 9.70.3   | Member Function Documentation . . . . .          | 296 |
| 9.70.3.1 | addNewElt . . . . .                              | 296 |
| 9.70.3.2 | check_ifVectorsEqual . . . . .                   | 296 |
| 9.70.3.3 | findKey_LengthBased . . . . .                    | 296 |
| 9.70.3.4 | sbgpGeneratorsWeight . . . . .                   | 297 |



|          |  |     |
|----------|--|-----|
| 9.70.3.5 | tryElt . . . . .                                 | 297 |
| 9.70.3.6 | tryElt . . . . .                                 | 297 |
| 9.70.3.7 | type . . . . .                                   | 297 |
| 9.71     | LengthAttack_A3 Class Reference . . . . .        | 298 |
| 9.71.1   | Detailed Description . . . . .                   | 299 |
| 9.71.2   | Constructor & Destructor Documentation . . . . . | 299 |
| 9.71.2.1 | LengthAttack_A3 . . . . .                        | 299 |
| 9.71.3   | Member Function Documentation . . . . .          | 299 |
| 9.71.3.1 | addAllProducts . . . . .                         | 299 |
| 9.71.3.2 | addNewElt . . . . .                              | 299 |
| 9.71.3.3 | addProducts . . . . .                            | 299 |
| 9.71.3.4 | check_ifVectorsEqual . . . . .                   | 299 |
| 9.71.3.5 | findKey_LengthBased . . . . .                    | 299 |
| 9.71.3.6 | sbgpGeneratorsWeight . . . . .                   | 300 |
| 9.71.3.7 | tryElt . . . . .                                 | 300 |
| 9.71.3.8 | tryElt . . . . .                                 | 300 |
| 9.71.3.9 | type . . . . .                                   | 300 |
| 9.72     | LengthAttackBase Class Reference . . . . .       | 301 |
| 9.72.1   | Detailed Description . . . . .                   | 301 |
| 9.72.2   | Member Function Documentation . . . . .          | 301 |
| 9.72.2.1 | findKey_LengthBased . . . . .                    | 301 |
| 9.72.2.2 | type . . . . .                                   | 302 |
| 9.73     | Levenstein Class Reference . . . . .             | 303 |
| 9.73.1   | Detailed Description . . . . .                   | 303 |
| 9.73.2   | Constructor & Destructor Documentation . . . . . | 303 |
| 9.73.2.1 | Levenstein . . . . .                             | 303 |
| 9.73.3   | Member Function Documentation . . . . .          | 303 |
| 9.73.3.1 | compute . . . . .                                | 303 |
| 9.73.3.2 | wordToString . . . . .                           | 304 |
| 9.74     | LinkedBraidStructure Class Reference . . . . .   | 305 |
| 9.74.1   | Detailed Description . . . . .                   | 306 |

|           |  |     |
|-----------|--|-----|
| 9.74.2    | Constructor & Destructor Documentation . . . . . | 307 |
| 9.74.2.1  | LinkedBraidStructure . . . . .                   | 307 |
| 9.74.2.2  | LinkedBraidStructure . . . . .                   | 307 |
| 9.74.2.3  | LinkedBraidStructure . . . . .                   | 307 |
| 9.74.2.4  | LinkedBraidStructure . . . . .                   | 307 |
| 9.74.2.5  | ~LinkedBraidStructure . . . . .                  | 307 |
| 9.74.3    | Member Function Documentation . . . . .          | 307 |
| 9.74.3.1  | checkIfStartsLeftHandle . . . . .                | 307 |
| 9.74.3.2  | checkIfStartsRightHandle . . . . .               | 307 |
| 9.74.3.3  | clear . . . . .                                  | 307 |
| 9.74.3.4  | clear . . . . .                                  | 307 |
| 9.74.3.5  | clearLinks . . . . .                             | 307 |
| 9.74.3.6  | getWord . . . . .                                | 307 |
| 9.74.3.7  | insert . . . . .                                 | 307 |
| 9.74.3.8  | insertBackLeft . . . . .                         | 307 |
| 9.74.3.9  | insertBackRight . . . . .                        | 307 |
| 9.74.3.10 | make_AddTransform . . . . .                      | 307 |
| 9.74.3.11 | make_ChangeType . . . . .                        | 307 |
| 9.74.3.12 | make_EraseTransform . . . . .                    | 307 |
| 9.74.3.13 | operator< . . . . .                              | 307 |
| 9.74.3.14 | operator= . . . . .                              | 307 |
| 9.74.3.15 | processNode . . . . .                            | 307 |
| 9.74.3.16 | processTree . . . . .                            | 308 |
| 9.74.3.17 | push_back . . . . .                              | 308 |
| 9.74.3.18 | push_back . . . . .                              | 308 |
| 9.74.3.19 | push_back . . . . .                              | 308 |
| 9.74.3.20 | push_front . . . . .                             | 309 |
| 9.74.3.21 | push_front . . . . .                             | 309 |
| 9.74.3.22 | removeLeftHandle . . . . .                       | 309 |
| 9.74.3.23 | removeLeftHandles . . . . .                      | 309 |
| 9.74.3.24 | removeNode . . . . .                             | 309 |

---

|           |  |     |
|-----------|--|-----|
| 9.74.3.25 | removeRightHandle . . . . .                              | 309 |
| 9.74.3.26 | removeRightHandles . . . . .                             | 309 |
| 9.74.3.27 | size . . . . .   | 309 |
| 9.74.3.28 | transformToDehornoyForm . . . . .                        | 310 |
| 9.74.3.29 | translateIntoWord . . . . .                              | 310 |
| 9.74.3.30 | undo . . . . .   | 310 |
| 9.74.3.31 | undo . . . . .   | 310 |
| 9.74.4    | Member Data Documentation . . . . .                      | 310 |
| 9.74.4.1  | backNodes . . . . .                                      | 310 |
| 9.74.4.2  | frontNodes . . . . .                                     | 310 |
| 9.74.4.3  | maxNodeNumber . . . . .                                  | 310 |
| 9.74.4.4  | theIndex . . . . .                                       | 310 |
| 9.74.4.5  | theMark . . . . .  | 310 |
| 9.74.4.6  | theNodes . . . . .                                       | 310 |
| 9.74.4.7  | theRank . . . . .  | 311 |
| 9.75      | LinkedBraidStructureTransform Struct Reference . . . . . | 312 |
| 9.75.1    | Detailed Description . . . . .                           | 312 |
| 9.75.2    | Member Enumeration Documentation . . . . .               | 312 |
| 9.75.2.1  | TRANSFORM . . . . .                                      | 312 |
| 9.75.3    | Constructor & Destructor Documentation . . . . .         | 313 |
| 9.75.3.1  | LinkedBraidStructureTransform . . . . .                  | 313 |
| 9.75.4    | Member Data Documentation . . . . .                      | 313 |
| 9.75.4.1  | ahead . . . . .  | 313 |
| 9.75.4.2  | back . . . . .   | 313 |
| 9.75.4.3  | back_left . . . . .                                      | 313 |
| 9.75.4.4  | back_right . . . . .                                     | 313 |
| 9.75.4.5  | left . . . . .   | 313 |
| 9.75.4.6  | right . . . . .  | 313 |
| 9.75.4.7  | theNumber . . . . .                                      | 313 |
| 9.75.4.8  | thePosition . . . . .                                    | 314 |
| 9.75.4.9  | theTransform . . . . .                                   | 314 |

|   |     |
|---|-----|
| 9.75.4.10 type . . . . .                                | 314 |
| 9.76 LookUpTable Class Reference . . . . .              | 315 |
| 9.76.1 Detailed Description . . . . .                   | 315 |
| 9.76.2 Constructor & Destructor Documentation . . . . . | 315 |
| 9.76.2.1 LookUpTable . . . . .                          | 315 |
| 9.76.2.2 LookUpTable . . . . .                          | 315 |
| 9.76.3 Member Function Documentation . . . . .          | 315 |
| 9.76.3.1 get . . . . .                                  | 315 |
| 9.76.3.2 operator= . . . . .                            | 315 |
| 9.76.3.3 set . . . . .                                  | 315 |
| 9.76.4 Member Data Documentation . . . . .              | 316 |
| 9.76.4.1 table . . . . .                                | 316 |
| 9.77 Itstr Struct Reference . . . . .                   | 317 |
| 9.77.1 Detailed Description . . . . .                   | 317 |
| 9.77.2 Member Function Documentation . . . . .          | 317 |
| 9.77.2.1 operator() . . . . .                           | 317 |
| 9.78 LUTImage Class Reference . . . . .                 | 318 |
| 9.78.1 Detailed Description . . . . .                   | 318 |
| 9.78.2 Constructor & Destructor Documentation . . . . . | 318 |
| 9.78.2.1 LUTImage . . . . .                             | 318 |
| 9.78.2.2 LUTImage . . . . .                             | 318 |
| 9.78.2.3 LUTImage . . . . .                             | 319 |
| 9.78.2.4 LUTImage . . . . .                             | 319 |
| 9.78.3 Member Function Documentation . . . . .          | 319 |
| 9.78.3.1 getInLT . . . . .                              | 319 |
| 9.78.3.2 setInLT . . . . .                              | 319 |
| 9.78.3.3 setLookUpTable . . . . .                       | 319 |
| 9.78.4 Member Data Documentation . . . . .              | 319 |
| 9.78.4.1 theLookUpTable . . . . .                       | 319 |
| 9.79 Map Class Reference . . . . .                      | 320 |
| 9.79.1 Detailed Description . . . . .                   | 322 |

|           |  |     |
|-----------|--|-----|
| 9.79.2    | Constructor & Destructor Documentation . . . . .     | 322 |
| 9.79.2.1  | Map . . . . .  | 322 |
| 9.79.2.2  | Map . . . . .  | 322 |
| 9.79.2.3  | Map . . . . .  | 322 |
| 9.79.2.4  | Map . . . . .  | 323 |
| 9.79.3    | Member Function Documentation . . . . .              | 323 |
| 9.79.3.1  | domainAlphabet . . . . .                             | 323 |
| 9.79.3.2  | domainSize . . . . .                                 | 323 |
| 9.79.3.3  | generatingImages . . . . .                           | 323 |
| 9.79.3.4  | generatingImages . . . . .                           | 324 |
| 9.79.3.5  | imageOf . . . . .                                    | 324 |
| 9.79.3.6  | operator== . . . . .                                 | 324 |
| 9.79.3.7  | operator  . . . . .                                  | 324 |
| 9.79.3.8  | printOn . . . . .                                    | 325 |
| 9.79.3.9  | rangeAlphabet . . . . .                              | 325 |
| 9.79.3.10 | rangeSize . . . . .                                  | 325 |
| 9.79.3.11 | readChar . . . . .                                   | 325 |
| 9.79.3.12 | readFrom . . . . .                                   | 325 |
| 9.79.3.13 | setGeneratingImages . . . . .                        | 325 |
| 9.79.3.14 | setGeneratingImages . . . . .                        | 325 |
| 9.79.4    | Friends And Related Function Documentation . . . . . | 326 |
| 9.79.4.1  | composition . . . . .                                | 326 |
| 9.79.4.2  | operator<< . . . . .                                 | 326 |
| 9.79.4.3  | operator>> . . . . .                                 | 326 |
| 9.79.5    | Member Data Documentation . . . . .                  | 326 |
| 9.79.5.1  | theDomain . . . . .                                  | 326 |
| 9.79.5.2  | theDomainAlphabet . . . . .                          | 326 |
| 9.79.5.3  | theGeneratingImages . . . . .                        | 327 |
| 9.79.5.4  | theRange . . . . .                                   | 327 |
| 9.79.5.5  | theRangeAlphabet . . . . .                           | 327 |
| 9.80      | __gnu_cxx::map_hash Struct Reference . . . . .       | 328 |

|   |     |
|---|-----|
| 9.80.1 Detailed Description . . . . .                   | 328 |
| 9.80.2 Member Function Documentation . . . . .          | 328 |
| 9.80.2.1 operator() . . . . .                           | 328 |
| 9.81 PC::Marking Class Reference . . . . .              | 329 |
| 9.81.1 Detailed Description . . . . .                   | 330 |
| 9.81.2 Constructor & Destructor Documentation . . . . . | 330 |
| 9.81.2.1 Marking . . . . .                              | 330 |
| 9.81.2.2 Marking . . . . .                              | 331 |
| 9.81.2.3 Marking . . . . .                              | 331 |
| 9.81.2.4 ~Marking . . . . .                             | 331 |
| 9.81.3 Member Function Documentation . . . . .          | 331 |
| 9.81.3.1 clone . . . . .                                | 331 |
| 9.81.3.2 copy . . . . .                                 | 331 |
| 9.81.3.3 getIncidentNode . . . . .                      | 331 |
| 9.81.3.4 getNodes . . . . .                             | 331 |
| 9.81.3.5 getSign . . . . .                              | 331 |
| 9.81.3.6 getSmallestNode . . . . .                      | 331 |
| 9.81.3.7 isDefined . . . . .                            | 331 |
| 9.81.3.8 isReduced . . . . .                            | 331 |
| 9.81.3.9 isSuccessorMarking . . . . .                   | 331 |
| 9.81.3.10 operator!= . . . . .                          | 331 |
| 9.81.3.11 operator& . . . . .                           | 331 |
| 9.81.3.12 operator+ . . . . .                           | 331 |
| 9.81.3.13 operator++ . . . . .                          | 331 |
| 9.81.3.14 operator- . . . . .                           | 331 |
| 9.81.3.15 operator< . . . . .                           | 331 |
| 9.81.3.16 operator<= . . . . .                          | 331 |
| 9.81.3.17 operator= . . . . .                           | 331 |
| 9.81.3.18 operator== . . . . .                          | 331 |
| 9.81.3.19 operator> . . . . .                           | 331 |
| 9.81.3.20 operator>= . . . . .                          | 331 |

|           |  |     |
|-----------|--|-----|
| 9.81.4    | Member Data Documentation . . . . .              | 331 |
| 9.81.4.1  | id . . . . .                                     | 331 |
| 9.81.4.2  | pc . . . . .                                     | 332 |
| 9.82      | MatrixFp Class Reference . . . . .               | 333 |
| 9.82.1    | Detailed Description . . . . .                   | 333 |
| 9.82.2    | Constructor & Destructor Documentation . . . . . | 334 |
| 9.82.2.1  | MatrixFp . . . . .                               | 334 |
| 9.82.2.2  | ~MatrixFp . . . . .                              | 334 |
| 9.82.2.3  | MatrixFp . . . . .                               | 334 |
| 9.82.3    | Member Function Documentation . . . . .          | 334 |
| 9.82.3.1  | clean . . . . .                                  | 334 |
| 9.82.3.2  | getPower . . . . .                               | 334 |
| 9.82.3.3  | ID . . . . .                                     | 334 |
| 9.82.3.4  | init . . . . .                                   | 334 |
| 9.82.3.5  | operator* . . . . .                              | 335 |
| 9.82.3.6  | operator+ . . . . .                              | 335 |
| 9.82.3.7  | operator= . . . . .                              | 335 |
| 9.82.3.8  | random . . . . .                                 | 335 |
| 9.82.3.9  | scalar_mult . . . . .                            | 335 |
| 9.82.3.10 | set . . . . .                                    | 335 |
| 9.82.4    | Member Data Documentation . . . . .              | 335 |
| 9.82.4.1  | the_n . . . . .                                  | 335 |
| 9.82.4.2  | the_p . . . . .                                  | 335 |
| 9.82.4.3  | theMatrix . . . . .                              | 335 |
| 9.83      | MaxCommutePartition Class Reference . . . . .    | 336 |
| 9.83.1    | Detailed Description . . . . .                   | 336 |
| 9.83.2    | Constructor & Destructor Documentation . . . . . | 336 |
| 9.83.2.1  | MaxCommutePartition . . . . .                    | 336 |
| 9.83.3    | Member Function Documentation . . . . .          | 336 |
| 9.83.3.1  | findCommutParts . . . . .                        | 336 |
| 9.83.3.2  | getPartition . . . . .                           | 337 |

|          |  |     |
|----------|--|-----|
| 9.83.4   | Member Data Documentation . . . . .              | 337 |
| 9.83.4.1 | N . . . . .                                      | 337 |
| 9.83.4.2 | partition . . . . .                              | 337 |
| 9.84     | MotivePattern Struct Reference . . . . .         | 338 |
| 9.84.1   | Detailed Description . . . . .                   | 338 |
| 9.84.2   | Member Data Documentation . . . . .              | 338 |
| 9.84.2.1 | occurrences . . . . .                            | 338 |
| 9.84.2.2 | sequences . . . . .                              | 338 |
| 9.84.2.3 | thePattern . . . . .                             | 338 |
| 9.85     | MotivePatternWrapper Class Reference . . . . .   | 339 |
| 9.85.1   | Detailed Description . . . . .                   | 339 |
| 9.85.2   | Constructor & Destructor Documentation . . . . . | 339 |
| 9.85.2.1 | MotivePatternWrapper . . . . .                   | 339 |
| 9.85.3   | Member Function Documentation . . . . .          | 340 |
| 9.85.3.1 | compute . . . . .                                | 340 |
| 9.85.3.2 | getPatterns . . . . .                            | 340 |
| 9.85.3.3 | initializeSequence . . . . .                     | 340 |
| 9.85.4   | Member Data Documentation . . . . .              | 340 |
| 9.85.4.1 | input_file . . . . .                             | 340 |
| 9.85.4.2 | output_file . . . . .                            | 340 |
| 9.85.4.3 | theGroup . . . . .                               | 340 |
| 9.86     | NielsenAutoSet Class Reference . . . . .         | 341 |
| 9.86.1   | Detailed Description . . . . .                   | 341 |
| 9.86.2   | Constructor & Destructor Documentation . . . . . | 341 |
| 9.86.2.1 | NielsenAutoSet . . . . .                         | 341 |
| 9.86.2.2 | ~NielsenAutoSet . . . . .                        | 342 |
| 9.86.3   | Member Function Documentation . . . . .          | 342 |
| 9.86.3.1 | getRandomAuto . . . . .                          | 342 |
| 9.86.3.2 | getSet . . . . .                                 | 342 |
| 9.86.4   | Member Data Documentation . . . . .              | 342 |
| 9.86.4.1 | nGens . . . . .                                  | 342 |



|          |  |     |
|----------|--|-----|
| 9.86.4.2 | theSet . . . . .   | 343 |
| 9.87     | PC::Node Class Reference . . . . .                               | 344 |
| 9.87.1   | Detailed Description . . . . .                                   | 344 |
| 9.87.2   | Constructor & Destructor Documentation . . . . .                 | 344 |
| 9.87.2.1 | Node . . . . .   | 344 |
| 9.87.2.2 | Node . . . . .   | 344 |
| 9.87.3   | Member Function Documentation . . . . .                          | 345 |
| 9.87.3.1 | clone . . . . .  | 345 |
| 9.87.3.2 | getOrder . . . . .   | 345 |
| 9.87.3.3 | getSuccessorMarking . . . . .                                    | 345 |
| 9.87.3.4 | isDefined . . . . .  | 345 |
| 9.87.3.5 | isReduced . . . . .  | 345 |
| 9.87.3.6 | operator== . . . . .   | 345 |
| 9.87.4   | Member Data Documentation . . . . .                              | 345 |
| 9.87.4.1 | id . . . . .   | 345 |
| 9.87.4.2 | pc . . . . .   | 345 |
| 9.88     | PC::PowerCircuitGraph::NodeUsedByType Struct Reference . . . . . | 346 |
| 9.88.1   | Detailed Description . . . . .                                   | 346 |
| 9.88.2   | Constructor & Destructor Documentation . . . . .                 | 346 |
| 9.88.2.1 | NodeUsedByType . . . . .   | 346 |
| 9.88.3   | Member Function Documentation . . . . .                          | 346 |
| 9.88.3.1 | operator== . . . . .   | 346 |
| 9.88.4   | Member Data Documentation . . . . .                              | 346 |
| 9.88.4.1 | id . . . . .   | 346 |
| 9.88.4.2 | nodeList . . . . .   | 347 |
| 9.88.4.3 | position . . . . .   | 347 |
| 9.89     | ObjectOf< Rep > Class Template Reference . . . . .               | 348 |
| 9.89.1   | Detailed Description . . . . .                                   | 348 |
| 9.89.2   | Constructor & Destructor Documentation . . . . .                 | 348 |
| 9.89.2.1 | ObjectOf . . . . .   | 348 |
| 9.89.2.2 | ~ObjectOf . . . . .  | 349 |

|          |  |     |
|----------|--|-----|
| 9.89.2.3 | ObjectOf . . . . .                                   | 349 |
| 9.89.3   | Member Function Documentation . . . . .              | 349 |
| 9.89.3.1 | acquireRep . . . . .                                 | 349 |
| 9.89.3.2 | change . . . . .                                     | 349 |
| 9.89.3.3 | enhance . . . . .                                    | 349 |
| 9.89.3.4 | force_derivation . . . . .                           | 349 |
| 9.89.3.5 | look . . . . .                                       | 350 |
| 9.89.3.6 | operator= . . . . .                                  | 350 |
| 9.89.4   | Member Data Documentation . . . . .                  | 350 |
| 9.89.4.1 | theRep . . . . .                                     | 350 |
| 9.90     | PairDistanceSimilarityTest Class Reference . . . . . | 351 |
| 9.90.1   | Detailed Description . . . . .                       | 351 |
| 9.90.2   | Constructor & Destructor Documentation . . . . .     | 351 |
| 9.90.2.1 | PairDistanceSimilarityTest . . . . .                 | 351 |
| 9.90.3   | Member Function Documentation . . . . .              | 352 |
| 9.90.3.1 | estimateTrueDistribution . . . . .                   | 352 |
| 9.90.3.2 | testFalsePair . . . . .                              | 352 |
| 9.90.3.3 | testPair . . . . .                                   | 352 |
| 9.90.3.4 | testTruePair . . . . .                               | 352 |
| 9.90.4   | Member Data Documentation . . . . .                  | 352 |
| 9.90.4.1 | measureDistr . . . . .                               | 352 |
| 9.90.4.2 | sampleSize . . . . .                                 | 352 |
| 9.90.4.3 | theGenerator . . . . .                               | 352 |
| 9.90.4.4 | theSimilarity . . . . .                              | 352 |
| 9.91     | PairGenerator Class Reference . . . . .              | 353 |
| 9.91.1   | Detailed Description . . . . .                       | 353 |
| 9.91.2   | Member Function Documentation . . . . .              | 353 |
| 9.91.2.1 | getFalsePair . . . . .                               | 353 |
| 9.91.2.2 | getTruePair . . . . .                                | 353 |
| 9.92     | Parser Class Reference . . . . .                     | 355 |
| 9.92.1   | Detailed Description . . . . .                       | 355 |

---

|          |  |     |
|----------|--|-----|
| 9.92.2   | Constructor & Destructor Documentation . . . . .     | 355 |
| 9.92.2.1 | Parser . . . . .                                     | 355 |
| 9.92.2.2 | ~Parser . . . . .                                    | 355 |
| 9.92.3   | Member Function Documentation . . . . .              | 355 |
| 9.92.3.1 | getWord . . . . .                                    | 355 |
| 9.92.3.2 | getWordTerminalSymbol . . . . .                      | 355 |
| 9.92.3.3 | parse . . . . .                                      | 355 |
| 9.92.4   | Member Data Documentation . . . . .                  | 355 |
| 9.92.4.1 | localFlexLexer . . . . .                             | 355 |
| 9.92.4.2 | theTS . . . . .                                      | 356 |
| 9.92.4.3 | theWord . . . . .                                    | 356 |
| 9.93     | Partition Class Reference . . . . .                  | 357 |
| 9.93.1   | Detailed Description . . . . .                       | 357 |
| 9.93.2   | Member Function Documentation . . . . .              | 357 |
| 9.93.2.1 | getPartition . . . . .                               | 357 |
| 9.94     | PBar Struct Reference . . . . .                      | 358 |
| 9.94.1   | Detailed Description . . . . .                       | 358 |
| 9.94.2   | Constructor & Destructor Documentation . . . . .     | 358 |
| 9.94.2.1 | PBar . . . . .                                       | 358 |
| 9.94.2.2 | PBar . . . . .                                       | 358 |
| 9.94.2.3 | PBar . . . . .                                       | 358 |
| 9.94.2.4 | PBar . . . . .                                       | 359 |
| 9.94.3   | Friends And Related Function Documentation . . . . . | 359 |
| 9.94.3.1 | operator<< . . . . .                                 | 359 |
| 9.94.3.2 | operator<< . . . . .                                 | 359 |
| 9.94.4   | Member Data Documentation . . . . .                  | 359 |
| 9.94.4.1 | progress . . . . .                                   | 359 |
| 9.95     | PDFPage Class Reference . . . . .                    | 360 |
| 9.95.1   | Detailed Description . . . . .                       | 361 |
| 9.95.2   | Constructor & Destructor Documentation . . . . .     | 361 |
| 9.95.2.1 | PDFPage . . . . .                                    | 361 |

---

|           |  |     |
|-----------|--|-----|
| 9.95.2.2  | ~PDFPage . . . . .                               | 361 |
| 9.95.3    | Member Function Documentation . . . . .          | 361 |
| 9.95.3.1  | addObject . . . . .                              | 361 |
| 9.95.3.2  | bMargin . . . . .                                | 362 |
| 9.95.3.3  | height . . . . .                                 | 362 |
| 9.95.3.4  | lMargin . . . . .                                | 362 |
| 9.95.3.5  | mBottomEnd . . . . .                             | 362 |
| 9.95.3.6  | mHeight . . . . .                                | 362 |
| 9.95.3.7  | mRightEnd . . . . .                              | 362 |
| 9.95.3.8  | mWidth . . . . .                                 | 363 |
| 9.95.3.9  | rMargin . . . . .                                | 363 |
| 9.95.3.10 | tMargin . . . . .                                | 363 |
| 9.95.3.11 | width . . . . .                                  | 363 |
| 9.95.3.12 | writeContents . . . . .                          | 363 |
| 9.95.4    | Member Data Documentation . . . . .              | 364 |
| 9.95.4.1  | theObjects . . . . .                             | 364 |
| 9.96      | PDFPageObject Class Reference . . . . .          | 365 |
| 9.96.1    | Detailed Description . . . . .                   | 365 |
| 9.96.2    | Member Function Documentation . . . . .          | 365 |
| 9.96.2.1  | write . . . . .                                  | 365 |
| 9.97      | PDFPageObjectLine Class Reference . . . . .      | 366 |
| 9.97.1    | Detailed Description . . . . .                   | 366 |
| 9.97.2    | Constructor & Destructor Documentation . . . . . | 366 |
| 9.97.2.1  | PDFPageObjectLine . . . . .                      | 366 |
| 9.97.3    | Member Function Documentation . . . . .          | 367 |
| 9.97.3.1  | write . . . . .                                  | 367 |
| 9.97.4    | Member Data Documentation . . . . .              | 367 |
| 9.97.4.1  | theX1 . . . . .                                  | 367 |
| 9.97.4.2  | theX2 . . . . .                                  | 367 |
| 9.97.4.3  | theY1 . . . . .                                  | 367 |
| 9.97.4.4  | theY2 . . . . .                                  | 367 |

|   |     |
|---|-----|
| 9.98 PDFStructure Class Reference . . . . .             | 368 |
| 9.98.1 Detailed Description . . . . .                   | 368 |
| 9.98.2 Constructor & Destructor Documentation . . . . . | 368 |
| 9.98.2.1 PDFStructure . . . . .                         | 368 |
| 9.98.2.2 ~PDFStructure . . . . .                        | 369 |
| 9.98.3 Member Function Documentation . . . . .          | 369 |
| 9.98.3.1 addObject . . . . .                            | 369 |
| 9.98.3.2 newPage . . . . .                              | 369 |
| 9.98.3.3 preparePageContents . . . . .                  | 369 |
| 9.98.3.4 save . . . . .                                 | 369 |
| 9.98.4 Member Data Documentation . . . . .              | 370 |
| 9.98.4.1 thePages . . . . .                             | 370 |
| 9.99 Permutation Class Reference . . . . .              | 371 |
| 9.99.1 Detailed Description . . . . .                   | 375 |
| 9.99.2 Constructor & Destructor Documentation . . . . . | 375 |
| 9.99.2.1 Permutation . . . . .                          | 375 |
| 9.99.2.2 Permutation . . . . .                          | 375 |
| 9.99.2.3 Permutation . . . . .                          | 375 |
| 9.99.2.4 Permutation . . . . .                          | 375 |
| 9.99.3 Member Function Documentation . . . . .          | 375 |
| 9.99.3.1 _sub_meet . . . . .                            | 375 |
| 9.99.3.2 change . . . . .                               | 375 |
| 9.99.3.3 computeConjugacyClassRepresentative . . . . .  | 376 |
| 9.99.3.4 computeConjugator . . . . .                    | 376 |
| 9.99.3.5 CYCLE . . . . .                                | 376 |
| 9.99.3.6 difference . . . . .                           | 376 |
| 9.99.3.7 flip . . . . .                                 | 376 |
| 9.99.3.8 geodesic . . . . .                             | 376 |
| 9.99.3.9 geodesicWord . . . . .                         | 376 |
| 9.99.3.10 getCyclePermutation . . . . .                 | 376 |
| 9.99.3.11 getHalfTwistPermutation . . . . .             | 377 |

|           |  |     |
|-----------|--|-----|
| 9.99.3.12 | getVector . . . . .                                  | 377 |
| 9.99.3.13 | getWordPresentation . . . . .                        | 377 |
| 9.99.3.14 | increaseSize . . . . .                               | 377 |
| 9.99.3.15 | inverse . . . . .                                    | 377 |
| 9.99.3.16 | isTrivial . . . . .                                  | 377 |
| 9.99.3.17 | join2 . . . . .                                      | 377 |
| 9.99.3.18 | left_mult_by_cycle . . . . .                         | 378 |
| 9.99.3.19 | LeftGCD . . . . .                                    | 378 |
| 9.99.3.20 | LeftLCM . . . . .                                    | 378 |
| 9.99.3.21 | length . . . . .                                     | 378 |
| 9.99.3.22 | lr_multiply_by_cycles . . . . .                      | 378 |
| 9.99.3.23 | meet2 . . . . .                                      | 378 |
| 9.99.3.24 | mixable . . . . .                                    | 378 |
| 9.99.3.25 | operator!= . . . . .                                 | 379 |
| 9.99.3.26 | operator* . . . . .                                  | 379 |
| 9.99.3.27 | operator*= . . . . .                                 | 379 |
| 9.99.3.28 | operator- . . . . .                                  | 379 |
| 9.99.3.29 | operator< . . . . .                                  | 379 |
| 9.99.3.30 | operator== . . . . .                                 | 379 |
| 9.99.3.31 | operator[] . . . . .                                 | 379 |
| 9.99.3.32 | operator[] . . . . .                                 | 379 |
| 9.99.3.33 | power . . . . .                                      | 380 |
| 9.99.3.34 | prepare_pairs . . . . .                              | 380 |
| 9.99.3.35 | random . . . . .                                     | 380 |
| 9.99.3.36 | RightGCD . . . . .                                   | 380 |
| 9.99.3.37 | RightLCM . . . . .                                   | 380 |
| 9.99.3.38 | size . . . . .                                       | 380 |
| 9.99.3.39 | tinyFlip . . . . .                                   | 380 |
| 9.99.4    | Friends And Related Function Documentation . . . . . | 381 |
| 9.99.4.1  | BraidGroup . . . . .                                 | 381 |
| 9.99.4.2  | operator<< . . . . .                                 | 381 |

|  |     |
|--|-----|
| 9.99.5 Member Data Documentation . . . . .               | 381 |
| 9.99.5.1 theValue . . . . .                              | 381 |
| 9.100PermutationEnumerator Class Reference . . . . .     | 382 |
| 9.100.1 Detailed Description . . . . .                   | 383 |
| 9.100.2 Constructor & Destructor Documentation . . . . . | 383 |
| 9.100.2.1 PermutationEnumerator . . . . .                | 383 |
| 9.100.2.2 PermutationEnumerator . . . . .                | 383 |
| 9.100.3 Member Function Documentation . . . . .          | 383 |
| 9.100.3.1 finish . . . . .                               | 383 |
| 9.100.3.2 getPermutation . . . . .                       | 384 |
| 9.100.3.3 next . . . . .                                 | 384 |
| 9.100.3.4 operator= . . . . .                            | 384 |
| 9.100.3.5 seqComplete . . . . .                          | 384 |
| 9.100.3.6 seqLimit . . . . .                             | 384 |
| 9.100.3.7 seqOK . . . . .                                | 384 |
| 9.100.3.8 start . . . . .                                | 385 |
| 9.100.3.9 stepBack . . . . .                             | 385 |
| 9.100.3.10stepTo . . . . .                               | 385 |
| 9.100.4 Member Data Documentation . . . . .              | 386 |
| 9.100.4.1 theLength . . . . .                            | 386 |
| 9.100.4.2 theUsedElements . . . . .                      | 386 |
| 9.101Perturbation Class Reference . . . . .              | 387 |
| 9.101.1 Detailed Description . . . . .                   | 387 |
| 9.101.2 Member Function Documentation . . . . .          | 387 |
| 9.101.2.1 perturb . . . . .                              | 387 |
| 9.102PlanarGraphEdge Struct Reference . . . . .          | 388 |
| 9.102.1 Detailed Description . . . . .                   | 389 |
| 9.102.2 Constructor & Destructor Documentation . . . . . | 389 |
| 9.102.2.1 PlanarGraphEdge . . . . .                      | 389 |
| 9.102.2.2 PlanarGraphEdge . . . . .                      | 389 |
| 9.102.3 Member Function Documentation . . . . .          | 389 |

|           |   |     |
|-----------|---|-----|
| 9.102.3.1 | inverse . . . . .                                     | 389 |
| 9.102.3.2 | operator!= . . . . .                                  | 390 |
| 9.102.3.3 | operator< . . . . .                                   | 390 |
| 9.102.3.4 | operator== . . . . .                                  | 390 |
| 9.102.4   | Member Data Documentation . . . . .                   | 390 |
| 9.102.4.1 | theCell1 . . . . .                                    | 390 |
| 9.102.4.2 | theCell2 . . . . .                                    | 391 |
| 9.103     | PlanarGraphIntLabelledEdge Struct Reference . . . . . | 392 |
| 9.103.1   | Detailed Description . . . . .                        | 392 |
| 9.103.2   | Constructor & Destructor Documentation . . . . .      | 393 |
| 9.103.2.1 | PlanarGraphIntLabelledEdge . . . . .                  | 393 |
| 9.103.2.2 | PlanarGraphIntLabelledEdge . . . . .                  | 393 |
| 9.103.3   | Member Function Documentation . . . . .               | 393 |
| 9.103.3.1 | inverse . . . . .                                     | 393 |
| 9.103.3.2 | operator!= . . . . .                                  | 393 |
| 9.103.3.3 | operator< . . . . .                                   | 394 |
| 9.103.3.4 | operator== . . . . .                                  | 394 |
| 9.104     | PC::PowerCircuit Class Reference . . . . .            | 395 |
| 9.104.1   | Detailed Description . . . . .                        | 396 |
| 9.104.2   | Constructor & Destructor Documentation . . . . .      | 396 |
| 9.104.2.1 | PowerCircuit . . . . .                                | 396 |
| 9.104.2.2 | ~PowerCircuit . . . . .                               | 397 |
| 9.104.3   | Member Function Documentation . . . . .               | 397 |
| 9.104.3.1 | addMarkings . . . . .                                 | 397 |
| 9.104.3.2 | clone . . . . .                                       | 397 |
| 9.104.3.3 | cloneMarking . . . . .                                | 397 |
| 9.104.3.4 | cloneNode . . . . .                                   | 397 |
| 9.104.3.5 | compareMarkings . . . . .                             | 397 |
| 9.104.3.6 | connect . . . . .                                     | 398 |
| 9.104.3.7 | connectInv . . . . .                                  | 398 |
| 9.104.3.8 | copyMarking . . . . .                                 | 398 |



|  |     |
|--|-----|
| 9.104.3.9 createMarking . . . . .                            | 398 |
| 9.104.3.10 createMarking . . . . .                           | 398 |
| 9.104.3.11 createMarking . . . . .                           | 399 |
| 9.104.3.12 createMarkingFromNodes . . . . .                  | 399 |
| 9.104.3.13 createNode . . . . .                              | 399 |
| 9.104.3.14 decMarkingRefCount . . . . .                      | 399 |
| 9.104.3.15 draw . . . . .                                    | 399 |
| 9.104.3.16 getIncidentNode . . . . .                         | 399 |
| 9.104.3.17 getMarkingNodes . . . . .                         | 400 |
| 9.104.3.18 getMarkings . . . . .                             | 400 |
| 9.104.3.19 getNodes . . . . .                                | 400 |
| 9.104.3.20 getNodeSignInMarking . . . . .                    | 400 |
| 9.104.3.21 getNumEdges . . . . .                             | 400 |
| 9.104.3.22 getNumMarkings . . . . .                          | 400 |
| 9.104.3.23 getNumNodes . . . . .                             | 400 |
| 9.104.3.24 getRedNodeOrd . . . . .                           | 400 |
| 9.104.3.25 getReducedNode . . . . .                          | 401 |
| 9.104.3.26 getSmallestNodeInMarking . . . . .                | 401 |
| 9.104.3.27 getSuccMarking . . . . .                          | 401 |
| 9.104.3.28 incMarking . . . . .                              | 401 |
| 9.104.3.29 incMarkingRefCount . . . . .                      | 401 |
| 9.104.3.30 intersectMarkings . . . . .                       | 401 |
| 9.104.3.31 invMarking . . . . .                              | 402 |
| 9.104.3.32 isMarkingReduced . . . . .                        | 402 |
| 9.104.3.33 isSuccessorMarking . . . . .                      | 402 |
| 9.104.3.34 print . . . . .                                   | 402 |
| 9.104.3.35 printStatistics . . . . .                         | 402 |
| 9.104.3.36 reduce . . . . .                                  | 402 |
| 9.104.3.37 reduce . . . . .                                  | 403 |
| 9.104.3.38 remove . . . . .                                  | 403 |
| 9.104.4 Friends And Related Function Documentation . . . . . | 403 |

|   |     |
|---|-----|
| 9.104.4.1 Marking . . . . .                               | 403 |
| 9.104.4.2 Node . . . . .                                  | 403 |
| 9.105PC::PowerCircuitCompMatrix Class Reference . . . . . | 404 |
| 9.105.1 Detailed Description . . . . .                    | 406 |
| 9.105.2 Member Typedef Documentation . . . . .            | 407 |
| 9.105.2.1 MarkingVector . . . . .                         | 407 |
| 9.105.2.2 NodeVector . . . . .                            | 407 |
| 9.105.3 Member Enumeration Documentation . . . . .        | 407 |
| 9.105.3.1 MarkingType . . . . .                           | 407 |
| 9.105.4 Constructor & Destructor Documentation . . . . .  | 407 |
| 9.105.4.1 PowerCircuitCompMatrix . . . . .                | 407 |
| 9.105.4.2 ~PowerCircuitCompMatrix . . . . .               | 407 |
| 9.105.5 Member Function Documentation . . . . .           | 407 |
| 9.105.5.1 addMarkings . . . . .                           | 407 |
| 9.105.5.2 allocateNewMarking . . . . .                    | 408 |
| 9.105.5.3 calculateCompactRepresentation . . . . .        | 408 |
| 9.105.5.4 checkCycles . . . . .                           | 408 |
| 9.105.5.5 checkCyclesRecursive . . . . .                  | 408 |
| 9.105.5.6 checkMarkingValid . . . . .                     | 408 |
| 9.105.5.7 checkNodeValid . . . . .                        | 408 |
| 9.105.5.8 clone . . . . .                                 | 408 |
| 9.105.5.9 cloneMarking . . . . .                          | 408 |
| 9.105.5.10cloneNode . . . . .                             | 408 |
| 9.105.5.11cloneNode . . . . .                             | 409 |
| 9.105.5.12compactify . . . . .                            | 409 |
| 9.105.5.13compactifyFromBottom . . . . .                  | 409 |
| 9.105.5.14compareMarkings . . . . .                       | 409 |
| 9.105.5.15connect . . . . .                               | 409 |
| 9.105.5.16connectInv . . . . .                            | 409 |
| 9.105.5.17copyMarking . . . . .                           | 409 |
| 9.105.5.18createMarking . . . . .                         | 409 |

|   |     |
|---|-----|
| 9.105.5.19createMarking . . . . .                 | 410 |
| 9.105.5.20createMarking . . . . .                 | 410 |
| 9.105.5.21createNode . . . . .                    | 410 |
| 9.105.5.22decMarkingRefCount . . . . .            | 410 |
| 9.105.5.23deleteCol . . . . .                     | 411 |
| 9.105.5.24deleteMarking . . . . .                 | 411 |
| 9.105.5.25deleteNode . . . . .                    | 411 |
| 9.105.5.26extendTree . . . . .                    | 411 |
| 9.105.5.27findNewPosOfCompactCol . . . . .        | 411 |
| 9.105.5.28getIncidentNode . . . . .               | 411 |
| 9.105.5.29getMarkingNodes . . . . .               | 411 |
| 9.105.5.30getMarkings . . . . .                   | 411 |
| 9.105.5.31getMatrixUsage . . . . .                | 411 |
| 9.105.5.32getNodes . . . . .                      | 411 |
| 9.105.5.33getNodeSignInMarking . . . . .          | 412 |
| 9.105.5.34getRedNodeOrd . . . . .                 | 412 |
| 9.105.5.35getReducedNode . . . . .                | 412 |
| 9.105.5.36getSmallestNodeInMarking . . . . .      | 412 |
| 9.105.5.37getSuccMarking . . . . .                | 412 |
| 9.105.5.38getTreedNodePosToGivenCol . . . . .     | 412 |
| 9.105.5.39incMarking . . . . .                    | 412 |
| 9.105.5.40incMarkingRefCount . . . . .            | 412 |
| 9.105.5.41insertCompactColIntoTreed . . . . .     | 413 |
| 9.105.5.42insertCompactMarkingIntoTreed . . . . . | 413 |
| 9.105.5.43insertNewPowerOfTwoNode . . . . .       | 413 |
| 9.105.5.44insertNodeIntoTreed . . . . .           | 413 |
| 9.105.5.45intersectMarkings . . . . .             | 413 |
| 9.105.5.46invMarking . . . . .                    | 413 |
| 9.105.5.47isMarkingReduced . . . . .              | 413 |
| 9.105.5.48isSuccessorMarking . . . . .            | 413 |
| 9.105.5.49mergeCols . . . . .                     | 414 |

|            |  |     |
|------------|--|-----|
| 9.105.5.50 | moveColsToTreed                        | 414 |
| 9.105.5.51 | moveNodeIntoTreedPartOfMatrix          | 414 |
| 9.105.5.52 | newCopyColMarking                      | 414 |
| 9.105.5.53 | newDoubleNode                          | 414 |
| 9.105.5.54 | newNodeFromMarking                     | 414 |
| 9.105.5.55 | newOneMarking                          | 414 |
| 9.105.5.56 | newOneNode                             | 414 |
| 9.105.5.57 | newUnitMarking                         | 414 |
| 9.105.5.58 | newZeroMarking                         | 414 |
| 9.105.5.59 | print                                  | 414 |
| 9.105.5.60 | printStatistics                        | 414 |
| 9.105.5.61 | reduce                                 | 415 |
| 9.105.5.62 | reduce                                 | 415 |
| 9.105.5.63 | remove                                 | 415 |
| 9.105.5.64 | removeDoubleNodesFromMarkings          | 415 |
| 9.105.5.65 | separateMarkingFromCol                 | 415 |
| 9.105.5.66 | setBV                                  | 415 |
| 9.105.5.67 | topSortNode                            | 415 |
| 9.105.6    | Member Data Documentation              | 415 |
| 9.105.6.1  | firstDeletedMarking                    | 415 |
| 9.105.6.2  | firstDeletedNode                       | 416 |
| 9.105.6.3  | markings                               | 416 |
| 9.105.6.4  | matrix                                 | 416 |
| 9.105.6.5  | nodes                                  | 416 |
| 9.105.6.6  | numTreedCols                           | 416 |
| 9.105.6.7  | numTreedNodes                          | 416 |
| 9.105.6.8  | totalNumMarkings                       | 416 |
| 9.105.6.9  | totalNumNodes                          | 416 |
| 9.106      | PC::PowerCircuitGraph Class Reference  | 418 |
| 9.106.1    | Detailed Description                   | 420 |
| 9.106.2    | Constructor & Destructor Documentation | 421 |

---

|  |     |
|--|-----|
| 9.106.2.1 PowerCircuitGraph . . . . .              | 421 |
| 9.106.2.2 ~PowerCircuitGraph . . . . .             | 421 |
| 9.106.3 Member Function Documentation . . . . .    | 421 |
| 9.106.3.1 addMarkings . . . . .                    | 421 |
| 9.106.3.2 calculateCompactRepresentation . . . . . | 421 |
| 9.106.3.3 checkConsistency . . . . .               | 421 |
| 9.106.3.4 checkCycles . . . . .                    | 421 |
| 9.106.3.5 checkCyclesRecursive . . . . .           | 421 |
| 9.106.3.6 checkMarkingValid . . . . .              | 421 |
| 9.106.3.7 checkNodeValid . . . . .                 | 421 |
| 9.106.3.8 clone . . . . .                          | 421 |
| 9.106.3.9 cloneMarking . . . . .                   | 421 |
| 9.106.3.10cloneNode . . . . .                      | 422 |
| 9.106.3.11compare . . . . .                        | 422 |
| 9.106.3.12compareMarkings . . . . .                | 422 |
| 9.106.3.13compareNodesLessThan . . . . .           | 422 |
| 9.106.3.14connect . . . . .                        | 422 |
| 9.106.3.15connectInv . . . . .                     | 422 |
| 9.106.3.16copyMarking . . . . .                    | 423 |
| 9.106.3.17createMarking . . . . .                  | 423 |
| 9.106.3.18createMarking . . . . .                  | 423 |
| 9.106.3.19createMarking . . . . .                  | 423 |
| 9.106.3.20createMarking . . . . .                  | 423 |
| 9.106.3.21createNode . . . . .                     | 423 |
| 9.106.3.22decMarkingRefCount . . . . .             | 423 |
| 9.106.3.23deleteMarking . . . . .                  | 424 |
| 9.106.3.24deleteNode . . . . .                     | 424 |
| 9.106.3.25extendTree . . . . .                     | 424 |
| 9.106.3.26findNewPosOfReducedNode . . . . .        | 424 |
| 9.106.3.27findNodeInList . . . . .                 | 424 |
| 9.106.3.28getIncidentNode . . . . .                | 424 |

|            |                                    |     |
|------------|------------------------------------|-----|
| 9.106.3.29 | getMarkingNodes . . . . .          | 424 |
| 9.106.3.30 | getMarkings . . . . .              | 424 |
| 9.106.3.31 | getMatrixUsage . . . . .           | 424 |
| 9.106.3.32 | getNodes . . . . .                 | 424 |
| 9.106.3.33 | getNodeSignInMarking . . . . .     | 425 |
| 9.106.3.34 | getRedNodeOrd . . . . .            | 425 |
| 9.106.3.35 | getReducedNode . . . . .           | 425 |
| 9.106.3.36 | getSmallestNodeInMarking . . . . . | 425 |
| 9.106.3.37 | getSuccMarking . . . . .           | 425 |
| 9.106.3.38 | inc . . . . .                      | 425 |
| 9.106.3.39 | incMarking . . . . .               | 425 |
| 9.106.3.40 | incMarkingRefCount . . . . .       | 425 |
| 9.106.3.41 | insertNewPowerOfTwoNode . . . . .  | 426 |
| 9.106.3.42 | insertNodeIntoReduced . . . . .    | 426 |
| 9.106.3.43 | intersectMarkings . . . . .        | 426 |
| 9.106.3.44 | invMarking . . . . .               | 426 |
| 9.106.3.45 | isMarkingReduced . . . . .         | 426 |
| 9.106.3.46 | isSuccessorMarking . . . . .       | 426 |
| 9.106.3.47 | markSucessors . . . . .            | 427 |
| 9.106.3.48 | moveNodeIntoReducedPart . . . . .  | 427 |
| 9.106.3.49 | newCopyMarking . . . . .           | 427 |
| 9.106.3.50 | newDoubleNode . . . . .            | 427 |
| 9.106.3.51 | newMarking . . . . .               | 427 |
| 9.106.3.52 | newNode . . . . .                  | 427 |
| 9.106.3.53 | newOneMarking . . . . .            | 427 |
| 9.106.3.54 | newOneNode . . . . .               | 427 |
| 9.106.3.55 | newUnitMarking . . . . .           | 427 |
| 9.106.3.56 | newZeroMarking . . . . .           | 427 |
| 9.106.3.57 | print . . . . .                    | 427 |
| 9.106.3.58 | printMarking . . . . .             | 428 |
| 9.106.3.59 | printMarking . . . . .             | 428 |

---

|            |  |     |
|------------|--|-----|
| 9.106.3.60 | printStatistics . . . . .                        | 428 |
| 9.106.3.61 | reduce . . . . .                                 | 428 |
| 9.106.3.62 | reduce . . . . .                                 | 428 |
| 9.106.3.63 | reduce . . . . .                                 | 428 |
| 9.106.3.64 | remove . . . . .                                 | 428 |
| 9.106.3.65 | removeDoubleNodesFromMarkings . . . . .          | 429 |
| 9.106.3.66 | setBV . . . . .                                  | 429 |
| 9.106.3.67 | sortNodeList . . . . .                           | 429 |
| 9.106.3.68 | topSortNode . . . . .                            | 429 |
| 9.106.4    | Member Data Documentation . . . . .              | 429 |
| 9.106.4.1  | firstDeletedMarking . . . . .                    | 429 |
| 9.106.4.2  | firstDeletedNode . . . . .                       | 429 |
| 9.106.4.3  | markings . . . . .                               | 429 |
| 9.106.4.4  | nodeOrder . . . . .                              | 429 |
| 9.106.4.5  | nodes . . . . .                                  | 429 |
| 9.106.4.6  | numMarkings . . . . .                            | 430 |
| 9.106.4.7  | numNodes . . . . .                               | 430 |
| 9.106.4.8  | numReducedNodes . . . . .                        | 430 |
| 9.107      | PowerWord Class Reference . . . . .              | 431 |
| 9.107.1    | Detailed Description . . . . .                   | 432 |
| 9.107.2    | Member Typedef Documentation . . . . .           | 432 |
| 9.107.2.1  | const_iterator . . . . .                         | 432 |
| 9.107.2.2  | iterator . . . . .                               | 433 |
| 9.107.2.3  | PII . . . . .                                    | 433 |
| 9.107.3    | Constructor & Destructor Documentation . . . . . | 433 |
| 9.107.3.1  | PowerWord . . . . .                              | 433 |
| 9.107.3.2  | PowerWord . . . . .                              | 433 |
| 9.107.3.3  | PowerWord . . . . .                              | 433 |
| 9.107.3.4  | PowerWord . . . . .                              | 433 |
| 9.107.3.5  | PowerWord . . . . .                              | 433 |
| 9.107.3.6  | PowerWord . . . . .                              | 433 |

|   |     |
|---|-----|
| 9.107.4 Member Function Documentation . . . . . | 433 |
| 9.107.4.1 begin . . . . .                       | 433 |
| 9.107.4.2 begin . . . . .                       | 434 |
| 9.107.4.3 cyclicallyPermute . . . . .           | 434 |
| 9.107.4.4 cyclicallyReduce . . . . .            | 434 |
| 9.107.4.5 cyclicallyReduce . . . . .            | 434 |
| 9.107.4.6 cyclicallyReduceWord . . . . .        | 434 |
| 9.107.4.7 cyclicallyReduceWord . . . . .        | 434 |
| 9.107.4.8 cyclicLeftShift . . . . .             | 434 |
| 9.107.4.9 cyclicRightShift . . . . .            | 435 |
| 9.107.4.10doesContain . . . . .                 | 435 |
| 9.107.4.11lend . . . . .                        | 435 |
| 9.107.4.12end . . . . .                         | 435 |
| 9.107.4.13exponentSum . . . . .                 | 435 |
| 9.107.4.14freelyReduce . . . . .                | 435 |
| 9.107.4.15freelyReduce . . . . .                | 435 |
| 9.107.4.16getList . . . . .                     | 436 |
| 9.107.4.17getList . . . . .                     | 436 |
| 9.107.4.18getPower . . . . .                    | 436 |
| 9.107.4.19initialSegment . . . . .              | 436 |
| 9.107.4.20insert . . . . .                      | 436 |
| 9.107.4.21inverse . . . . .                     | 436 |
| 9.107.4.22isIn . . . . .                        | 436 |
| 9.107.4.23length . . . . .                      | 437 |
| 9.107.4.24operator!= . . . . .                  | 437 |
| 9.107.4.25operator* . . . . .                   | 437 |
| 9.107.4.26operator*= . . . . .                  | 437 |
| 9.107.4.27operator- . . . . .                   | 437 |
| 9.107.4.28operator< . . . . .                   | 437 |
| 9.107.4.29operator== . . . . .                  | 437 |
| 9.107.4.30operator> . . . . .                   | 438 |



|  |     |
|--|-----|
| 9.107.4.31power . . . . .                                    | 438 |
| 9.107.4.32printOn . . . . .                                  | 438 |
| 9.107.4.33push_back . . . . .                                | 438 |
| 9.107.4.34push_back . . . . .                                | 438 |
| 9.107.4.35push_front . . . . .                               | 438 |
| 9.107.4.36push_front . . . . .                               | 438 |
| 9.107.4.37randomWord . . . . .                               | 439 |
| 9.107.4.38segment . . . . .                                  | 439 |
| 9.107.4.39terminalSegment . . . . .                          | 439 |
| 9.107.5 Friends And Related Function Documentation . . . . . | 439 |
| 9.107.5.1 operator<< . . . . .                               | 439 |
| 9.108PowerWordRep Class Reference . . . . .                  | 440 |
| 9.108.1 Detailed Description . . . . .                       | 441 |
| 9.108.2 Member Typedef Documentation . . . . .               | 441 |
| 9.108.2.1 PII . . . . .                                      | 441 |
| 9.108.3 Constructor & Destructor Documentation . . . . .     | 441 |
| 9.108.3.1 PowerWordRep . . . . .                             | 441 |
| 9.108.3.2 PowerWordRep . . . . .                             | 442 |
| 9.108.3.3 PowerWordRep . . . . .                             | 442 |
| 9.108.3.4 PowerWordRep . . . . .                             | 442 |
| 9.108.3.5 PowerWordRep . . . . .                             | 442 |
| 9.108.3.6 PowerWordRep . . . . .                             | 442 |
| 9.108.3.7 PowerWordRep . . . . .                             | 442 |
| 9.108.4 Member Function Documentation . . . . .              | 442 |
| 9.108.4.1 clone . . . . .                                    | 442 |
| 9.108.4.2 cyclicallyPermute . . . . .                        | 442 |
| 9.108.4.3 cyclicallyReduce . . . . .                         | 442 |
| 9.108.4.4 cyclicallyReduce . . . . .                         | 442 |
| 9.108.4.5 cyclicLeftShift . . . . .                          | 442 |
| 9.108.4.6 cyclicRightShift . . . . .                         | 442 |
| 9.108.4.7 doesContain . . . . .                              | 442 |

|   |     |
|---|-----|
| 9.108.4.8 exponentSum . . . . .                                     | 443 |
| 9.108.4.9 getList . . . . .   | 443 |
| 9.108.4.10getList . . . . .   | 443 |
| 9.108.4.11getPower . . . . .  | 443 |
| 9.108.4.12initialSegment . . . . .                                  | 443 |
| 9.108.4.13insert . . . . .  | 443 |
| 9.108.4.14inverse . . . . .   | 443 |
| 9.108.4.15isIn . . . . .  | 443 |
| 9.108.4.16length . . . . .  | 443 |
| 9.108.4.17operator* . . . . .                                       | 444 |
| 9.108.4.18operator*= . . . . .                                      | 444 |
| 9.108.4.19operator< . . . . .                                       | 444 |
| 9.108.4.20operator= . . . . .                                       | 444 |
| 9.108.4.21operator== . . . . .                                      | 444 |
| 9.108.4.22operator> . . . . .                                       | 444 |
| 9.108.4.23printOn . . . . .   | 444 |
| 9.108.4.24pushGeneratorBack . . . . .                               | 444 |
| 9.108.4.25pushGeneratorFront . . . . .                              | 444 |
| 9.108.4.26segment . . . . .   | 445 |
| 9.108.4.27terminalSegment . . . . .                                 | 445 |
| 9.108.5 Friends And Related Function Documentation . . . . .        | 445 |
| 9.108.5.1 PowerWord . . . . .                                       | 445 |
| 9.108.6 Member Data Documentation . . . . .                         | 445 |
| 9.108.6.1 theElements . . . . .                                     | 445 |
| 9.108.6.2 theLength . . . . .                                       | 445 |
| 9.109StraightLineProgramWord::Production Struct Reference . . . . . | 446 |
| 9.109.1 Detailed Description . . . . .                              | 446 |
| 9.109.2 Constructor & Destructor Documentation . . . . .            | 446 |
| 9.109.2.1 Production . . . . .                                      | 446 |
| 9.109.3 Member Data Documentation . . . . .                         | 446 |
| 9.109.3.1 reduced . . . . .   | 446 |

|   |     |
|---|-----|
| 9.109.3.2 theHeight . . . . .                                       | 446 |
| 9.109.3.3 theLength . . . . .                                       | 447 |
| 9.109.3.4 theTerm1 . . . . .  | 447 |
| 9.109.3.5 theTerm2 . . . . .  | 447 |
| 9.110QuadEquationTransformationGraph Class Reference . . . . .      | 448 |
| 9.110.1 Detailed Description . . . . .                              | 449 |
| 9.110.2 Constructor & Destructor Documentation . . . . .            | 449 |
| 9.110.2.1 QuadEquationTransformationGraph . . . . .                 | 449 |
| 9.110.3 Member Function Documentation . . . . .                     | 449 |
| 9.110.3.1 applyAdjointTransformation . . . . .                      | 449 |
| 9.110.3.2 extend . . . . .  | 449 |
| 9.110.3.3 getGraph . . . . .  | 449 |
| 9.110.3.4 getNeighbours . . . . .                                   | 449 |
| 9.110.3.5 isDone . . . . .  | 449 |
| 9.110.3.6 isTrivialSolution . . . . .                               | 450 |
| 9.110.3.7 solutionFound . . . . .                                   | 450 |
| 9.110.4 Member Data Documentation . . . . .                         | 450 |
| 9.110.4.1 equationsInProgress . . . . .                             | 450 |
| 9.110.4.2 hasSolution . . . . .                                     | 450 |
| 9.110.4.3 processedEquations . . . . .                              | 450 |
| 9.110.4.4 theEquation . . . . .                                     | 450 |
| 9.110.4.5 theGraph . . . . .  | 450 |
| 9.111quadruple< T1, T2, T3, T4 > Class Template Reference . . . . . | 452 |
| 9.111.1 Detailed Description . . . . .                              | 452 |
| 9.111.2 Constructor & Destructor Documentation . . . . .            | 452 |
| 9.111.2.1 quadruple . . . . .                                       | 452 |
| 9.111.3 Member Function Documentation . . . . .                     | 453 |
| 9.111.3.1 operator< . . . . .                                       | 453 |
| 9.111.4 Friends And Related Function Documentation . . . . .        | 453 |
| 9.111.4.1 operator<< . . . . .                                      | 453 |
| 9.111.5 Member Data Documentation . . . . .                         | 453 |

|   |     |
|---|-----|
| 9.111.5.1 first . . . . .   | 453 |
| 9.111.5.2 fourth . . . . .  | 453 |
| 9.111.5.3 second . . . . .  | 453 |
| 9.111.5.4 third . . . . .   | 453 |
| 9.112quintuple< T1, T2, T3, T4, T5 > Class Template Reference . . . . . | 455 |
| 9.112.1 Detailed Description . . . . .                                  | 455 |
| 9.112.2 Constructor & Destructor Documentation . . . . .                | 455 |
| 9.112.2.1 quintuple . . . . .   | 455 |
| 9.112.3 Member Function Documentation . . . . .                         | 456 |
| 9.112.3.1 operator< . . . . .   | 456 |
| 9.112.4 Friends And Related Function Documentation . . . . .            | 456 |
| 9.112.4.1 operator<< . . . . .  | 456 |
| 9.112.5 Member Data Documentation . . . . .                             | 456 |
| 9.112.5.1 fifth . . . . .   | 456 |
| 9.112.5.2 first . . . . .   | 456 |
| 9.112.5.3 fourth . . . . .  | 456 |
| 9.112.5.4 second . . . . .  | 456 |
| 9.112.5.5 third . . . . .   | 457 |
| 9.113RandLib Class Reference . . . . .                                  | 458 |
| 9.113.1 Detailed Description . . . . .                                  | 458 |
| 9.113.2 Member Function Documentation . . . . .                         | 458 |
| 9.113.2.1 chiPValue . . . . .   | 458 |
| 9.113.3 Member Data Documentation . . . . .                             | 458 |
| 9.113.3.1 ur . . . . .  | 458 |
| 9.114RandLibURG Class Reference . . . . .                               | 460 |
| 9.114.1 Detailed Description . . . . .                                  | 460 |
| 9.114.2 Constructor & Destructor Documentation . . . . .                | 461 |
| 9.114.2.1 RandLibURG . . . . .  | 461 |
| 9.114.2.2 RandLibURG . . . . .  | 461 |
| 9.114.3 Member Function Documentation . . . . .                         | 461 |
| 9.114.3.1 getseed . . . . .   | 461 |

|  |     |
|--|-----|
| 9.114.3.2 irand . . . . .                                | 461 |
| 9.114.3.3 rand . . . . .                                 | 462 |
| 9.114.3.4 rand . . . . .                                 | 462 |
| 9.114.3.5 reset . . . . .                                | 462 |
| 9.114.3.6 reset . . . . .                                | 462 |
| 9.114.3.7 setSeedPID . . . . .                           | 462 |
| 9.115 RandomPairGenerator Class Reference . . . . .      | 463 |
| 9.115.1 Detailed Description . . . . .                   | 463 |
| 9.115.2 Constructor & Destructor Documentation . . . . . | 463 |
| 9.115.2.1 RandomPairGenerator . . . . .                  | 463 |
| 9.115.3 Member Function Documentation . . . . .          | 464 |
| 9.115.3.1 getFalsePair . . . . .                         | 464 |
| 9.115.3.2 getTruePair . . . . .                          | 464 |
| 9.115.4 Member Data Documentation . . . . .              | 464 |
| 9.115.4.1 Len . . . . .                                  | 464 |
| 9.115.4.2 nGens . . . . .                                | 464 |
| 9.116 RefCounter Class Reference . . . . .               | 466 |
| 9.116.1 Detailed Description . . . . .                   | 466 |
| 9.116.2 Member Typedef Documentation . . . . .           | 466 |
| 9.116.2.1 refCounterType . . . . .                       | 466 |
| 9.116.3 Constructor & Destructor Documentation . . . . . | 467 |
| 9.116.3.1 RefCounter . . . . .                           | 467 |
| 9.116.3.2 RefCounter . . . . .                           | 467 |
| 9.116.4 Member Function Documentation . . . . .          | 467 |
| 9.116.4.1 addRef . . . . .                               | 467 |
| 9.116.4.2 delRef . . . . .                               | 467 |
| 9.116.4.3 lastRef . . . . .                              | 467 |
| 9.116.4.4 operator= . . . . .                            | 467 |
| 9.116.4.5 sharedRef . . . . .                            | 467 |
| 9.116.5 Member Data Documentation . . . . .              | 467 |
| 9.116.5.1 xrefs . . . . .                                | 467 |

|  |     |
|--|-----|
| 9.117RestrictedWhiteheadAutoSet Class Reference . . . . .          | 469 |
| 9.117.1 Detailed Description . . . . .                             | 469 |
| 9.117.2 Constructor & Destructor Documentation . . . . .           | 470 |
| 9.117.2.1 RestrictedWhiteheadAutoSet . . . . .                     | 470 |
| 9.117.2.2 ~RestrictedWhiteheadAutoSet . . . . .                    | 470 |
| 9.117.3 Member Function Documentation . . . . .                    | 470 |
| 9.117.3.1 getRandomAuto . . . . .                                  | 470 |
| 9.117.3.2 getSet . . . . .   | 470 |
| 9.117.4 Member Data Documentation . . . . .                        | 471 |
| 9.117.4.1 nGens . . . . .  | 471 |
| 9.117.4.2 theSet . . . . .   | 471 |
| 9.118RGB Class Reference . . . . .                                 | 472 |
| 9.118.1 Detailed Description . . . . .                             | 472 |
| 9.118.2 Constructor & Destructor Documentation . . . . .           | 472 |
| 9.118.2.1 RGB . . . . .  | 472 |
| 9.118.2.2 RGB . . . . .  | 472 |
| 9.118.3 Member Data Documentation . . . . .                        | 472 |
| 9.118.3.1 B . . . . .  | 472 |
| 9.118.3.2 G . . . . .  | 472 |
| 9.118.3.3 R . . . . .  | 473 |
| 9.119PC::Row Struct Reference . . . . .                            | 474 |
| 9.119.1 Detailed Description . . . . .                             | 474 |
| 9.119.2 Constructor & Destructor Documentation . . . . .           | 474 |
| 9.119.2.1 Row . . . . .  | 474 |
| 9.119.3 Member Data Documentation . . . . .                        | 474 |
| 9.119.3.1 rowid . . . . .  | 474 |
| 9.119.3.2 UNDEFINED . . . . .                                      | 475 |
| 9.120PC::PowerCircuitCompMatrix::RowHdr Struct Reference . . . . . | 476 |
| 9.120.1 Detailed Description . . . . .                             | 476 |
| 9.120.2 Constructor & Destructor Documentation . . . . .           | 476 |
| 9.120.2.1 RowHdr . . . . .   | 476 |

---

|  |     |
|--|-----|
| 9.120.2.2 RowHdr . . . . .                                 | 476 |
| 9.120.3 Member Data Documentation . . . . .                | 476 |
| 9.120.3.1 BV . . . . .                                     | 476 |
| 9.120.3.2 indexInNodes . . . . .                           | 476 |
| 9.120.3.3 lambdaMarking . . . . .                          | 477 |
| 9.121 ShftConjKeyInstance Class Reference . . . . .        | 478 |
| 9.121.1 Detailed Description . . . . .                     | 478 |
| 9.121.2 Constructor & Destructor Documentation . . . . .   | 479 |
| 9.121.2.1 ShftConjKeyInstance . . . . .                    | 479 |
| 9.121.3 Member Function Documentation . . . . .            | 479 |
| 9.121.3.1 getBraidRank . . . . .                           | 479 |
| 9.121.3.2 getPrivateKey . . . . .                          | 479 |
| 9.121.3.3 getPublicKey . . . . .                           | 479 |
| 9.121.3.4 random . . . . .                                 | 479 |
| 9.121.4 Member Data Documentation . . . . .                | 480 |
| 9.121.4.1 thePrivateKey . . . . .                          | 480 |
| 9.121.4.2 thePublicKey . . . . .                           | 480 |
| 9.121.4.3 theRank . . . . .                                | 480 |
| 9.122 ShftConjKeyInstanceGarside Class Reference . . . . . | 481 |
| 9.122.1 Detailed Description . . . . .                     | 482 |
| 9.122.2 Constructor & Destructor Documentation . . . . .   | 482 |
| 9.122.2.1 ShftConjKeyInstanceGarside . . . . .             | 482 |
| 9.122.3 Member Function Documentation . . . . .            | 482 |
| 9.122.3.1 getBraidRank . . . . .                           | 482 |
| 9.122.3.2 getPrivateKey . . . . .                          | 482 |
| 9.122.3.3 getPublicKey . . . . .                           | 482 |
| 9.122.3.4 random . . . . .                                 | 483 |
| 9.122.4 Member Data Documentation . . . . .                | 483 |
| 9.122.4.1 thePrivateKey . . . . .                          | 483 |
| 9.122.4.2 thePublicKey . . . . .                           | 483 |
| 9.122.4.3 theRank . . . . .                                | 483 |

|   |     |
|---|-----|
| 9.123PC::SignMatrix< RowHdr, ColHdr, INTERNAL_TYPE > Class Template Reference . . . . . | 484 |
| 9.123.1 Detailed Description . . . . .  | 486 |
| 9.123.2 Constructor & Destructor Documentation . . . . .                                | 486 |
| 9.123.2.1 SignMatrix . . . . .  | 486 |
| 9.123.2.2 ~SignMatrix . . . . .   | 486 |
| 9.123.3 Member Function Documentation . . . . .   | 487 |
| 9.123.3.1 addToCol . . . . .  | 487 |
| 9.123.3.2 clone . . . . .   | 487 |
| 9.123.3.3 colHdr . . . . .  | 488 |
| 9.123.3.4 colHdr . . . . .  | 488 |
| 9.123.3.5 columnsEqual . . . . .  | 488 |
| 9.123.3.6 deleteCol . . . . .   | 488 |
| 9.123.3.7 deleteRow . . . . .   | 489 |
| 9.123.3.8 extend . . . . .  | 489 |
| 9.123.3.9 free . . . . .  | 490 |
| 9.123.3.10 getCol . . . . .   | 490 |
| 9.123.3.11 getColNumber . . . . .   | 490 |
| 9.123.3.12 getNumCols . . . . .   | 491 |
| 9.123.3.13 getNumRows . . . . .   | 491 |
| 9.123.3.14 getRow . . . . .   | 491 |
| 9.123.3.15 getRowNumber . . . . .   | 491 |
| 9.123.3.16 insertCol . . . . .  | 492 |
| 9.123.3.17 insertColCopy . . . . .  | 492 |
| 9.123.3.18 insertColSum . . . . .   | 493 |
| 9.123.3.19 insertIntersectionCol . . . . .  | 493 |
| 9.123.3.20 insertInvCol . . . . .   | 493 |
| 9.123.3.21 insertMatrix . . . . .   | 494 |
| 9.123.3.22 insertRow . . . . .  | 494 |
| 9.123.3.23 insertRowCopy . . . . .  | 495 |
| 9.123.3.24 insertUnitCol . . . . .  | 495 |



|  |     |
|--|-----|
| 9.123.3.25insertZeroCol . . . . .                      | 496 |
| 9.123.3.26insertZeroRow . . . . .                      | 496 |
| 9.123.3.27moveCol . . . . .                            | 496 |
| 9.123.3.28moveRow . . . . .                            | 497 |
| 9.123.3.29operator() . . . . .                         | 497 |
| 9.123.3.30operator() . . . . .                         | 497 |
| 9.123.3.31operator() . . . . .                         | 497 |
| 9.123.3.32operator() . . . . .                         | 498 |
| 9.123.3.33printMatrix . . . . .                        | 498 |
| 9.123.3.34rowHdr . . . . .                             | 499 |
| 9.123.3.35rowHdr . . . . .                             | 499 |
| 9.123.3.36setEntry . . . . .                           | 499 |
| 9.123.3.37setEntry . . . . .                           | 499 |
| 9.123.3.38setEntry . . . . .                           | 500 |
| 9.123.3.39setEntry . . . . .                           | 500 |
| 9.123.4 Member Data Documentation . . . . .            | 500 |
| 9.123.4.1 colHdrs . . . . .                            | 500 |
| 9.123.4.2 EXTEND_CONDITION . . . . .                   | 501 |
| 9.123.4.3 EXTEND_FACTOR . . . . .                      | 501 |
| 9.123.4.4 firstUnusedCol . . . . .                     | 501 |
| 9.123.4.5 firstUnusedRow . . . . .                     | 502 |
| 9.123.4.6 INTERNAL_SIZE . . . . .                      | 502 |
| 9.123.4.7 mask1 . . . . .                              | 502 |
| 9.123.4.8 mask2 . . . . .                              | 503 |
| 9.123.4.9 matrix . . . . .                             | 503 |
| 9.123.4.10maxNumCols . . . . .                         | 503 |
| 9.123.4.11maxNumRows . . . . .                         | 504 |
| 9.123.4.12numCols . . . . .                            | 505 |
| 9.123.4.13numRows . . . . .                            | 505 |
| 9.123.4.14rowHdrs . . . . .                            | 506 |
| 9.124StraightLineProgramWord Class Reference . . . . . | 507 |

|  |     |
|--|-----|
| 9.124.1 Detailed Description . . . . .                   | 511 |
| 9.124.2 Constructor & Destructor Documentation . . . . . | 511 |
| 9.124.2.1 StraightLineProgramWord . . . . .              | 511 |
| 9.124.2.2 StraightLineProgramWord . . . . .              | 511 |
| 9.124.2.3 StraightLineProgramWord . . . . .              | 511 |
| 9.124.3 Member Function Documentation . . . . .          | 512 |
| 9.124.3.1 applyMapping . . . . .                         | 512 |
| 9.124.3.2 assertionType . . . . .                        | 512 |
| 9.124.3.3 cancellationLength . . . . .                   | 512 |
| 9.124.3.4 equal . . . . .                                | 512 |
| 9.124.3.5 extendTerminals . . . . .                      | 512 |
| 9.124.3.6 getGenerator . . . . .                         | 512 |
| 9.124.3.7 getWord . . . . .                              | 513 |
| 9.124.3.8 getWord . . . . .                              | 513 |
| 9.124.3.9 height_rule . . . . .                          | 513 |
| 9.124.3.10 initialSegment . . . . .                      | 513 |
| 9.124.3.11 invertProductionPair . . . . .                | 513 |
| 9.124.3.12 leftGCDLength . . . . .                       | 514 |
| 9.124.3.13 leftGCDLength . . . . .                       | 514 |
| 9.124.3.14 length . . . . .                              | 514 |
| 9.124.3.15 length_rule . . . . .                         | 514 |
| 9.124.3.16 map_rules . . . . .                           | 514 |
| 9.124.3.17 max_rule_number . . . . .                     | 515 |
| 9.124.3.18 operator* . . . . .                           | 515 |
| 9.124.3.19 operator- . . . . .                           | 515 |
| 9.124.3.20 operator= . . . . .                           | 515 |
| 9.124.3.21 operator[] . . . . .                          | 515 |
| 9.124.3.22 order_vertices . . . . .                      | 515 |
| 9.124.3.23 order_vertices . . . . .                      | 516 |
| 9.124.3.24 reduce . . . . .                              | 516 |
| 9.124.3.25 reduce_rule . . . . .                         | 516 |

|            |  |     |
|------------|--|-----|
| 9.124.3.26 | simplify . . . . .                                   | 516 |
| 9.124.3.27 | splitAssertion . . . . .                             | 516 |
| 9.124.3.28 | terminalSegment . . . . .                            | 516 |
| 9.124.3.29 | truncateVertexLeft . . . . .                         | 517 |
| 9.124.3.30 | truncateVertexRight . . . . .                        | 517 |
| 9.124.3.31 | update_height . . . . .                              | 517 |
| 9.124.3.32 | update_heights . . . . .                             | 517 |
| 9.124.3.33 | update_length . . . . .                              | 517 |
| 9.124.3.34 | update_lengths . . . . .                             | 517 |
| 9.124.4    | Friends And Related Function Documentation . . . . . | 518 |
| 9.124.4.1  | operator<< . . . . .                                 | 518 |
| 9.124.5    | Member Data Documentation . . . . .                  | 518 |
| 9.124.5.1  | theRoot . . . . .                                    | 518 |
| 9.124.5.2  | theRules . . . . .                                   | 518 |
| 9.124.5.3  | theTerminals . . . . .                               | 518 |
| 9.125      | StringScrambler Class Reference . . . . .            | 519 |
| 9.125.1    | Detailed Description . . . . .                       | 519 |
| 9.125.2    | Member Function Documentation . . . . .              | 519 |
| 9.125.2.1  | fracChanged . . . . .                                | 519 |
| 9.125.2.2  | scramble . . . . .                                   | 519 |
| 9.126      | StringSimilarityMeasure Class Reference . . . . .    | 520 |
| 9.126.1    | Detailed Description . . . . .                       | 520 |
| 9.126.2    | Member Function Documentation . . . . .              | 520 |
| 9.126.2.1  | measure . . . . .                                    | 520 |
| 9.127      | SubgroupFG Class Reference . . . . .                 | 522 |
| 9.127.1    | Detailed Description . . . . .                       | 524 |
| 9.127.2    | Constructor & Destructor Documentation . . . . .     | 524 |
| 9.127.2.1  | SubgroupFG . . . . .                                 | 524 |
| 9.127.2.2  | SubgroupFG . . . . .                                 | 524 |
| 9.127.2.3  | SubgroupFG . . . . .                                 | 524 |
| 9.127.2.4  | SubgroupFG . . . . .                                 | 524 |

|   |     |
|---|-----|
| 9.127.3 Member Function Documentation . . . . . | 524 |
| 9.127.3.1 areConjugate . . . . .                | 524 |
| 9.127.3.2 centralizer . . . . .                 | 524 |
| 9.127.3.3 checkIsomorphism . . . . .            | 525 |
| 9.127.3.4 computeFSA . . . . .                  | 525 |
| 9.127.3.5 computeNielsenGenerators . . . . .    | 525 |
| 9.127.3.6 doesBelong . . . . .                  | 525 |
| 9.127.3.7 express . . . . .                     | 525 |
| 9.127.3.8 getFSA . . . . .                      | 525 |
| 9.127.3.9 getGenerators . . . . .               | 525 |
| 9.127.3.10getIndex . . . . .                    | 525 |
| 9.127.3.11getNielsenGenerators . . . . .        | 525 |
| 9.127.3.12getRank . . . . .                     | 526 |
| 9.127.3.13graphviz_format . . . . .             | 526 |
| 9.127.3.14normalizer . . . . .                  | 526 |
| 9.127.3.15operator* . . . . .                   | 526 |
| 9.127.3.16operator+ . . . . .                   | 526 |
| 9.127.3.17operator+ . . . . .                   | 526 |
| 9.127.3.18operator+= . . . . .                  | 526 |
| 9.127.3.19operator+= . . . . .                  | 526 |
| 9.127.3.20operator== . . . . .                  | 527 |
| 9.127.3.21operator^ . . . . .                   | 527 |
| 9.127.3.22operator^= . . . . .                  | 527 |
| 9.127.3.23trim . . . . .                        | 527 |
| 9.127.4 Member Data Documentation . . . . .     | 527 |
| 9.127.4.1 foldDetails . . . . .                 | 527 |
| 9.127.4.2 fsaDone . . . . .                     | 527 |
| 9.127.4.3 nielsDone . . . . .                   | 527 |
| 9.127.4.4 theFSA . . . . .                      | 527 |
| 9.127.4.5 theGenerators . . . . .               | 528 |
| 9.127.4.6 theNielsenGenerators . . . . .        | 528 |

|   |     |
|---|-----|
| 9.127.4.7 theNumberOfGenerators . . . . .                   | 528 |
| 9.128SubwordEditingDistanceCyclic Class Reference . . . . . | 529 |
| 9.128.1 Detailed Description . . . . .                      | 529 |
| 9.128.2 Constructor & Destructor Documentation . . . . .    | 529 |
| 9.128.2.1 SubwordEditingDistanceCyclic . . . . .            | 529 |
| 9.128.3 Member Function Documentation . . . . .             | 530 |
| 9.128.3.1 measure . . . . .                                 | 530 |
| 9.129SubwordHammingDistanceCyclic Class Reference . . . . . | 531 |
| 9.129.1 Detailed Description . . . . .                      | 531 |
| 9.129.2 Constructor & Destructor Documentation . . . . .    | 531 |
| 9.129.2.1 SubwordHammingDistanceCyclic . . . . .            | 531 |
| 9.129.3 Member Function Documentation . . . . .             | 532 |
| 9.129.3.1 measure . . . . .                                 | 532 |
| 9.130SubwordScrambler Class Reference . . . . .             | 533 |
| 9.130.1 Detailed Description . . . . .                      | 533 |
| 9.130.2 Constructor & Destructor Documentation . . . . .    | 533 |
| 9.130.2.1 SubwordScrambler . . . . .                        | 533 |
| 9.130.3 Member Function Documentation . . . . .             | 533 |
| 9.130.3.1 fracChanged . . . . .                             | 533 |
| 9.130.3.2 scramble . . . . .                                | 534 |
| 9.130.4 Member Data Documentation . . . . .                 | 534 |
| 9.130.4.1 numGens . . . . .                                 | 534 |
| 9.130.4.2 theChangeFrac . . . . .                           | 534 |
| 9.131TheGrigorchukGroupAlgorithms Class Reference . . . . . | 535 |
| 9.131.1 Detailed Description . . . . .                      | 536 |
| 9.131.2 Constructor & Destructor Documentation . . . . .    | 537 |
| 9.131.2.1 TheGrigorchukGroupAlgorithms . . . . .            | 537 |
| 9.131.3 Member Function Documentation . . . . .             | 537 |
| 9.131.3.1 abelianImage . . . . .                            | 537 |
| 9.131.3.2 conjugate . . . . .                               | 537 |
| 9.131.3.3 conjugate_Kcosets . . . . .                       | 537 |

|  |     |
|--|-----|
| 9.131.3.4 cosetRepresentativeKSbgp . . . . .             | 537 |
| 9.131.3.5 cosetRepresentativeKSbgp . . . . .             | 538 |
| 9.131.3.6 decompositionBSbgp . . . . .                   | 538 |
| 9.131.3.7 findConjugator_Kcosets . . . . .               | 538 |
| 9.131.3.8 findOrder . . . . .                            | 538 |
| 9.131.3.9 KCosetLiftTable . . . . .                      | 538 |
| 9.131.3.10 liftPairKcosetsUP . . . . .                   | 538 |
| 9.131.3.11 liftPairsKcosetsUP . . . . .                  | 539 |
| 9.131.3.12 liftToSTone . . . . .                         | 539 |
| 9.131.3.13 push_back . . . . .                           | 539 |
| 9.131.3.14 push_front . . . . .                          | 539 |
| 9.131.3.15 randomWord . . . . .                          | 539 |
| 9.131.3.16 reduce . . . . .                              | 539 |
| 9.131.3.17 split . . . . .                               | 539 |
| 9.131.3.18 trivial . . . . .                             | 540 |
| 9.131.3.19 trivial_reduced . . . . .                     | 540 |
| 9.132 ThLeftNormalForm Class Reference . . . . .         | 541 |
| 9.132.1 Detailed Description . . . . .                   | 545 |
| 9.132.2 Member Typedef Documentation . . . . .           | 545 |
| 9.132.2.1 NF . . . . .                                   | 545 |
| 9.132.3 Member Enumeration Documentation . . . . .       | 546 |
| 9.132.3.1 transformationResult . . . . .                 | 546 |
| 9.132.4 Constructor & Destructor Documentation . . . . . | 546 |
| 9.132.4.1 ThLeftNormalForm . . . . .                     | 546 |
| 9.132.4.2 ThLeftNormalForm . . . . .                     | 546 |
| 9.132.4.3 ThLeftNormalForm . . . . .                     | 546 |
| 9.132.4.4 ThLeftNormalForm . . . . .                     | 546 |
| 9.132.4.5 ThLeftNormalForm . . . . .                     | 547 |
| 9.132.4.6 ThLeftNormalForm . . . . .                     | 547 |
| 9.132.5 Member Function Documentation . . . . .          | 547 |
| 9.132.5.1 adjust . . . . .                               | 547 |

|            |                                      |     |
|------------|--------------------------------------|-----|
| 9.132.5.2  | adjustDecomposition . . . . .        | 547 |
| 9.132.5.3  | areConjugate . . . . .               | 547 |
| 9.132.5.4  | areConjugate_uss . . . . .           | 548 |
| 9.132.5.5  | computePeriod . . . . .              | 548 |
| 9.132.5.6  | cycle . . . . .                      | 548 |
| 9.132.5.7  | decycle . . . . .                    | 548 |
| 9.132.5.8  | findSSSRepresentative . . . . .      | 548 |
| 9.132.5.9  | findUSSRepresentative . . . . .      | 548 |
| 9.132.5.10 | getDecomposition . . . . .           | 549 |
| 9.132.5.11 | getMainPullback . . . . .            | 549 |
| 9.132.5.12 | getPower . . . . .                   | 549 |
| 9.132.5.13 | getPullback . . . . .                | 549 |
| 9.132.5.14 | getRank . . . . .                    | 549 |
| 9.132.5.15 | getSimpleConjugator . . . . .        | 550 |
| 9.132.5.16 | getSimpleConjugators . . . . .       | 550 |
| 9.132.5.17 | getSimpleSummitConjugator . . . . .  | 550 |
| 9.132.5.18 | getSimpleSummitConjugators . . . . . | 550 |
| 9.132.5.19 | getSimpleUltraConjugator . . . . .   | 550 |
| 9.132.5.20 | getSimpleUltraConjugators . . . . .  | 551 |
| 9.132.5.21 | getTransport . . . . .               | 551 |
| 9.132.5.22 | getTransports . . . . .              | 551 |
| 9.132.5.23 | getWord . . . . .                    | 551 |
| 9.132.5.24 | increaseRank . . . . .               | 551 |
| 9.132.5.25 | inverse . . . . .                    | 552 |
| 9.132.5.26 | isTrivial . . . . .                  | 552 |
| 9.132.5.27 | multiply . . . . .                   | 552 |
| 9.132.5.28 | operator NF . . . . .                | 552 |
| 9.132.5.29 | operator ThRightNormalForm . . . . . | 552 |
| 9.132.5.30 | operator != . . . . .                | 552 |
| 9.132.5.31 | operator * . . . . .                 | 552 |
| 9.132.5.32 | operator *= . . . . .                | 553 |

|  |     |
|--|-----|
| 9.132.5.33operator-                                | 553 |
| 9.132.5.34operator<                                | 553 |
| 9.132.5.35operator=                                | 553 |
| 9.132.5.36operator=                                | 553 |
| 9.132.5.37operator==                               | 554 |
| 9.132.5.38reverse                                  | 554 |
| 9.132.5.39setDecomposition                         | 554 |
| 9.132.5.40setPower                                 | 554 |
| 9.132.5.41transform                                | 554 |
| 9.132.5.42ussAddTrajectory                         | 554 |
| 9.132.5.43ussConstructionIteration                 | 555 |
| 9.132.6 Member Data Documentation                  | 555 |
| 9.132.6.1 theDecomposition                         | 555 |
| 9.132.6.2 theOmegaPower                            | 555 |
| 9.132.6.3 theRank                                  | 555 |
| 9.133ThompsonGroupFNormalForm Class Reference      | 556 |
| 9.133.1 Detailed Description                       | 557 |
| 9.133.2 Constructor & Destructor Documentation     | 557 |
| 9.133.2.1 ThompsonGroupFNormalForm                 | 557 |
| 9.133.2.2 ThompsonGroupFNormalForm                 | 557 |
| 9.133.3 Member Function Documentation              | 557 |
| 9.133.3.1 mergeUnits                               | 557 |
| 9.133.3.2 nextOperation                            | 557 |
| 9.133.3.3 operator*                                | 558 |
| 9.133.3.4 operator*=                               | 558 |
| 9.133.3.5 operator-                                | 558 |
| 9.133.3.6 operator==                               | 558 |
| 9.133.3.7 removeBadPairs                           | 558 |
| 9.133.3.8 semiNormalFormFor                        | 559 |
| 9.133.4 Friends And Related Function Documentation | 559 |
| 9.133.4.1 operator<<                               | 559 |



|  |     |
|--|-----|
| 9.134ThRightNormalForm Class Reference . . . . .         | 560 |
| 9.134.1 Detailed Description . . . . .                   | 564 |
| 9.134.2 Member Typedef Documentation . . . . .           | 565 |
| 9.134.2.1 NF . . . . .                                   | 565 |
| 9.134.3 Member Enumeration Documentation . . . . .       | 565 |
| 9.134.3.1 transformationResult . . . . .                 | 565 |
| 9.134.4 Constructor & Destructor Documentation . . . . . | 565 |
| 9.134.4.1 ThRightNormalForm . . . . .                    | 565 |
| 9.134.4.2 ThRightNormalForm . . . . .                    | 565 |
| 9.134.4.3 ThRightNormalForm . . . . .                    | 566 |
| 9.134.4.4 ThRightNormalForm . . . . .                    | 566 |
| 9.134.4.5 ThRightNormalForm . . . . .                    | 566 |
| 9.134.5 Member Function Documentation . . . . .          | 566 |
| 9.134.5.1 adjust . . . . .                               | 566 |
| 9.134.5.2 adjustDecomposition . . . . .                  | 566 |
| 9.134.5.3 areConjugate . . . . .                         | 567 |
| 9.134.5.4 computeCentralizer . . . . .                   | 567 |
| 9.134.5.5 cycle . . . . .                                | 567 |
| 9.134.5.6 decycle . . . . .                              | 567 |
| 9.134.5.7 findSSSRepresentative . . . . .                | 568 |
| 9.134.5.8 findUSSRepresentative . . . . .                | 568 |
| 9.134.5.9 getDecomposition . . . . .                     | 568 |
| 9.134.5.10getPower . . . . .                             | 568 |
| 9.134.5.11getPullback . . . . .                          | 568 |
| 9.134.5.12getPullback . . . . .                          | 568 |
| 9.134.5.13getRank . . . . .                              | 569 |
| 9.134.5.14getShortWord . . . . .                         | 569 |
| 9.134.5.15getSimpleConjugator . . . . .                  | 569 |
| 9.134.5.16getSimpleConjugators . . . . .                 | 569 |
| 9.134.5.17getSimpleSummitConjugator . . . . .            | 569 |
| 9.134.5.18getSimpleSummitConjugators . . . . .           | 570 |

|  |     |
|--|-----|
| 9.134.5.19getSimpleUltraConjugator . . . . .             | 570 |
| 9.134.5.20getSimpleUltraConjugators . . . . .            | 570 |
| 9.134.5.21getTransport . . . . .                         | 570 |
| 9.134.5.22getTransports . . . . .                        | 571 |
| 9.134.5.23getWord . . . . .                              | 571 |
| 9.134.5.24increaseRank . . . . .                         | 571 |
| 9.134.5.25inverse . . . . .                              | 571 |
| 9.134.5.26isTrivial . . . . .                            | 571 |
| 9.134.5.27multiply . . . . .                             | 571 |
| 9.134.5.28operator NF . . . . .                          | 572 |
| 9.134.5.29operator ThLeftNormalForm . . . . .            | 572 |
| 9.134.5.30operator!= . . . . .                           | 572 |
| 9.134.5.31operator* . . . . .                            | 572 |
| 9.134.5.32operator*= . . . . .                           | 572 |
| 9.134.5.33operator- . . . . .                            | 572 |
| 9.134.5.34operator< . . . . .                            | 573 |
| 9.134.5.35operator= . . . . .                            | 573 |
| 9.134.5.36operator== . . . . .                           | 573 |
| 9.134.5.37randomPositive . . . . .                       | 573 |
| 9.134.5.38setDecomposition . . . . .                     | 573 |
| 9.134.5.39setPower . . . . .                             | 573 |
| 9.134.5.40sssConstructionIteration . . . . .             | 574 |
| 9.134.5.41transform . . . . .                            | 574 |
| 9.134.6 Member Data Documentation . . . . .              | 574 |
| 9.134.6.1 theDecomposition . . . . .                     | 574 |
| 9.134.6.2 theOmegaPower . . . . .                        | 574 |
| 9.134.6.3 theRank . . . . .                              | 574 |
| 9.135Permutation::triple Struct Reference . . . . .      | 576 |
| 9.135.1 Detailed Description . . . . .                   | 576 |
| 9.135.2 Constructor & Destructor Documentation . . . . . | 576 |
| 9.135.2.1 triple . . . . .                               | 576 |

|  |     |
|--|-----|
| 9.135.3 Member Data Documentation . . . . .                        | 576 |
| 9.135.3.1 c1 . . . . .   | 576 |
| 9.135.3.2 c2 . . . . .   | 576 |
| 9.135.3.3 c3 . . . . .   | 576 |
| 9.136triple< T1, T2, T3 > Class Template Reference . . . . .       | 577 |
| 9.136.1 Detailed Description . . . . .                             | 577 |
| 9.136.2 Constructor & Destructor Documentation . . . . .           | 577 |
| 9.136.2.1 triple . . . . .   | 577 |
| 9.136.3 Member Function Documentation . . . . .                    | 578 |
| 9.136.3.1 operator!= . . . . .                                     | 578 |
| 9.136.3.2 operator< . . . . .                                      | 578 |
| 9.136.3.3 operator== . . . . .                                     | 578 |
| 9.136.3.4 operator> . . . . .                                      | 578 |
| 9.136.4 Friends And Related Function Documentation . . . . .       | 578 |
| 9.136.4.1 operator<< . . . . .                                     | 578 |
| 9.136.5 Member Data Documentation . . . . .                        | 578 |
| 9.136.5.1 first . . . . .  | 578 |
| 9.136.5.2 second . . . . .   | 579 |
| 9.136.5.3 third . . . . .  | 579 |
| 9.137TripleDecompositionProtocolInstance Class Reference . . . . . | 580 |
| 9.137.1 Detailed Description . . . . .                             | 581 |
| 9.137.2 Constructor & Destructor Documentation . . . . .           | 582 |
| 9.137.2.1 TripleDecompositionProtocolInstance . . . . .            | 582 |
| 9.137.3 Member Function Documentation . . . . .                    | 582 |
| 9.137.3.1 getBraidRank . . . . .                                   | 582 |
| 9.137.3.2 getPrivateKeyA . . . . .                                 | 582 |
| 9.137.3.3 getPrivateKeyB . . . . .                                 | 582 |
| 9.137.3.4 getPublicKeyA . . . . .                                  | 583 |
| 9.137.3.5 getPublicKeyB . . . . .                                  | 583 |
| 9.137.3.6 getSharedKey . . . . .                                   | 583 |
| 9.137.3.7 random . . . . .   | 583 |

|  |     |
|--|-----|
| 9.137.3.8 randomWord . . . . .                               | 583 |
| 9.137.4 Member Data Documentation . . . . .                  | 584 |
| 9.137.4.1 theConjugators . . . . .                           | 584 |
| 9.137.4.2 thePrivateKeyA . . . . .                           | 584 |
| 9.137.4.3 thePrivateKeyB . . . . .                           | 584 |
| 9.137.4.4 thePublicKeyA . . . . .                            | 584 |
| 9.137.4.5 thePublicKeyB . . . . .                            | 585 |
| 9.137.4.6 theRank . . . . .                                  | 585 |
| 9.137.4.7 theSharedKey . . . . .                             | 585 |
| 9.138TTP_Conf Struct Reference . . . . .                     | 586 |
| 9.138.1 Detailed Description . . . . .                       | 586 |
| 9.138.2 Friends And Related Function Documentation . . . . . | 586 |
| 9.138.2.1 operator<< . . . . .                               | 586 |
| 9.138.3 Member Data Documentation . . . . .                  | 586 |
| 9.138.3.1 len_w . . . . .                                    | 586 |
| 9.138.3.2 len_z . . . . .                                    | 586 |
| 9.138.3.3 N . . . . .  | 587 |
| 9.138.3.4 nBL . . . . .                                      | 587 |
| 9.138.3.5 nBR . . . . .                                      | 587 |
| 9.138.3.6 nGamma . . . . .                                   | 587 |
| 9.139TTPAttack Class Reference . . . . .                     | 588 |
| 9.139.1 Detailed Description . . . . .                       | 589 |
| 9.139.2 Member Enumeration Documentation . . . . .           | 589 |
| 9.139.2.1 TTP_Result . . . . .                               | 589 |
| 9.139.3 Constructor & Destructor Documentation . . . . .     | 589 |
| 9.139.3.1 TTPAttack . . . . .                                | 589 |
| 9.139.4 Member Function Documentation . . . . .              | 590 |
| 9.139.4.1 cycleDecycle . . . . .                             | 590 |
| 9.139.4.2 LBA . . . . .                                      | 590 |
| 9.139.4.3 oneOfSSSReps . . . . .                             | 590 |
| 9.139.4.4 printStats . . . . .                               | 590 |

|  |     |
|--|-----|
| 9.139.4.5 reduceDeltaLBA . . . . .                           | 590 |
| 9.139.4.6 run . . . . .                                      | 590 |
| 9.139.4.7 simpleLBA . . . . .                                | 590 |
| 9.139.5 Member Data Documentation . . . . .                  | 590 |
| 9.139.5.1 BS . . . . .                                       | 590 |
| 9.139.5.2 N . . . . .  | 590 |
| 9.140 TTPLBA Class Reference . . . . .                       | 592 |
| 9.140.1 Detailed Description . . . . .                       | 592 |
| 9.140.2 Constructor & Destructor Documentation . . . . .     | 592 |
| 9.140.2.1 TTPLBA . . . . .                                   | 592 |
| 9.140.3 Member Function Documentation . . . . .              | 593 |
| 9.140.3.1 addNewElt . . . . .                                | 593 |
| 9.140.3.2 reduce . . . . .                                   | 593 |
| 9.140.3.3 simpleLBA . . . . .                                | 593 |
| 9.140.3.4 tryNode . . . . .                                  | 593 |
| 9.140.4 Member Data Documentation . . . . .                  | 593 |
| 9.140.4.1 savTuple . . . . .                                 | 593 |
| 9.141 TTPTuple Class Reference . . . . .                     | 594 |
| 9.141.1 Detailed Description . . . . .                       | 595 |
| 9.141.2 Constructor & Destructor Documentation . . . . .     | 595 |
| 9.141.2.1 TTPTuple . . . . .                                 | 595 |
| 9.141.2.2 TTPTuple . . . . .                                 | 595 |
| 9.141.2.3 TTPTuple . . . . .                                 | 595 |
| 9.141.3 Member Function Documentation . . . . .              | 595 |
| 9.141.3.1 length . . . . .                                   | 595 |
| 9.141.3.2 shortAndTestTuples . . . . .                       | 595 |
| 9.141.3.3 shorten . . . . .                                  | 595 |
| 9.141.3.4 testTuples . . . . .                               | 596 |
| 9.141.4 Friends And Related Function Documentation . . . . . | 596 |
| 9.141.4.1 operator< . . . . .                                | 596 |
| 9.141.5 Member Data Documentation . . . . .                  | 596 |

|  |     |
|--|-----|
| 9.141.5.1 origWL . . . . .                               | 596 |
| 9.141.5.2 origWR . . . . .                               | 596 |
| 9.141.5.3 WL . . . . .                                   | 596 |
| 9.141.5.4 WR . . . . .                                   | 596 |
| 9.141.5.5 z . . . . .                                    | 597 |
| 9.142udPDFPageObjectCircle Class Reference . . . . .     | 598 |
| 9.142.1 Detailed Description . . . . .                   | 598 |
| 9.142.2 Constructor & Destructor Documentation . . . . . | 598 |
| 9.142.2.1 udPDFPageObjectCircle . . . . .                | 598 |
| 9.142.3 Member Function Documentation . . . . .          | 599 |
| 9.142.3.1 write . . . . .                                | 599 |
| 9.142.4 Member Data Documentation . . . . .              | 599 |
| 9.142.4.1 border_color . . . . .                         | 599 |
| 9.142.4.2 theFill . . . . .                              | 599 |
| 9.142.4.3 theRad . . . . .                               | 599 |
| 9.142.4.4 theX . . . . .                                 | 599 |
| 9.142.4.5 theY . . . . .                                 | 599 |
| 9.143udPDFPageObjectHorizLine Class Reference . . . . .  | 600 |
| 9.143.1 Detailed Description . . . . .                   | 600 |
| 9.143.2 Constructor & Destructor Documentation . . . . . | 600 |
| 9.143.2.1 udPDFPageObjectHorizLine . . . . .             | 600 |
| 9.144udPDFPageObjectLine Class Reference . . . . .       | 602 |
| 9.144.1 Detailed Description . . . . .                   | 602 |
| 9.144.2 Constructor & Destructor Documentation . . . . . | 603 |
| 9.144.2.1 udPDFPageObjectLine . . . . .                  | 603 |
| 9.144.3 Member Function Documentation . . . . .          | 603 |
| 9.144.3.1 write . . . . .                                | 603 |
| 9.144.4 Member Data Documentation . . . . .              | 603 |
| 9.144.4.1 dashed . . . . .                               | 603 |
| 9.144.4.2 gray_color . . . . .                           | 603 |
| 9.144.4.3 theX1 . . . . .                                | 603 |

|  |     |
|--|-----|
| 9.144.4.4 theX2 . . . . .                                | 603 |
| 9.144.4.5 theY1 . . . . .                                | 604 |
| 9.144.4.6 theY2 . . . . .                                | 604 |
| 9.145udPDFPageObjectRect Class Reference . . . . .       | 605 |
| 9.145.1 Detailed Description . . . . .                   | 605 |
| 9.145.2 Constructor & Destructor Documentation . . . . . | 606 |
| 9.145.2.1 udPDFPageObjectRect . . . . .                  | 606 |
| 9.145.3 Member Function Documentation . . . . .          | 606 |
| 9.145.3.1 write . . . . .                                | 606 |
| 9.145.4 Member Data Documentation . . . . .              | 606 |
| 9.145.4.1 border_color . . . . .                         | 606 |
| 9.145.4.2 theFill . . . . .                              | 606 |
| 9.145.4.3 theH . . . . .                                 | 606 |
| 9.145.4.4 theW . . . . .                                 | 606 |
| 9.145.4.5 theX . . . . .                                 | 607 |
| 9.145.4.6 theY . . . . .                                 | 607 |
| 9.146udPDFPageObjectSquare Class Reference . . . . .     | 608 |
| 9.146.1 Detailed Description . . . . .                   | 608 |
| 9.146.2 Constructor & Destructor Documentation . . . . . | 608 |
| 9.146.2.1 udPDFPageObjectSquare . . . . .                | 608 |
| 9.147udPDFPageObjectText Class Reference . . . . .       | 610 |
| 9.147.1 Detailed Description . . . . .                   | 610 |
| 9.147.2 Constructor & Destructor Documentation . . . . . | 610 |
| 9.147.2.1 udPDFPageObjectText . . . . .                  | 610 |
| 9.147.3 Member Function Documentation . . . . .          | 611 |
| 9.147.3.1 write . . . . .                                | 611 |
| 9.147.4 Member Data Documentation . . . . .              | 611 |
| 9.147.4.1 text . . . . .                                 | 611 |
| 9.147.4.2 theX . . . . .                                 | 611 |
| 9.147.4.3 theY . . . . .                                 | 611 |
| 9.147.4.4 tSize . . . . .                                | 611 |

|           |   |     |
|-----------|---|-----|
| 9.148     | udPDFPageObjectVertLine Class Reference . . . . . | 612 |
| 9.148.1   | Detailed Description . . . . .                    | 612 |
| 9.148.2   | Constructor & Destructor Documentation . . . . .  | 612 |
| 9.148.2.1 | udPDFPageObjectVertLine . . . . .                 | 612 |
| 9.149     | UniformPartition Class Reference . . . . .        | 614 |
| 9.149.1   | Detailed Description . . . . .                    | 614 |
| 9.149.2   | Constructor & Destructor Documentation . . . . .  | 614 |
| 9.149.2.1 | UniformPartition . . . . .                        | 614 |
| 9.149.3   | Member Function Documentation . . . . .           | 614 |
| 9.149.3.1 | getPartition . . . . .                            | 614 |
| 9.150     | UniformScrambler Class Reference . . . . .        | 615 |
| 9.150.1   | Detailed Description . . . . .                    | 615 |
| 9.150.2   | Constructor & Destructor Documentation . . . . .  | 615 |
| 9.150.2.1 | UniformScrambler . . . . .                        | 615 |
| 9.150.3   | Member Function Documentation . . . . .           | 615 |
| 9.150.3.1 | fracChanged . . . . .                             | 615 |
| 9.150.3.2 | scramble . . . . .                                | 616 |
| 9.150.4   | Member Data Documentation . . . . .               | 616 |
| 9.150.4.1 | numGens . . . . .                                 | 616 |
| 9.150.4.2 | theChangeProb . . . . .                           | 616 |
| 9.150.4.3 | theFracChanged . . . . .                          | 616 |
| 9.151     | Value Class Reference . . . . .                   | 617 |
| 9.151.1   | Detailed Description . . . . .                    | 617 |
| 9.151.2   | Constructor & Destructor Documentation . . . . .  | 618 |
| 9.151.2.1 | Value . . . . .                                   | 618 |
| 9.151.2.2 | Value . . . . .                                   | 618 |
| 9.151.2.3 | Value . . . . .                                   | 618 |
| 9.151.3   | Member Function Documentation . . . . .           | 618 |
| 9.151.3.1 | operator double . . . . .                         | 618 |
| 9.151.3.2 | operator int . . . . .                            | 619 |
| 9.151.3.3 | operator string . . . . .                         | 619 |



|  |     |
|--|-----|
| 9.151.4 Friends And Related Function Documentation . . . . . | 619 |
| 9.151.4.1 operator>> . . . . .                               | 619 |
| 9.151.5 Member Data Documentation . . . . .                  | 619 |
| 9.151.5.1 theValue . . . . .                                 | 619 |
| 9.152 VectorEnumerator Class Reference . . . . .             | 620 |
| 9.152.1 Detailed Description . . . . .                       | 621 |
| 9.152.2 Constructor & Destructor Documentation . . . . .     | 621 |
| 9.152.2.1 VectorEnumerator . . . . .                         | 621 |
| 9.152.3 Member Function Documentation . . . . .              | 622 |
| 9.152.3.1 end . . . . .                                      | 622 |
| 9.152.3.2 finish . . . . .                                   | 622 |
| 9.152.3.3 getLength . . . . .                                | 622 |
| 9.152.3.4 getSeq . . . . .                                   | 622 |
| 9.152.3.5 next . . . . .                                     | 622 |
| 9.152.3.6 operator* . . . . .                                | 623 |
| 9.152.3.7 operator++ . . . . .                               | 623 |
| 9.152.3.8 printSeq . . . . .                                 | 623 |
| 9.152.3.9 seqComplete . . . . .                              | 623 |
| 9.152.3.10 seqLimit . . . . .                                | 623 |
| 9.152.3.11 seqOK . . . . .                                   | 623 |
| 9.152.3.12 start . . . . .                                   | 623 |
| 9.152.3.13 stepBack . . . . .                                | 624 |
| 9.152.3.14 stepTo . . . . .                                  | 624 |
| 9.152.4 Member Data Documentation . . . . .                  | 624 |
| 9.152.4.1 curLength . . . . .                                | 624 |
| 9.152.4.2 curVector . . . . .                                | 624 |
| 9.153 WhiteheadAutoSetType2 Class Reference . . . . .        | 625 |
| 9.153.1 Detailed Description . . . . .                       | 626 |
| 9.153.2 Constructor & Destructor Documentation . . . . .     | 626 |
| 9.153.2.1 WhiteheadAutoSetType2 . . . . .                    | 626 |
| 9.153.2.2 ~WhiteheadAutoSetType2 . . . . .                   | 626 |

|  |     |
|--|-----|
| 9.153.3 Member Function Documentation . . . . .          | 626 |
| 9.153.3.1 computeSet . . . . .                           | 626 |
| 9.153.3.2 getMap . . . . .                               | 626 |
| 9.153.3.3 getRandomAuto . . . . .                        | 626 |
| 9.153.3.4 getSet . . . . .                               | 626 |
| 9.153.4 Member Data Documentation . . . . .              | 627 |
| 9.153.4.1 nElemAutos . . . . .                           | 627 |
| 9.153.4.2 nGens . . . . .                                | 627 |
| 9.153.4.3 theSet . . . . .                               | 627 |
| 9.154 WhiteheadGraph Class Reference . . . . .           | 628 |
| 9.154.1 Detailed Description . . . . .                   | 628 |
| 9.154.2 Constructor & Destructor Documentation . . . . . | 629 |
| 9.154.2.1 WhiteheadGraph . . . . .                       | 629 |
| 9.154.2.2 WhiteheadGraph . . . . .                       | 629 |
| 9.154.3 Member Function Documentation . . . . .          | 629 |
| 9.154.3.1 assign . . . . .                               | 629 |
| 9.154.3.2 getGraph . . . . .                             | 629 |
| 9.154.3.3 getWord . . . . .                              | 629 |
| 9.154.4 Member Data Documentation . . . . .              | 630 |
| 9.154.4.1 nOfGenerators . . . . .                        | 630 |
| 9.154.4.2 theGraph . . . . .                             | 630 |
| 9.154.4.3 theWord . . . . .                              | 630 |
| 9.155 WhiteheadMinimization Class Reference . . . . .    | 631 |
| 9.155.1 Detailed Description . . . . .                   | 631 |
| 9.155.2 Constructor & Destructor Documentation . . . . . | 631 |
| 9.155.2.1 WhiteheadMinimization . . . . .                | 631 |
| 9.155.3 Member Function Documentation . . . . .          | 632 |
| 9.155.3.1 findMinimal . . . . .                          | 632 |
| 9.155.3.2 getSet . . . . .                               | 632 |
| 9.155.3.3 isMinimal . . . . .                            | 632 |
| 9.155.4 Member Data Documentation . . . . .              | 633 |

|  |     |
|--|-----|
| 9.155.4.1 wSet . . . . .                                     | 633 |
| 9.156 WhiteheadMultiGraph Class Reference . . . . .          | 634 |
| 9.156.1 Detailed Description . . . . .                       | 634 |
| 9.156.2 Constructor & Destructor Documentation . . . . .     | 634 |
| 9.156.2.1 WhiteheadMultiGraph . . . . .                      | 634 |
| 9.157 WhiteheadSimpleGraph Class Reference . . . . .         | 635 |
| 9.157.1 Detailed Description . . . . .                       | 637 |
| 9.157.2 Constructor & Destructor Documentation . . . . .     | 637 |
| 9.157.2.1 WhiteheadSimpleGraph . . . . .                     | 637 |
| 9.157.2.2 ~WhiteheadSimpleGraph . . . . .                    | 637 |
| 9.157.3 Member Function Documentation . . . . .              | 637 |
| 9.157.3.1 cutVertices . . . . .                              | 637 |
| 9.157.3.2 cutVerticesBruteForce . . . . .                    | 637 |
| 9.157.3.3 genToIndex . . . . .                               | 637 |
| 9.157.3.4 getCount . . . . .                                 | 637 |
| 9.157.3.5 getSize . . . . .                                  | 638 |
| 9.157.3.6 getWeightNames . . . . .                           | 638 |
| 9.157.3.7 getWeightVector . . . . .                          | 638 |
| 9.157.3.8 indToGenerator . . . . .                           | 639 |
| 9.157.3.9 isUndirected . . . . .                             | 639 |
| 9.157.3.10 makeUndirected . . . . .                          | 639 |
| 9.157.3.11 inOfComponents . . . . .                          | 639 |
| 9.157.3.12 numberOfComponents . . . . .                      | 639 |
| 9.157.3.13 numberOfCutVertices . . . . .                     | 639 |
| 9.157.3.14 numberOfCutVerticesBruteForce . . . . .           | 640 |
| 9.157.3.15 printOn . . . . .                                 | 640 |
| 9.157.4 Friends And Related Function Documentation . . . . . | 640 |
| 9.157.4.1 operator<< . . . . .                               | 640 |
| 9.157.5 Member Data Documentation . . . . .                  | 640 |
| 9.157.5.1 theAdjMatrix . . . . .                             | 640 |
| 9.157.5.2 theSize . . . . .                                  | 640 |

|  |     |
|--|-----|
| 9.157.5.3 undirected . . . . .                           | 640 |
| 9.158 Word Class Reference . . . . .                     | 642 |
| 9.158.1 Detailed Description . . . . .                   | 646 |
| 9.158.2 Member Typedef Documentation . . . . .           | 647 |
| 9.158.2.1 const_iterator . . . . .                       | 647 |
| 9.158.2.2 iterator . . . . .                             | 647 |
| 9.158.2.3 PII . . . . .                                  | 647 |
| 9.158.3 Constructor & Destructor Documentation . . . . . | 647 |
| 9.158.3.1 Word . . . . .                                 | 647 |
| 9.158.3.2 Word . . . . .                                 | 647 |
| 9.158.3.3 Word . . . . .                                 | 647 |
| 9.158.3.4 Word . . . . .                                 | 648 |
| 9.158.3.5 Word . . . . .                                 | 648 |
| 9.158.4 Member Function Documentation . . . . .          | 648 |
| 9.158.4.1 _cyclicallyPermute . . . . .                   | 648 |
| 9.158.4.2 begin . . . . .                                | 648 |
| 9.158.4.3 begin . . . . .                                | 648 |
| 9.158.4.4 cyclicallyPermute . . . . .                    | 648 |
| 9.158.4.5 cyclicallyReduce . . . . .                     | 649 |
| 9.158.4.6 cyclicallyReduce . . . . .                     | 649 |
| 9.158.4.7 cyclicallyReduceWord . . . . .                 | 649 |
| 9.158.4.8 cyclicallyReduceWord . . . . .                 | 649 |
| 9.158.4.9 cyclicLeftShift . . . . .                      | 649 |
| 9.158.4.10 cyclicRightShift . . . . .                    | 650 |
| 9.158.4.11 doesContain . . . . .                         | 650 |
| 9.158.4.12 end . . . . .                                 | 650 |
| 9.158.4.13 end . . . . .                                 | 650 |
| 9.158.4.14 exponentSum . . . . .                         | 650 |
| 9.158.4.15 freelyReduce . . . . .                        | 650 |
| 9.158.4.16 freelyReduce . . . . .                        | 650 |
| 9.158.4.17 getList . . . . .                             | 651 |

|            |                       |     |
|------------|-----------------------|-----|
| 9.158.4.18 | getList               | 651 |
| 9.158.4.19 | getPower              | 651 |
| 9.158.4.20 | initialSegment        | 651 |
| 9.158.4.21 | insert                | 651 |
| 9.158.4.22 | insert                | 652 |
| 9.158.4.23 | insert                | 652 |
| 9.158.4.24 | insert                | 652 |
| 9.158.4.25 | inverse               | 652 |
| 9.158.4.26 | isIn                  | 652 |
| 9.158.4.27 | length                | 652 |
| 9.158.4.28 | minimalEquivalentForm | 652 |
| 9.158.4.29 | operator!=            | 653 |
| 9.158.4.30 | operator*             | 653 |
| 9.158.4.31 | operator*=            | 653 |
| 9.158.4.32 | operator-             | 653 |
| 9.158.4.33 | operator<             | 653 |
| 9.158.4.34 | operator==            | 654 |
| 9.158.4.35 | operator>             | 654 |
| 9.158.4.36 | operator^             | 654 |
| 9.158.4.37 | operator^             | 654 |
| 9.158.4.38 | operator^=            | 654 |
| 9.158.4.39 | operator^=            | 654 |
| 9.158.4.40 | pop_back              | 654 |
| 9.158.4.41 | pop_front             | 655 |
| 9.158.4.42 | power                 | 655 |
| 9.158.4.43 | printOn               | 655 |
| 9.158.4.44 | push_back             | 655 |
| 9.158.4.45 | push_back             | 655 |
| 9.158.4.46 | push_front            | 655 |
| 9.158.4.47 | push_front            | 655 |
| 9.158.4.48 | randomWord            | 656 |

|  |     |
|--|-----|
| 9.158.4.49 randomWord . . . . .                              | 656 |
| 9.158.4.50 replace . . . . .                                 | 656 |
| 9.158.4.51 replace . . . . .                                 | 656 |
| 9.158.4.52 replace . . . . .                                 | 656 |
| 9.158.4.53 replaceGenerators . . . . .                       | 656 |
| 9.158.4.54 segment . . . . .                                 | 657 |
| 9.158.4.55 terminalSegment . . . . .                         | 657 |
| 9.158.5 Friends And Related Function Documentation . . . . . | 657 |
| 9.158.5.1 operator<< . . . . .                               | 657 |
| 9.158.5.2 operator>> . . . . .                               | 657 |
| 9.159 WordDraw Class Reference . . . . .                     | 658 |
| 9.159.1 Detailed Description . . . . .                       | 658 |
| 9.159.2 Constructor & Destructor Documentation . . . . .     | 658 |
| 9.159.2.1 WordDraw . . . . .                                 | 658 |
| 9.159.2.2 WordDraw . . . . .                                 | 659 |
| 9.159.2.3 ~WordDraw . . . . .                                | 659 |
| 9.159.3 Member Function Documentation . . . . .              | 659 |
| 9.159.3.1 drawCompressedBraid . . . . .                      | 659 |
| 9.159.3.2 drawGenerator . . . . .                            | 659 |
| 9.159.3.3 drawHorizontalGrid . . . . .                       | 659 |
| 9.159.3.4 drawVerticalGrid . . . . .                         | 660 |
| 9.159.3.5 saveTo . . . . .                                   | 660 |
| 9.159.4 Member Data Documentation . . . . .                  | 660 |
| 9.159.4.1 betBraids . . . . .                                | 660 |
| 9.159.4.2 N . . . . .  | 660 |
| 9.159.4.3 ss . . . . .                                       | 660 |
| 9.159.4.4 theImage . . . . .                                 | 660 |
| 9.159.4.5 theLength . . . . .                                | 661 |
| 9.160 WordIterator Class Reference . . . . .                 | 662 |
| 9.160.1 Detailed Description . . . . .                       | 663 |
| 9.160.2 Member Typedef Documentation . . . . .               | 663 |

|  |     |
|--|-----|
| 9.160.2.1 PII . . . . .                                      | 663 |
| 9.160.2.2 PII . . . . .                                      | 663 |
| 9.160.3 Constructor & Destructor Documentation . . . . .     | 664 |
| 9.160.3.1 WordIterator . . . . .                             | 664 |
| 9.160.3.2 WordIterator . . . . .                             | 664 |
| 9.160.3.3 WordIterator . . . . .                             | 664 |
| 9.160.3.4 WordIterator . . . . .                             | 664 |
| 9.160.3.5 WordIterator . . . . .                             | 664 |
| 9.160.4 Member Function Documentation . . . . .              | 664 |
| 9.160.4.1 operator!= . . . . .                               | 664 |
| 9.160.4.2 operator!= . . . . .                               | 664 |
| 9.160.4.3 operator* . . . . .                                | 664 |
| 9.160.4.4 operator* . . . . .                                | 664 |
| 9.160.4.5 operator++ . . . . .                               | 664 |
| 9.160.4.6 operator++ . . . . .                               | 664 |
| 9.160.4.7 operator++ . . . . .                               | 664 |
| 9.160.4.8 operator++ . . . . .                               | 664 |
| 9.160.4.9 operator-- . . . . .                               | 664 |
| 9.160.4.10operator-- . . . . .                               | 664 |
| 9.160.4.11operator-- . . . . .                               | 664 |
| 9.160.4.12operator-- . . . . .                               | 664 |
| 9.160.4.13operator== . . . . .                               | 664 |
| 9.160.4.14operator== . . . . .                               | 664 |
| 9.160.5 Friends And Related Function Documentation . . . . . | 664 |
| 9.160.5.1 ConstWordIterator . . . . .                        | 664 |
| 9.160.5.2 Word . . . . .                                     | 665 |
| 9.160.5.3 WordRep . . . . .                                  | 665 |
| 9.160.6 Member Data Documentation . . . . .                  | 665 |
| 9.160.6.1 theIterator . . . . .                              | 665 |
| 9.160.6.2 theIterator . . . . .                              | 665 |
| 9.160.6.3 theList . . . . .                                  | 665 |

|  |     |
|--|-----|
| 9.160.6.4 theOffset . . . . .                            | 665 |
| 9.160.6.5 theWord . . . . .                              | 665 |
| 9.161 WordMultiplyScrambler Class Reference . . . . .    | 666 |
| 9.161.1 Detailed Description . . . . .                   | 666 |
| 9.161.2 Constructor & Destructor Documentation . . . . . | 666 |
| 9.161.2.1 WordMultiplyScrambler . . . . .                | 666 |
| 9.161.3 Member Function Documentation . . . . .          | 666 |
| 9.161.3.1 fracChanged . . . . .                          | 666 |
| 9.161.3.2 scramble . . . . .                             | 667 |
| 9.161.4 Member Data Documentation . . . . .              | 667 |
| 9.161.4.1 numGens . . . . .                              | 667 |
| 9.161.4.2 theChangeFrac . . . . .                        | 667 |
| 9.162 WordPairComparison Class Reference . . . . .       | 668 |
| 9.162.1 Detailed Description . . . . .                   | 668 |
| 9.162.2 Constructor & Destructor Documentation . . . . . | 669 |
| 9.162.2.1 WordPairComparison . . . . .                   | 669 |
| 9.162.2.2 WordPairComparison . . . . .                   | 669 |
| 9.162.3 Member Function Documentation . . . . .          | 669 |
| 9.162.3.1 comparePair . . . . .                          | 669 |
| 9.162.3.2 comparePair . . . . .                          | 669 |
| 9.162.3.3 distrEstimate . . . . .                        | 670 |
| 9.162.3.4 operator= . . . . .                            | 670 |
| 9.162.4 Member Data Documentation . . . . .              | 670 |
| 9.162.4.1 pSSM . . . . .                                 | 670 |
| 9.162.4.2 theRank . . . . .                              | 670 |
| 9.163 WordRep Class Reference . . . . .                  | 671 |
| 9.163.1 Detailed Description . . . . .                   | 673 |
| 9.163.2 Member Typedef Documentation . . . . .           | 673 |
| 9.163.2.1 PII . . . . .                                  | 673 |
| 9.163.3 Constructor & Destructor Documentation . . . . . | 673 |
| 9.163.3.1 WordRep . . . . .                              | 673 |



|   |     |
|---|-----|
| 9.163.3.2 WordRep . . . . .                     | 673 |
| 9.163.3.3 WordRep . . . . .                     | 673 |
| 9.163.3.4 WordRep . . . . .                     | 673 |
| 9.163.3.5 WordRep . . . . .                     | 673 |
| 9.163.3.6 WordRep . . . . .                     | 673 |
| 9.163.4 Member Function Documentation . . . . . | 673 |
| 9.163.4.1 clear . . . . .                       | 673 |
| 9.163.4.2 clone . . . . .                       | 674 |
| 9.163.4.3 cyclicallyPermute . . . . .           | 674 |
| 9.163.4.4 cyclicallyReduce . . . . .            | 674 |
| 9.163.4.5 cyclicallyReduce . . . . .            | 674 |
| 9.163.4.6 cyclicLeftShift . . . . .             | 674 |
| 9.163.4.7 cyclicRightShift . . . . .            | 674 |
| 9.163.4.8 doesContain . . . . .                 | 674 |
| 9.163.4.9 exponentSum . . . . .                 | 674 |
| 9.163.4.10freelyReduce . . . . .                | 674 |
| 9.163.4.11getList . . . . .                     | 674 |
| 9.163.4.12getList . . . . .                     | 674 |
| 9.163.4.13getPower . . . . .                    | 675 |
| 9.163.4.14initialSegment . . . . .              | 675 |
| 9.163.4.15insert . . . . .                      | 675 |
| 9.163.4.16insert . . . . .                      | 675 |
| 9.163.4.17insert . . . . .                      | 675 |
| 9.163.4.18insert . . . . .                      | 675 |
| 9.163.4.19inverse . . . . .                     | 675 |
| 9.163.4.20isIn . . . . .                        | 675 |
| 9.163.4.21length . . . . .                      | 675 |
| 9.163.4.22operator* . . . . .                   | 675 |
| 9.163.4.23operator*= . . . . .                  | 676 |
| 9.163.4.24operator< . . . . .                   | 676 |
| 9.163.4.25operator= . . . . .                   | 676 |

|            |  |            |
|------------|--|------------|
| 9.163.4.26 | operator== . . . . .                                 | 676        |
| 9.163.4.27 | operator> . . . . .                                  | 676        |
| 9.163.4.28 | operator^ . . . . .                                  | 676        |
| 9.163.4.29 | operator^ . . . . .                                  | 676        |
| 9.163.4.30 | operator^= . . . . .                                 | 676        |
| 9.163.4.31 | operator^= . . . . .                                 | 676        |
| 9.163.4.32 | pop_back . . . . .                                   | 676        |
| 9.163.4.33 | pop_front . . . . .                                  | 677        |
| 9.163.4.34 | printOn . . . . .                                    | 677        |
| 9.163.4.35 | push_back . . . . .                                  | 677        |
| 9.163.4.36 | push_front . . . . .                                 | 677        |
| 9.163.4.37 | replace . . . . .                                    | 677        |
| 9.163.4.38 | replace . . . . .                                    | 677        |
| 9.163.4.39 | replace . . . . .                                    | 677        |
| 9.163.4.40 | segment . . . . .                                    | 677        |
| 9.163.4.41 | terminalSegment . . . . .                            | 677        |
| 9.163.5    | Friends And Related Function Documentation . . . . . | 677        |
| 9.163.5.1  | Word . . . . .                                       | 677        |
| 9.163.6    | Member Data Documentation . . . . .                  | 677        |
| 9.163.6.1  | theElements . . . . .                                | 677        |
| 9.164      | YYSTYPE Union Reference . . . . .                    | 679        |
| 9.164.1    | Detailed Description . . . . .                       | 679        |
| 9.164.2    | Member Data Documentation . . . . .                  | 679        |
| 9.164.2.1  | lst . . . . .  | 679        |
| 9.164.2.2  | num . . . . .  | 679        |
| 9.164.2.3  | str . . . . .  | 679        |
| <b>10</b>  | <b>File Documentation</b>                            | <b>681</b> |
| 10.1       | Alphabet/include/Alphabet.h File Reference . . . . . | 681        |
| 10.1.1     | Function Documentation . . . . .                     | 682        |
| 10.1.1.1   | readFPPresentation . . . . .                         | 682        |

|           |  |     |
|-----------|--|-----|
| 10.2      | Alphabet/include/AlphabetBisonGrammar.h File Reference . . . . .   | 683 |
| 10.2.1    | Define Documentation . . . . .                                     | 683 |
| 10.2.1.1  | yystype . . . . .  | 683 |
| 10.2.1.2  | YYSTYPE_IS_DECLARED . . . . .                                      | 683 |
| 10.2.1.3  | YYSTYPE_IS_TRIVIAL . . . . .                                       | 683 |
| 10.2.2    | Enumeration Type Documentation . . . . .                           | 684 |
| 10.2.2.1  | y tokentype . . . . .  | 684 |
| 10.2.3    | Variable Documentation . . . . .                                   | 684 |
| 10.2.3.1  | aalval . . . . .   | 684 |
| 10.3      | Alphabet/include/Parser.h File Reference . . . . .                 | 685 |
| 10.3.1    | Enumeration Type Documentation . . . . .                           | 685 |
| 10.3.1.1  | AInputType . . . . .   | 685 |
| 10.4      | Alphabet/include/WordBisonGrammar.h File Reference . . . . .       | 686 |
| 10.4.1    | Enumeration Type Documentation . . . . .                           | 686 |
| 10.4.1.1  | y tokentype . . . . .  | 686 |
| 10.4.2    | Variable Documentation . . . . .                                   | 686 |
| 10.4.2.1  | yylval . . . . .   | 686 |
| 10.5      | BraidGroup/include/BKLLeftNormalForm.h File Reference . . . . .    | 687 |
| 10.5.1    | Function Documentation . . . . .                                   | 687 |
| 10.5.1.1  | operator<< . . . . .   | 687 |
| 10.6      | BraidGroup/include/BKLRightNormalForm.h File Reference . . . . .   | 688 |
| 10.6.1    | Define Documentation . . . . .                                     | 688 |
| 10.6.1.1  | ERROR_EXISTS . . . . .   | 688 |
| 10.6.2    | Function Documentation . . . . .                                   | 688 |
| 10.6.2.1  | operator<< . . . . .   | 688 |
| 10.7      | BraidGroup/include/BraidGroup.h File Reference . . . . .           | 689 |
| 10.8      | BraidGroup/include/DehornoyForm.h File Reference . . . . .         | 690 |
| 10.9      | BraidGroup/include/DehornoyForm_old.h File Reference . . . . .     | 691 |
| 10.10     | BraidGroup/include/LinkedBraidStructure.h File Reference . . . . . | 692 |
| 10.10.1   | Function Documentation . . . . .                                   | 692 |
| 10.10.1.1 | operator<< . . . . .   | 692 |

|   |     |
|---|-----|
| 10.11BraidGroup/include/LinkedBraidStructure_old.h File Reference . . . | 693 |
| 10.12BraidGroup/include/ShortBraidForm.h File Reference . . . . .       | 694 |
| 10.12.1 Function Documentation . . . . .                                | 694 |
| 10.12.1.1 shortBraidForm . . . . .                                      | 694 |
| 10.12.1.2 shortBraidSbgpForm . . . . .                                  | 694 |
| 10.12.1.3 shortenBraid . . . . .  | 694 |
| 10.12.1.4 shortenLBS . . . . .  | 694 |
| 10.13BraidGroup/include/ShortBraidForm_old.h File Reference . . . . .   | 695 |
| 10.13.1 Function Documentation . . . . .                                | 695 |
| 10.13.1.1 shortBraidForm . . . . .                                      | 695 |
| 10.13.1.2 shortBraidSbgpForm . . . . .                                  | 695 |
| 10.13.1.3 shortenBraid . . . . .  | 695 |
| 10.14BraidGroup/include/ThLeftNormalForm.h File Reference . . . . .     | 696 |
| 10.14.1 Function Documentation . . . . .                                | 696 |
| 10.14.1.1 operator<< . . . . .  | 696 |
| 10.15BraidGroup/include/ThRightNormalForm.h File Reference . . . . .    | 697 |
| 10.15.1 Function Documentation . . . . .                                | 697 |
| 10.15.1.1 operator<< . . . . .  | 697 |
| 10.16BraidGroup/include/ThRightNormalFormAlgorithms.h File Reference    | 698 |
| 10.16.1 Function Documentation . . . . .                                | 698 |
| 10.16.1.1 getSimpleConjugator . . . . .                                 | 698 |
| 10.16.1.2 getSimpleConjugators . . . . .                                | 698 |
| 10.16.1.3 getSummitSetRepresentative . . . . .                          | 698 |
| 10.17CryptoAAG/include/AAGChallengeGeneration.h File Reference . . .    | 699 |
| 10.18CryptoAAG/include/AAGKeyGeneration.h File Reference . . . . .      | 700 |
| 10.18.1 Function Documentation . . . . .                                | 700 |
| 10.18.1.1 conjugateSubgroup_PGPF . . . . .                              | 700 |
| 10.19CryptoAAG/include/LengthAttack.h File Reference . . . . .          | 701 |
| 10.19.1 Define Documentation . . . . .                                  | 702 |
| 10.19.1.1 AL1 . . . . .   | 702 |
| 10.19.1.2 AL2 . . . . .   | 702 |

|  |     |
|--|-----|
| 10.19.1.3 AL3 . . . . .  | 702 |
| 10.19.2 Typedef Documentation . . . . .  | 702 |
| 10.19.2.1 ELT . . . . .  | 702 |
| 10.19.3 Enumeration Type Documentation . . . . .   | 702 |
| 10.19.3.1 findKey_LengthBasedResult . . . . .  | 702 |
| 10.20CryptoAAG/include/MajorDump.h File Reference . . . . .  | 703 |
| 10.21CryptoAE/include/AEProtocol.h File Reference . . . . .  | 704 |
| 10.21.1 Typedef Documentation . . . . .  | 704 |
| 10.21.1.1 BureauGenerator . . . . .  | 704 |
| 10.21.1.2 ProdElement . . . . .  | 704 |
| 10.22CryptoAE/include/TTPAttack.h File Reference . . . . .   | 705 |
| 10.22.1 Typedef Documentation . . . . .  | 705 |
| 10.22.1.1 NODE . . . . .   | 705 |
| 10.23CryptoKL/include/KLKeyGeneration.h File Reference . . . . .                                   | 706 |
| 10.24CryptoShftConj/include/ShftConjKeyGeneration.h File Reference . . . . .                       | 707 |
| 10.24.1 Function Documentation . . . . .   | 707 |
| 10.24.1.1 generatorShift . . . . .   | 707 |
| 10.25CryptoShftConj/include/ShftConjKeyGenerationGarside.h File Reference . . . . .                | 708 |
| 10.25.1 Function Documentation . . . . .   | 708 |
| 10.25.1.1 shiftedConjugation . . . . .   | 708 |
| 10.26CryptoTripleDecomposition/include/TripleDecompositionKeyGeneration.h File Reference . . . . . | 709 |
| 10.27Elt/include/PowerWord.h File Reference . . . . .  | 710 |
| 10.28Elt/include/PowerWordIterator.h File Reference . . . . .                                      | 711 |
| 10.29Elt/include/PowerWordRep.h File Reference . . . . .   | 712 |
| 10.30Elt/include/Word.h File Reference . . . . .   | 713 |
| 10.31Elt/include/WordIterator.h File Reference . . . . .   | 714 |
| 10.32Elt/include/WordRep.h File Reference . . . . .  | 715 |
| 10.32.1 Typedef Documentation . . . . .  | 715 |
| 10.32.1.1 Generator . . . . .  | 715 |
| 10.33Equation/include/Equation.h File Reference . . . . .  | 716 |

|  |     |
|--|-----|
| 10.34Equation/include/QuadEquatTransformationGraph.h File Reference . . . . .                | 717 |
| 10.35Experiments/include/AAGKeyPairGenerator.h File Reference . . . . .                      | 718 |
| 10.36Experiments/include/DCBraidReduction.h File Reference . . . . .                         | 719 |
| 10.37Experiments/include/DDL.h File Reference . . . . .                                      | 720 |
| 10.38Experiments/include/FRDFIT.h File Reference . . . . .                                   | 721 |
| 10.39FreeGroup/include/FreeGroup.h File Reference . . . . .                                  | 722 |
| 10.40FreeGroup/include/StraightLineProgramWord.h File Reference . . . . .                    | 723 |
| 10.40.1 Typedef Documentation . . . . .  | 723 |
| 10.40.1.1 LongInteger . . . . .  | 723 |
| 10.41FreeGroup/include/WhiteheadGraph.h File Reference . . . . .                             | 724 |
| 10.41.1 Typedef Documentation . . . . .  | 724 |
| 10.41.1.1 Generator . . . . .  | 724 |
| 10.42SbgpFG/include/WhiteheadGraph.h File Reference . . . . .                                | 725 |
| 10.43FreeMetabelianGroup/include/FreeMetabelianGroupAlgorithms.h<br>File Reference . . . . . | 726 |
| 10.44general/include/BalancedTree.h File Reference . . . . .                                 | 727 |
| 10.45general/include/ConfigFile.h File Reference . . . . .                                   | 728 |
| 10.45.1 Typedef Documentation . . . . .  | 728 |
| 10.45.1.1 ParameterType . . . . .  | 728 |
| 10.46general/include/dump.h File Reference . . . . .   | 729 |
| 10.47general/include/errmsgs.h File Reference . . . . .                                      | 730 |
| 10.48general/include/FormatOutput.h File Reference . . . . .                                 | 731 |
| 10.49general/include/ObjectOf.h File Reference . . . . .                                     | 732 |
| 10.50general/include/Permutation.h File Reference . . . . .                                  | 733 |
| 10.51general/include/PermutationEnumerator.h File Reference . . . . .                        | 734 |
| 10.52general/include/ProgressBar.h File Reference . . . . .                                  | 735 |
| 10.53general/include/RefCounter.h File Reference . . . . .                                   | 736 |
| 10.54general/include/tuples.h File Reference . . . . .                                       | 737 |
| 10.55general/include/VectorEnumerator.h File Reference . . . . .                             | 738 |
| 10.56Graph/include/FSA.h File Reference . . . . .  | 739 |
| 10.56.1 Function Documentation . . . . .   | 739 |

|  |     |
|--|-----|
| 10.56.1.1 operator<< . . . . .                                       | 739 |
| 10.57Graph/include/FSAREp.h File Reference . . . . .                 | 740 |
| 10.57.1 Function Documentation . . . . .                             | 740 |
| 10.57.1.1 reducePath . . . . .                                       | 740 |
| 10.58Graph/include/Graph.h File Reference . . . . .                  | 741 |
| 10.58.1 Function Documentation . . . . .                             | 741 |
| 10.58.1.1 getHyperbolicityConst . . . . .                            | 741 |
| 10.58.1.2 innerProductTable . . . . .                                | 741 |
| 10.58.1.3 lengthTable . . . . .                                      | 741 |
| 10.58.1.4 operator<< . . . . .                                       | 742 |
| 10.58.1.5 randomGraph . . . . .                                      | 742 |
| 10.59Graph/include/GraphAlgorithms.h File Reference . . . . .        | 743 |
| 10.59.1 Function Documentation . . . . .                             | 743 |
| 10.59.1.1 getDistances_out . . . . .                                 | 743 |
| 10.59.1.2 getGeodesicTree_in . . . . .                               | 744 |
| 10.59.1.3 getGeodesicTree_out . . . . .                              | 744 |
| 10.59.1.4 readoffGeodesicTree . . . . .                              | 744 |
| 10.59.1.5 trace . . . . .  | 744 |
| 10.59.1.6 trace_path . . . . .                                       | 744 |
| 10.60Graph/include/GraphConcept.h File Reference . . . . .           | 745 |
| 10.61Graph/include/GraphConceptAlgorithms.h File Reference . . . . . | 746 |
| 10.62Graph/include/GraphDrawingAttributes.h File Reference . . . . . | 748 |
| 10.63Graph/include/GraphRep.h File Reference . . . . .               | 749 |
| 10.64Graph/include/GraphType.h File Reference . . . . .              | 750 |
| 10.64.1 Typedef Documentation . . . . .                              | 752 |
| 10.64.1.1 Graph . . . . .  | 752 |
| 10.64.1.2 IntLabeledGraph . . . . .                                  | 752 |
| 10.64.2 Function Documentation . . . . .                             | 752 |
| 10.64.2.1 addLoop . . . . .  | 752 |
| 10.64.2.2 addRay . . . . .   | 752 |
| 10.64.2.3 getHyperbolicityConst . . . . .                            | 752 |

|  |     |
|--|-----|
| 10.64.2.4 graphviz_format . . . . .  | 753 |
| 10.64.2.5 graphviz_format . . . . .  | 753 |
| 10.64.2.6 innerProductTable . . . . .                                      | 753 |
| 10.64.2.7 lengthTable . . . . .  | 753 |
| 10.64.2.8 operator<< . . . . .   | 753 |
| 10.64.2.9 operator<< . . . . .   | 753 |
| 10.64.2.10prepareToFold . . . . .  | 753 |
| 10.64.2.11randomGraph . . . . .  | 753 |
| 10.65Graph/include/RandomFSA.h File Reference . . . . .                    | 754 |
| 10.65.1 Function Documentation . . . . .                                   | 754 |
| 10.65.1.1 randomFSA . . . . .  | 754 |
| 10.66Graphics/include/AImage.h File Reference . . . . .                    | 755 |
| 10.66.1 Enumeration Type Documentation . . . . .                           | 755 |
| 10.66.1.1 FILE_TYPE . . . . .  | 755 |
| 10.66.1.2 IMAGE_TYPE . . . . .   | 756 |
| 10.66.2 Function Documentation . . . . .                                   | 756 |
| 10.66.2.1 convert . . . . .  | 756 |
| 10.66.2.2 getFileType . . . . .  | 756 |
| 10.66.3 Variable Documentation . . . . .                                   | 756 |
| 10.66.3.1 MAXGRAY . . . . .  | 756 |
| 10.67Graphics/include/PDFgraphing.h File Reference . . . . .               | 757 |
| 10.68Graphics/include/WordDraw.h File Reference . . . . .                  | 758 |
| 10.68.1 Variable Documentation . . . . .                                   | 758 |
| 10.68.1.1 ssConst . . . . .  | 758 |
| 10.69Group/include/AdvDehnAlgorithm.h File Reference . . . . .             | 759 |
| 10.70Group/include/FPGroup.h File Reference . . . . .                      | 760 |
| 10.71HigmanGroup/include/BaumslagGersten.h File Reference . . . . .        | 761 |
| 10.72HigmanGroup/include/PowerCircuit.h File Reference . . . . .           | 762 |
| 10.73HigmanGroup/include/PowerCircuitCompMatrix.h File Reference . . . . . | 763 |
| 10.74HigmanGroup/include/PowerCircuitGraph.h File Reference . . . . .      | 764 |
| 10.75HigmanGroup/include/Sign.h File Reference . . . . .                   | 765 |



|  |     |
|--|-----|
| 10.76HigmanGroup/include/SignMatrix.h File Reference . . . . . | 766 |
| 10.77Maps/include/Map.h File Reference . . . . .               | 767 |
| 10.78Maps/include/WhiteheadAutoSet.h File Reference . . . . .  | 768 |
| 10.78.1 Typedef Documentation . . . . .                        | 769 |
| 10.78.1.1 SetOfMaps . . . . .                                  | 769 |
| 10.79ranlib/include/cdflib.h File Reference . . . . .          | 770 |
| 10.79.1 Function Documentation . . . . .                       | 773 |
| 10.79.1.1 algdiv . . . . .                                     | 773 |
| 10.79.1.2 alngam . . . . .                                     | 773 |
| 10.79.1.3 alnrel . . . . .                                     | 773 |
| 10.79.1.4 apser . . . . .                                      | 773 |
| 10.79.1.5 basym . . . . .                                      | 773 |
| 10.79.1.6 bcorr . . . . .                                      | 773 |
| 10.79.1.7 betaln . . . . .                                     | 773 |
| 10.79.1.8 bfrac . . . . .                                      | 773 |
| 10.79.1.9 bgrat . . . . .                                      | 773 |
| 10.79.1.10bpser . . . . .                                      | 773 |
| 10.79.1.11bratio . . . . .                                     | 773 |
| 10.79.1.12brcmpl . . . . .                                     | 773 |
| 10.79.1.13brcomp . . . . .                                     | 773 |
| 10.79.1.14bup . . . . .  | 773 |
| 10.79.1.15cdfbet . . . . .                                     | 773 |
| 10.79.1.16cdfbin . . . . .                                     | 773 |
| 10.79.1.17cdfchi . . . . .                                     | 773 |
| 10.79.1.18cdfchn . . . . .                                     | 774 |
| 10.79.1.19cdf . . . . .  | 774 |
| 10.79.1.20cdfnc . . . . .                                      | 774 |
| 10.79.1.21cdfgam . . . . .                                     | 774 |
| 10.79.1.22cdfnbn . . . . .                                     | 774 |
| 10.79.1.23cdfnor . . . . .                                     | 774 |
| 10.79.1.24cdfpoi . . . . .                                     | 774 |

|                  |     |
|------------------|-----|
| 10.79.1.25cdf    | 774 |
| 10.79.1.26cumbet | 774 |
| 10.79.1.27cumbin | 774 |
| 10.79.1.28cumchi | 774 |
| 10.79.1.29cumchn | 774 |
| 10.79.1.30cumf   | 774 |
| 10.79.1.31cumfnc | 774 |
| 10.79.1.32cumgam | 774 |
| 10.79.1.33cumnbn | 774 |
| 10.79.1.34cumnor | 774 |
| 10.79.1.35cumpoi | 774 |
| 10.79.1.36cumt   | 774 |
| 10.79.1.37dbetrm | 774 |
| 10.79.1.38devlpl | 774 |
| 10.79.1.39dexpml | 774 |
| 10.79.1.40dinvr  | 774 |
| 10.79.1.41dinvr  | 774 |
| 10.79.1.42dlanor | 774 |
| 10.79.1.43dlnlmx | 774 |
| 10.79.1.44dlnlpx | 774 |
| 10.79.1.45dlnbet | 774 |
| 10.79.1.46dlngam | 774 |
| 10.79.1.47dstinv | 774 |
| 10.79.1.48dstrem | 774 |
| 10.79.1.49dstzr  | 774 |
| 10.79.1.50dt1    | 774 |
| 10.79.1.51dzror  | 774 |
| 10.79.1.52E0000  | 774 |
| 10.79.1.53E0001  | 774 |
| 10.79.1.54erf1   | 774 |
| 10.79.1.55erfc1  | 774 |

|            |  |     |
|------------|--|-----|
| 10.79.1.56 | esum . . . . .                                   | 774 |
| 10.79.1.57 | exparg . . . . .                                 | 774 |
| 10.79.1.58 | ifdint . . . . .                                 | 774 |
| 10.79.1.59 | ifdmax1 . . . . .                                | 774 |
| 10.79.1.60 | ifdmin1 . . . . .                                | 774 |
| 10.79.1.61 | ifdsign . . . . .                                | 774 |
| 10.79.1.62 | ifidint . . . . .                                | 774 |
| 10.79.1.63 | ifmod . . . . .                                  | 774 |
| 10.79.1.64 | fpser . . . . .                                  | 774 |
| 10.79.1.65 | ftnstop . . . . .                                | 774 |
| 10.79.1.66 | gam1 . . . . .                                   | 774 |
| 10.79.1.67 | gaminv . . . . .                                 | 774 |
| 10.79.1.68 | gamln . . . . .                                  | 774 |
| 10.79.1.69 | gamln1 . . . . .                                 | 774 |
| 10.79.1.70 | grat1 . . . . .                                  | 774 |
| 10.79.1.71 | lgratio . . . . .                                | 774 |
| 10.79.1.72 | gsumln . . . . .                                 | 774 |
| 10.79.1.73 | ipmpar . . . . .                                 | 774 |
| 10.79.1.74 | psi . . . . .                                    | 774 |
| 10.79.1.75 | rcomp . . . . .                                  | 774 |
| 10.79.1.76 | rex . . . . .                                    | 774 |
| 10.79.1.77 | rlog . . . . .                                   | 774 |
| 10.79.1.78 | rlog1 . . . . .                                  | 774 |
| 10.79.1.79 | spmpar . . . . .                                 | 774 |
| 10.79.1.80 | stvaln . . . . .                                 | 774 |
| 10.79.1.81 | Xgamm . . . . .                                  | 774 |
| 10.80      | ranlib/include/ranlib.h File Reference . . . . . | 775 |
| 10.80.1    | Function Documentation . . . . .                 | 776 |
| 10.80.1.1  | advnst . . . . .                                 | 776 |
| 10.80.1.2  | genbet . . . . .                                 | 776 |
| 10.80.1.3  | genchi . . . . .                                 | 776 |

|  |     |
|--|-----|
| 10.80.1.4 genexp . . . . .                                 | 776 |
| 10.80.1.5 genf . . . . .                                   | 776 |
| 10.80.1.6 gengam . . . . .                                 | 776 |
| 10.80.1.7 genmn . . . . .                                  | 776 |
| 10.80.1.8 genmul . . . . .                                 | 776 |
| 10.80.1.9 gennch . . . . .                                 | 776 |
| 10.80.1.10 gennf . . . . .                                 | 776 |
| 10.80.1.11 gennor . . . . .                                | 776 |
| 10.80.1.12 genprm . . . . .                                | 776 |
| 10.80.1.13 genunf . . . . .                                | 776 |
| 10.80.1.14 getsd . . . . .                                 | 776 |
| 10.80.1.15 gscgn . . . . .                                 | 777 |
| 10.80.1.16 gnbin . . . . .                                 | 777 |
| 10.80.1.17 gnlgi . . . . .                                 | 777 |
| 10.80.1.18 gnnbn . . . . .                                 | 777 |
| 10.80.1.19 gnpoi . . . . .                                 | 777 |
| 10.80.1.20 guin . . . . .                                  | 777 |
| 10.80.1.21 linitgn . . . . .                               | 777 |
| 10.80.1.22 mltmod . . . . .                                | 777 |
| 10.80.1.23 phrtsd . . . . .                                | 777 |
| 10.80.1.24 ranf . . . . .                                  | 777 |
| 10.80.1.25 setall . . . . .                                | 777 |
| 10.80.1.26 setant . . . . .                                | 777 |
| 10.80.1.27 setgm . . . . .                                 | 777 |
| 10.80.1.28 setsd . . . . .                                 | 777 |
| 10.80.1.29 sexpo . . . . .                                 | 777 |
| 10.80.1.30 gamma . . . . .                                 | 777 |
| 10.80.1.31 lsnorm . . . . .                                | 777 |
| 10.81 ranlib/include/RanlibCPP.h File Reference . . . . .  | 778 |
| 10.82 SbgpFG/include/SubgroupFG.h File Reference . . . . . | 779 |
| 10.82.1 Function Documentation . . . . .                   | 779 |

---

|  |     |
|--|-----|
| 10.82.1.1 operator<< . . . . .   | 779 |
| 10.82.1.2 substitute . . . . .   | 779 |
| 10.82.1.3 substitute . . . . .   | 779 |
| 10.83StringSimilarity/include/Levenstein.h File Reference . . . . .                        | 780 |
| 10.84StringSimilarity/include/MotivePatternWrapper.h File Reference . . .                  | 781 |
| 10.85StringSimilarity/include/PairDistanceTest.h File Reference . . . . .                  | 782 |
| 10.86StringSimilarity/include/SimilarityMeasures.h File Reference . . . . .                | 783 |
| 10.86.1 Function Documentation . . . . .   | 784 |
| 10.86.1.1 getLeftTaleConfidenceValue . . . . .   | 784 |
| 10.87StringSimilarity/include/StringScramblers.h File Reference . . . . .                  | 785 |
| 10.88TheGrigorchukGroup/include/TheGrigorchukGroupAlgorithms.h File<br>Reference . . . . . | 786 |
| 10.89ThompsonGroup/include/ThompsonGroupFNormalForm.h File Refer-<br>ence . . . . .        | 787 |



# Chapter 1

## Bug List

**Class ConfigFile (p. 168)** On some platforms scanning may hang if an empty string is contained in the input file.





## Chapter 2

# Directory Hierarchy

### 2.1 Directories

This directory hierarchy is sorted roughly, but not completely, alphabetically:

|                                     |    |
|-------------------------------------|----|
| Alphabet . . . . .                  | 23 |
| include . . . . .                   | 62 |
| BraidGroup . . . . .                | 24 |
| include . . . . .                   | 61 |
| CryptoAAG . . . . .                 | 25 |
| include . . . . .                   | 60 |
| CryptoAE . . . . .                  | 26 |
| include . . . . .                   | 59 |
| CryptoKL . . . . .                  | 27 |
| include . . . . .                   | 58 |
| CryptoShftConj . . . . .            | 28 |
| include . . . . .                   | 57 |
| CryptoTripleDecomposition . . . . . | 29 |
| include . . . . .                   | 56 |
| Elt . . . . .                       | 30 |
| include . . . . .                   | 55 |
| Equation . . . . .                  | 31 |
| include . . . . .                   | 54 |
| Experiments . . . . .               | 32 |
| include . . . . .                   | 53 |
| FreeGroup . . . . .                 | 33 |
| include . . . . .                   | 52 |

|                               |    |
|-------------------------------|----|
| FreeMetabelianGroup . . . . . | 34 |
| include . . . . .             | 50 |
| general . . . . .             | 35 |
| include . . . . .             | 49 |
| Graph . . . . .               | 36 |
| include . . . . .             | 48 |
| Graphics . . . . .            | 37 |
| include . . . . .             | 47 |
| Group . . . . .               | 38 |
| include . . . . .             | 46 |
| HigmanGroup . . . . .         | 39 |
| include . . . . .             | 45 |
| Maps . . . . .                | 63 |
| include . . . . .             | 44 |
| ranlib . . . . .              | 64 |
| include . . . . .             | 43 |
| SbgpFG . . . . .              | 65 |
| include . . . . .             | 51 |
| StringSimilarity . . . . .    | 66 |
| include . . . . .             | 42 |
| TheGrigorchukGroup . . . . .  | 67 |
| include . . . . .             | 41 |
| ThompsonGroup . . . . .       | 68 |
| include . . . . .             | 40 |

# Chapter 3

## Namespace Index

### 3.1 Namespace List

Here is a list of all namespaces with brief descriptions:

|  |    |
|--|----|
| <b>__gnu_cxx</b> . . . . .   | 69 |
| <b>AAGChallenge</b> . . . . .  | 70 |
| <b>BG</b> . . . . .  | 72 |
| <b>Graphs</b> . . . . .  | 73 |
| <b>msgs</b> (Errors output handling ) . . . . .                      | 78 |
| <b>PC</b> . . . . .  | 80 |
| <b>RMap</b> (Namespace for random map generating methods ) . . . . . | 84 |
| <b>WhiteheadAutoSet</b> . . . . .                                    | 85 |



## Chapter 4

# Class Index

### 4.1 Class Hierarchy

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

|  |     |
|--|-----|
| AAGProtocolInstance . . . . .                | 90  |
| AdvDehnAlgorithm . . . . .                   | 94  |
| AEKeyExchange . . . . .                      | 97  |
| AImage . . . . .                             | 100 |
| CImage . . . . .                             | 152 |
| CLUTImage . . . . .                          | 159 |
| GRImage . . . . .                            | 262 |
| GRLUTImage . . . . .                         | 266 |
| Alphabet . . . . .                           | 103 |
| FiniteAlphabet . . . . .                     | 199 |
| InfiniteAlphabet . . . . .                   | 272 |
| AParser . . . . .                            | 105 |
| StraightLineProgramWord::Assertion . . . . . | 107 |
| AutoSet . . . . .                            | 110 |
| NielsenAutoSet . . . . .                     | 341 |
| RestrictedWhiteheadAutoSet . . . . .         | 469 |
| WhiteheadAutoSetType2 . . . . .              | 625 |
| BalancedTree< Obj > . . . . .                | 111 |
| BG::BGMonomial . . . . .                     | 116 |
| BKLLeftNormalForm . . . . .                  | 118 |
| BKLRightNormalForm . . . . .                 | 127 |
| BraidDrawPDF . . . . .                       | 135 |
| BraidGroup . . . . .                         | 140 |
| BraidNode . . . . .                          | 142 |

|  |     |
|--|-----|
| BSets  | 147 |
| BalancedTree< Obj >::BTNode                                | 149 |
| PC::Col  | 162 |
| PC::PowerCircuitCompMatrix::ColHdr                         | 164 |
| CommDivider  | 165 |
| compMaps   | 167 |
| ConfigFile   | 168 |
| ConstWordIterator  | 172 |
| CutVertices  | 176 |
| DCBraidReduction   | 181 |
| DDL  | 183 |
| DDLNode  | 185 |
| DehornoyForm   | 187 |
| dump< T >  | 190 |
| Dump   | 192 |
| Equation   | 195 |
| FoldDetails  | 200 |
| Graphs::FoldDetails< VertexType, EdgeType >                | 202 |
| FPGGroup   | 204 |
| FRDFIT   | 209 |
| FreeGroup  | 211 |
| FreeMetabelianGroupAlgorithms                              | 214 |
| FSAEdge  | 222 |
| FSASState  | 231 |
| GraphDrawingAttributes                                     | 246 |
| GraphEdge  | 251 |
| IntLabeledEdge   | 277 |
| PlanarGraphIntLabelledEdge                                 | 392 |
| PlanarGraphEdge  | 388 |
| PlanarGraphIntLabelledEdge                                 | 392 |
| GraphState   | 258 |
| GraphVertex< EdgeType >                                    | 260 |
| PC::SignMatrix< RowHdr, ColHdr, INTERNAL_TYPE >::intColHdr | 275 |
| PC::PowerCircuitGraph::IntMarking                          | 280 |
| PC::PowerCircuitCompMatrix::IntMarking                     | 281 |
| PC::PowerCircuitCompMatrix::IntNode                        | 283 |
| PC::PowerCircuitGraph::IntNode                             | 284 |
| PC::SignMatrix< RowHdr, ColHdr, INTERNAL_TYPE >::intRowHdr | 285 |
| KLProtocolInstance   | 287 |
| lCommDivider   | 291 |
| LengthAttackBase   | 301 |
| LengthAttack_A1  | 292 |
| LengthAttack_A2  | 295 |
| LengthAttack_A3  | 298 |
| Levenstein   | 303 |

|   |     |
|---|-----|
| LinkedBraidStructure . . . . .                                | 305 |
| LinkedBraidStructureTransform . . . . .                       | 312 |
| LookUpTable . . . . .   | 315 |
| Itstr . . . . .   | 317 |
| LUTImage . . . . .  | 318 |
| CLUTImage . . . . .   | 159 |
| GRLUTImage . . . . .  | 266 |
| Map . . . . .   | 320 |
| __gnu_cxx::map_hash . . . . .                                 | 328 |
| PC::Marking . . . . .   | 329 |
| MatrixFp . . . . .  | 333 |
| MaxCommutePartition . . . . .                                 | 336 |
| MotivePattern . . . . .                                       | 338 |
| MotivePatternWrapper . . . . .                                | 339 |
| PC::Node . . . . .  | 344 |
| PC::PowerCircuitGraph::NodeUsedByType . . . . .               | 346 |
| ObjectOf< Rep > . . . . .                                     | 348 |
| ObjectOf< FSAREp > . . . . .                                  | 348 |
| FSA . . . . .   | 216 |
| ObjectOf< GraphConceptRep< VertexType, EdgeType > > . . . . . | 348 |
| Graphs::GraphConcept< VertexType, EdgeType > . . . . .        | 235 |
| ObjectOf< GraphRep > . . . . .                                | 348 |
| Graph . . . . .   | 233 |
| ObjectOf< PowerWordRep > . . . . .                            | 348 |
| PowerWord . . . . .   | 431 |
| ObjectOf< WordRep > . . . . .                                 | 348 |
| Word . . . . .  | 642 |
| ThompsonGroupFNormalForm . . . . .                            | 556 |
| PairDistanceSimilarityTest . . . . .                          | 351 |
| PairGenerator . . . . .                                       | 353 |
| AAGKeyPairGenerator . . . . .                                 | 87  |
| RandomPairGenerator . . . . .                                 | 463 |
| Parser . . . . .  | 355 |
| Partition . . . . .   | 357 |
| UniformPartition . . . . .                                    | 614 |
| PBar . . . . .  | 358 |
| PDFPage . . . . .   | 360 |
| PDFPageObject . . . . .                                       | 365 |
| PDFPageObjectLine . . . . .                                   | 366 |
| udPDFPageObjectCircle . . . . .                               | 598 |
| udPDFPageObjectLine . . . . .                                 | 602 |
| udPDFPageObjectHorizLine . . . . .                            | 600 |
| udPDFPageObjectVertLine . . . . .                             | 612 |

|   |     |
|---|-----|
| udPDFPageObjectRect . . . . .                             | 605 |
| udPDFPageObjectSquare . . . . .                           | 608 |
| udPDFPageObjectText . . . . .                             | 610 |
| PDFStructure . . . . .                                    | 368 |
| Permutation . . . . .                                     | 371 |
| Perturbation . . . . .                                    | 387 |
| PC::PowerCircuit . . . . .                                | 395 |
| PC::PowerCircuitCompMatrix . . . . .                      | 404 |
| PC::PowerCircuitGraph . . . . .                           | 418 |
| StraightLineProgramWord::Production . . . . .             | 446 |
| QuadEquationTranformationGraph . . . . .                  | 448 |
| quadruple< T1, T2, T3, T4 > . . . . .                     | 452 |
| quintuple< T1, T2, T3, T4, T5 > . . . . .                 | 455 |
| RandLib . . . . .   | 458 |
| RandLibURG . . . . .                                      | 460 |
| RefCounter . . . . .                                      | 466 |
| FSAREp . . . . .  | 224 |
| GraphRep . . . . .  | 255 |
| Graphs::GraphConceptRep< VertexType, EdgeType > . . . . . | 240 |
| PowerWordRep . . . . .                                    | 440 |
| WordRep . . . . .   | 671 |
| RGB . . . . .   | 472 |
| PC::Row . . . . .   | 474 |
| PC::PowerCircuitCompMatrix::RowHdr . . . . .              | 476 |
| ShftConjKeyInstance . . . . .                             | 478 |
| ShftConjKeyInstanceGarside . . . . .                      | 481 |
| PC::SignMatrix< RowHdr, ColHdr, INTERNAL_TYPE > . . . . . | 484 |
| StraightLineProgramWord . . . . .                         | 507 |
| StringScrambler . . . . .                                 | 519 |
| SubwordScrambler . . . . .                                | 533 |
| UniformScrambler . . . . .                                | 615 |
| WordMultiplyScrambler . . . . .                           | 666 |
| StringSimilarityMeasure . . . . .                         | 520 |
| EditingDistance . . . . .                                 | 193 |
| HammingDistance . . . . .                                 | 268 |
| HammingDistanceCyclic . . . . .                           | 270 |
| SubwordEditingDistanceCyclic . . . . .                    | 529 |
| SubwordHammingDistanceCyclic . . . . .                    | 531 |
| SubgroupFG . . . . .                                      | 522 |
| TheGrigorchukGroupAlgorithms . . . . .                    | 535 |
| ThLeftNormalForm . . . . .                                | 541 |
| ThRightNormalForm . . . . .                               | 560 |
| Permutation::triple . . . . .                             | 576 |
| triple< T1, T2, T3 > . . . . .                            | 577 |



---

|   |     |
|---|-----|
| TripleDecompositionProtocolInstance . . . . . | 580 |
| TTP_Conf . . . . .                            | 586 |
| TTPAttack . . . . .                           | 588 |
| TTPLBA . . . . .                              | 592 |
| TTPTuple . . . . .                            | 594 |
| Value . . . . .                               | 617 |
| VectorEnumerator . . . . .                    | 620 |
| PermutationEnumerator . . . . .               | 382 |
| WhiteheadGraph . . . . .                      | 628 |
| WhiteheadMultiGraph . . . . .                 | 634 |
| WhiteheadSimpleGraph . . . . .                | 635 |
| WhiteheadMinimization . . . . .               | 631 |
| WordDraw . . . . .                            | 658 |
| WordIterator . . . . .                        | 662 |
| WordPairComparison . . . . .                  | 668 |
| YYSTYPE . . . . .                             | 679 |



## Chapter 5

# Class Index

### 5.1 Class List

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

|  |     |
|--|-----|
| <b>AAGKeyPairGenerator</b> . . . . .   | 87  |
| <b>AAGProtocolInstance</b> . . . . .   | 90  |
| <b>AdvDehnAlgorithm</b> . . . . .  | 94  |
| <b>AEKeyExchange</b> . . . . .   | 97  |
| <b>AImage</b> . . . . .  | 100 |
| <b>Alphabet</b> (Implements an abstract interface for all alphabet realizations ) . .  | 103 |
| <b>AParser</b> . . . . .   | 105 |
| <b>StraightLineProgramWord::Assertion</b> (Structure used in function <b>equal()</b><br>(p. 512) only ) . . . . .  | 107 |
| <b>AutoSet</b> (Abstract interface for a set of Maps (Automorphisms) ) . . . . .   | 110 |
| <b>BalancedTree&lt; Obj &gt;</b> (Class <b>BalancedTree</b> (p. 111) (container class in-<br>tended to keep an ordered sequence of objects, supports fast $n \log n$<br>ninsertions) ) . . . . . | 111 |
| <b>BG::BGMonomial</b> . . . . .  | 116 |
| <b>BKLLeftNormalForm</b> (Birman-Ko-Lee left normal form ) . . . . .   | 118 |
| <b>BKLRightNormalForm</b> ((Contains errors!!! Not to be used yet) Birman-<br>Ko-Lee right normal form ) . . . . .   | 127 |
| <b>BraidDrawPDF</b> (Class for drawing a braid on n strands as a square table ) . .  | 135 |
| <b>BraidGroup</b> (Class <b>BraidGroup</b> (p. 140) (defines a representation of a<br>Braid Group)// ) . . . . .   | 140 |
| <b>BraidNode</b> (Defines a crossing in a linked braid structure// ) . . . . .   | 142 |
| <b>BSets</b> (Implements construction of the initial commuting sets of subgroup<br>generators ) . . . . .  | 147 |
| <b>BalancedTree&lt; Obj &gt;::BTNode</b> . . . . .   | 149 |
| <b>CImage</b> . . . . .  | 152 |

|  |     |
|--|-----|
| <b>CLUTImage</b>   | 159 |
| <b>PC::Col</b>   | 162 |
| <b>PC::PowerCircuitCompMatrix::ColHdr</b>  | 164 |
| <b>CommDivider</b>   | 165 |
| <b>compMaps</b> (Implements a comparison operator of two maps )  | 167 |
| <b>ConfigFile</b> (Implements a mechanism for passing parameters from a configuration file )   | 168 |
| <b>ConstWordIterator</b>   | 172 |
| <b>CutVertices</b> (Implements an algorithm to find all articulation points of a graph// )   | 176 |
| <b>DCBraidReduction</b>  | 181 |
| <b>DDL</b>   | 183 |
| <b>DDLNode</b>   | 185 |
| <b>DehornoyForm</b> (Dehornoy Form of a braid word (aka/ handle free form) )   | 187 |
| <b>dump&lt; T &gt;</b>   | 190 |
| <b>Dump</b>  | 192 |
| <b>EditingDistance</b> (Implements the Editing ( <b>Levenstein</b> (p. 303)) distance between two words )  | 193 |
| <b>Equation</b>  | 195 |
| <b>FiniteAlphabet</b> (Implements a finite size alphabet )   | 199 |
| <b>FoldDetails</b>   | 200 |
| <b>Graphs::FoldDetails&lt; VertexType, EdgeType &gt;</b>   | 202 |
| <b>FPGGroup</b> (Class <b>FPGGroup</b> (p. 204) (finitely presented group) )   | 204 |
| <b>FRDFIT</b>  | 209 |
| <b>FreeGroup</b>   | 211 |
| <b>FreeMetabelianGroupAlgorithms</b> (Static class <b>FreeMetabelianGroupAlgorithms</b> (p. 214) encapsulates algorithms for FreeMetabelian-Group groups ) | 214 |
| <b>FSA</b>   | 216 |
| <b>FSAEdge</b>   | 222 |
| <b>FSARep</b>  | 224 |
| <b>FSASState</b>   | 231 |
| <b>Graph</b>   | 233 |
| <b>Graphs::GraphConcept&lt; VertexType, EdgeType &gt;</b> (The main class for graph types )  | 235 |
| <b>Graphs::GraphConceptRep&lt; VertexType, EdgeType &gt;</b> (Representation class for graph types )   | 240 |
| <b>GraphDrawingAttributes</b>  | 246 |
| <b>GraphEdge</b> (Defines non-labelled edge of the directed graph )  | 251 |
| <b>GraphRep</b>  | 255 |
| <b>GraphState</b>  | 258 |
| <b>GraphVertex&lt; EdgeType &gt;</b> ( <b>Graph</b> (p. 233) vertex )  | 260 |
| <b>GRIImage</b>  | 262 |
| <b>GRLUTImage</b>  | 266 |
| <b>HammingDistance</b> (Implements the Hamming distance between two words )  | 268 |

|  |     |
|--|-----|
| <b>HammingDistanceCyclic</b> (Implements the Hamming distance between two cyclic words ) . . . . .   | 270 |
| <b>InfiniteAlphabet</b> (Implements an infinite size alphabet ) . . . . .  | 272 |
| <b>PC::SignMatrix&lt; RowHdr, ColHdr, INTERNAL_TYPE &gt;::intColHdr</b> . . . . .  | 275 |
| <b>IntLabeledEdge</b> (Defines labelled (by integer) edge of the directed graph ) . . . . .  | 277 |
| <b>PC::PowerCircuitGraph::IntMarking</b> . . . . .   | 280 |
| <b>PC::PowerCircuitCompMatrix::IntMarking</b> . . . . .  | 281 |
| <b>PC::PowerCircuitCompMatrix::IntNode</b> . . . . .   | 283 |
| <b>PC::PowerCircuitGraph::IntNode</b> . . . . .  | 284 |
| <b>PC::SignMatrix&lt; RowHdr, ColHdr, INTERNAL_TYPE &gt;::intRowHdr</b> . . . . .  | 285 |
| <b>KLProtocolInstance</b> . . . . .  | 287 |
| <b>ICommDivider</b> . . . . .  | 291 |
| <b>LengthAttack_A1</b> (Implements Length-based attack on AAG protocol ) . . . . .   | 292 |
| <b>LengthAttack_A2</b> (Implements Length-based attack on AAG protocol ) . . . . .   | 295 |
| <b>LengthAttack_A3</b> (Implements Length-based attack on AAG protocol ) . . . . .   | 298 |
| <b>LengthAttackBase</b> (Basis interface for classes implementing the Length-Based attack ) . . . . .                                      | 301 |
| <b>Levenstein</b> (A class for computing the <b>Levenstein</b> (p. 303) distance between two words ) . . . . .                             | 303 |
| <b>LinkedBraidStructure</b> . . . . .  | 305 |
| <b>LinkedBraidStructureTransform</b> . . . . .   | 312 |
| <b>LookUpTable</b> . . . . .   | 315 |
| <b>ltstr</b> (Implements a comparison operator on two strings ) . . . . .  | 317 |
| <b>LUTImage</b> . . . . .  | 318 |
| <b>Map</b> (Defines a representation of a map between two sets of words ) . . . . .  | 320 |
| <b>__gnu_cxx::map_hash</b> . . . . .   | 328 |
| <b>PC::Marking</b> . . . . .   | 329 |
| <b>MatrixFp</b> . . . . .  | 333 |
| <b>MaxCommutePartition</b> . . . . .   | 336 |
| <b>MotivePattern</b> . . . . .   | 338 |
| <b>MotivePatternWrapper</b> . . . . .  | 339 |
| <b>NielsenAutoSet</b> (Implements a set of Nielsen automorphisms of a free group ) . . . . .   | 341 |
| <b>PC::Node</b> . . . . .  | 344 |
| <b>PC::PowerCircuitGraph::NodeUsedByType</b> . . . . .   | 346 |
| <b>ObjectOf&lt; Rep &gt;</b> . . . . .   | 348 |
| <b>PairDistanceSimilarityTest</b> (Implements a hypothesis testing that two words are similar according to some given criteria ) . . . . . | 351 |
| <b>PairGenerator</b> (Abstract interface for classes defining sword pairs similarities ) . . . . .   | 353 |
| <b>Parser</b> . . . . .  | 355 |
| <b>Partition</b> . . . . .   | 357 |
| <b>PBar</b> . . . . .  | 358 |
| <b>PDFPage</b> (Class implements a page of a pdf document ) . . . . .  | 360 |
| <b>PDFPageObject</b> (Implements interface for PDF drawing objects ) . . . . .   | 365 |
| <b>PDFPageObjectLine</b> (Class for pdf line object ) . . . . .  | 366 |
| <b>PDFStructure</b> (Implements a pdf document consisting of several pages ) . . . . .   | 368 |

|  |     |
|--|-----|
| <b>Permutation</b> ( <b>Permutation</b> (p. 371) ) . . . . .   | 371 |
| <b>PermutationEnumerator</b> . . . . .   | 382 |
| <b>Perturbation</b> . . . . .  | 387 |
| <b>PlanarGraphEdge</b> (Defines non-labelled edge of the directed planar graph ) . . . . .   | 388 |
| <b>PlanarGraphIntLabelledEdge</b> (Defines non-labelled edge of the directed planar graph ) . . . . .  | 392 |
| <b>PC::PowerCircuit</b> . . . . .  | 395 |
| <b>PC::PowerCircuitCompMatrix</b> . . . . .  | 404 |
| <b>PC::PowerCircuitGraph</b> . . . . .   | 418 |
| <b>PowerWord</b> . . . . .   | 431 |
| <b>PowerWordRep</b> . . . . .  | 440 |
| <b>StraightLineProgramWord::Production</b> (A production for a rule of a composition system ) . . . . .  | 446 |
| <b>QuadEquationTransformationGraph</b> . . . . .   | 448 |
| <b>quadruple</b> < <b>T1</b> , <b>T2</b> , <b>T3</b> , <b>T4</b> > . . . . .   | 452 |
| <b>quintuple</b> < <b>T1</b> , <b>T2</b> , <b>T3</b> , <b>T4</b> , <b>T5</b> > . . . . .   | 455 |
| <b>RandLib</b> (Static wrapper class for RANDLIB Library ) . . . . .   | 458 |
| <b>RandLibURG</b> (A wrapper class for RANDLIB Library ) . . . . .   | 460 |
| <b>RandomPairGenerator</b> (Implements a class for generating pseudo-random pairs of words ) . . . . .   | 463 |
| <b>RefCounter</b> . . . . .  | 466 |
| <b>RestrictedWhiteheadAutoSet</b> (Implements a so-called restricted set of Whitehead automorphisms of a free group ) . . . . .                                      | 469 |
| <b>RGB</b> . . . . .   | 472 |
| <b>PC::Row</b> . . . . .   | 474 |
| <b>PC::PowerCircuitCompMatrix::RowHdr</b> . . . . .  | 476 |
| <b>ShftConjKeyInstance</b> . . . . .   | 478 |
| <b>ShftConjKeyInstanceGarside</b> (Container for public and private information in Dehornoy's authentication protocol ) . . . . .                                    | 481 |
| <b>PC::SignMatrix</b> < <b>RowHdr</b> , <b>ColHdr</b> , <b>INTERNAL_TYPE</b> > . . . . .   | 484 |
| <b>StraightLineProgramWord</b> (Class <b>StraightLineProgramWord</b> (p. 507) ) . . . . .  | 507 |
| <b>StringScrambler</b> . . . . .   | 519 |
| <b>StringSimilarityMeasure</b> (Abstract interface for a class implementing string ( <b>Word</b> (p. 642)) similarity measure ) . . . . .                            | 520 |
| <b>SubgroupFG</b> . . . . .  | 522 |
| <b>SubwordEditingDistanceCyclic</b> (Implements the Editing distance between two cyclic words ) . . . . .  | 529 |
| <b>SubwordHammingDistanceCyclic</b> (Implements the Hamming distance between two cyclic words ) . . . . .  | 531 |
| <b>SubwordScrambler</b> . . . . .  | 533 |
| <b>TheGrigorchukGroupAlgorithms</b> (Static class <b>TheGrigorchukGroupAlgorithms</b> (p. 535) encapsulates algorithms for the original Grigorchuk group ) . . . . . | 535 |
| <b>ThLeftNormalForm</b> (Defines a representation of a left Garside normal form ) . . . . .  | 541 |

|  |     |
|--|-----|
| <b>ThompsonGroupFNormalForm</b> (Class <b>ThompsonGroupFNormalForm</b> (p. 556) (defines a representation of a normal form of an element of the Thompson's group F (infinitely generated))// ) . . . . . | 556 |
| <b>ThRightNormalForm</b> (Defines a representation of a right Garside normal form ) . . . . .  | 560 |
| <b>Permutation::triple</b> . . . . .   | 576 |
| <b>triple&lt; T1, T2, T3 &gt;</b> . . . . .  | 577 |
| <b>TripleDecompositionProtocolInstance</b> (Definition of the class <b>TripleDecompositionProtocolInstance</b> (p. 580) ) . . . . .  | 580 |
| <b>TTP_Conf</b> (Set of parameters required to construct the protocol instance ) . . . . .   | 586 |
| <b>TTPAttack</b> (This is an implementation of an attack on TTP algorithm for generating public sets of generators of the Algebraic Eraser protocol ) . . . . .  | 588 |
| <b>TTPLBA</b> . . . . .  | 592 |
| <b>TTPTuple</b> (Implements tuples corresponding to the putput of TTP algorithm ) . . . . .  | 594 |
| <b>udPDFPageObjectCircle</b> (Class implements a square pdf object ) . . . . .   | 598 |
| <b>udPDFPageObjectHorizLine</b> (Class implements a horizontal line pdf object ) . . . . .   | 600 |
| <b>udPDFPageObjectLine</b> (Class implements pdf line object ) . . . . .   | 602 |
| <b>udPDFPageObjectRect</b> (Class implements a rectangle pdf object ) . . . . .  | 605 |
| <b>udPDFPageObjectSquare</b> (Class implements a square pdf object ) . . . . .   | 608 |
| <b>udPDFPageObjectText</b> (Class for pdf text object ) . . . . .  | 610 |
| <b>udPDFPageObjectVertLine</b> (Class implements a vertical line pdf object ) . . . . .  | 612 |
| <b>UniformPartition</b> . . . . .  | 614 |
| <b>UniformScrambler</b> . . . . .  | 615 |
| <b>Value</b> (Implements data types of parameters obtained from a configuration file ) . . . . .   | 617 |
| <b>VectorEnumerator</b> (Class <b>VectorEnumerator</b> (p. 620) ) . . . . .  | 620 |
| <b>WhiteheadAutoSetType2</b> (Implements the set of Whitehead automorphisms of type II ) . . . . .   | 625 |
| <b>WhiteheadGraph</b> (Interface class for Whitehead graphs ) . . . . .  | 628 |
| <b>WhiteheadMinimization</b> (Implements a greedy procedure of reducing a word to its minimal length ) . . . . .   | 631 |
| <b>WhiteheadMultiGraph</b> (Whitehead Multi-Graph (Not implemented yet) ) . . . . .  | 634 |
| <b>WhiteheadSimpleGraph</b> (Implements Whitehead Simple graph (i.e. no multiple edges or loops allowed) ) . . . . .   | 635 |
| <b>Word</b> (Class <b>Word</b> (p. 642) (defines a representation of a <b>Word</b> (p. 642) over a group alphabet)// ) . . . . .   | 642 |
| <b>WordDraw</b> (CREATES A PPM image of a table for braid word ) . . . . .   | 658 |
| <b>WordIterator</b> . . . . .  | 662 |
| <b>WordMultiplyScrambler</b> . . . . .   | 666 |
| <b>WordPairComparison</b> (Implements a probabilistic measure for comparing two words ) . . . . .  | 668 |
| <b>WordRep</b> . . . . .   | 671 |
| <b>YYSTYPE</b> . . . . .   | 679 |





## Chapter 6

# File Index

### 6.1 File List

Here is a list of all files with brief descriptions:

|   |     |
|---|-----|
| Alphabet/include/ <b>Alphabet.h</b>                           | 681 |
| Alphabet/include/ <b>AlphabetBisonGrammar.h</b>               | 683 |
| Alphabet/include/ <b>Parser.h</b>                             | 685 |
| Alphabet/include/ <b>WordBisonGrammar.h</b>                   | 686 |
| BraidGroup/include/ <b>BKLLeftNormalForm.h</b>                | 687 |
| BraidGroup/include/ <b>BKLRightNormalForm.h</b>               | 688 |
| BraidGroup/include/ <b>BraidGroup.h</b>                       | 689 |
| BraidGroup/include/ <b>DehornoyForm.h</b>                     | 690 |
| BraidGroup/include/ <b>DehornoyForm_old.h</b>                 | 691 |
| BraidGroup/include/ <b>LinkedBraidStructure.h</b>             | 692 |
| BraidGroup/include/ <b>LinkedBraidStructure_old.h</b>         | 693 |
| BraidGroup/include/ <b>ShortBraidForm.h</b>                   | 694 |
| BraidGroup/include/ <b>ShortBraidForm_old.h</b>               | 695 |
| BraidGroup/include/ <b>ThLeftNormalForm.h</b>                 | 696 |
| BraidGroup/include/ <b>ThRightNormalForm.h</b>                | 697 |
| BraidGroup/include/ <b>ThRightNormalFormAlgorithms.h</b>      | 698 |
| CryptoAAG/include/ <b>AAGChallengeGeneration.h</b>            | 699 |
| CryptoAAG/include/ <b>AAGKeyGeneration.h</b>                  | 700 |
| CryptoAAG/include/ <b>LengthAttack.h</b>                      | 701 |
| CryptoAAG/include/ <b>MajorDump.h</b>                         | 703 |
| CryptoAE/include/ <b>AEProtocol.h</b>                         | 704 |
| CryptoAE/include/ <b>TTPAttack.h</b>                          | 705 |
| CryptoKL/include/ <b>KLKeyGeneration.h</b>                    | 706 |
| CryptoShftConj/include/ <b>ShftConjKeyGeneration.h</b>        | 707 |
| CryptoShftConj/include/ <b>ShftConjKeyGenerationGarside.h</b> | 708 |

|  |     |
|--|-----|
| CryptoTripleDecomposition/include/ <b>TripleDecompositionKeyGeneration.h</b> |     |
| 709  |     |
| Elt/include/ <b>PowerWord.h</b>  | 710 |
| Elt/include/ <b>PowerWordIterator.h</b>                                      | 711 |
| Elt/include/ <b>PowerWordRep.h</b>   | 712 |
| Elt/include/ <b>Word.h</b>   | 713 |
| Elt/include/ <b>WordIterator.h</b>   | 714 |
| Elt/include/ <b>WordRep.h</b>  | 715 |
| Equation/include/ <b>Equation.h</b>  | 716 |
| Equation/include/ <b>QuadEquatTransformationGraph.h</b>                      | 717 |
| Experiments/include/ <b>AAGKeyPairGenerator.h</b>                            | 718 |
| Experiments/include/ <b>DCBraidReduction.h</b>                               | 719 |
| Experiments/include/ <b>DDL.h</b>  | 720 |
| Experiments/include/ <b>FRDFIT.h</b>   | 721 |
| FreeGroup/include/ <b>FreeGroup.h</b>  | 722 |
| FreeGroup/include/ <b>StraightLineProgramWord.h</b>                          | 723 |
| FreeGroup/include/ <b>WhiteheadGraph.h</b>                                   | 724 |
| FreeMetabelianGroup/include/ <b>FreeMetabelianGroupAlgorithms.h</b>          | 726 |
| general/include/ <b>BalancedTree.h</b>                                       | 727 |
| general/include/ <b>ConfigFile.h</b>   | 728 |
| general/include/ <b>dump.h</b>   | 729 |
| general/include/ <b>errmsgs.h</b>  | 730 |
| general/include/ <b>FormatOutput.h</b>                                       | 731 |
| general/include/ <b>ObjectOf.h</b>   | 732 |
| general/include/ <b>Permutation.h</b>  | 733 |
| general/include/ <b>PermutationEnumerator.h</b>                              | 734 |
| general/include/ <b>ProgressBar.h</b>  | 735 |
| general/include/ <b>RefCounter.h</b>   | 736 |
| general/include/ <b>tuples.h</b>   | 737 |
| general/include/ <b>VectorEnumerator.h</b>                                   | 738 |
| Graph/include/ <b>FSA.h</b>  | 739 |
| Graph/include/ <b>FSARep.h</b>   | 740 |
| Graph/include/ <b>Graph.h</b>  | 741 |
| Graph/include/ <b>GraphAlgorithms.h</b>                                      | 743 |
| Graph/include/ <b>GraphConcept.h</b>   | 745 |
| Graph/include/ <b>GraphConceptAlgorithms.h</b>                               | 746 |
| Graph/include/ <b>GraphDrawingAttributes.h</b>                               | 748 |
| Graph/include/ <b>GraphRep.h</b>   | 749 |
| Graph/include/ <b>GraphType.h</b>  | 750 |
| Graph/include/ <b>RandomFSA.h</b>  | 754 |
| Graphics/include/ <b>AImage.h</b>  | 755 |
| Graphics/include/ <b>PDFgraphing.h</b>                                       | 757 |
| Graphics/include/ <b>WordDraw.h</b>  | 758 |
| Group/include/ <b>AdvDehnAlgorithm.h</b>                                     | 759 |
| Group/include/ <b>FPGroup.h</b>  | 760 |
| HigmanGroup/include/ <b>BaumslagGersten.h</b>                                | 761 |

|   |     |
|---|-----|
| HigmanGroup/include/ <b>PowerCircuit.h</b> . . . . .                        | 762 |
| HigmanGroup/include/ <b>PowerCircuitCompMatrix.h</b> . . . . .              | 763 |
| HigmanGroup/include/ <b>PowerCircuitGraph.h</b> . . . . .                   | 764 |
| HigmanGroup/include/ <b>Sign.h</b> . . . . .                                | 765 |
| HigmanGroup/include/ <b>SignMatrix.h</b> . . . . .                          | 766 |
| Maps/include/ <b>Map.h</b> . . . . .  | 767 |
| Maps/include/ <b>WhiteheadAutoSet.h</b> . . . . .                           | 768 |
| ranlib/include/ <b>cdflib.h</b> . . . . .                                   | 770 |
| ranlib/include/ <b>ranlib.h</b> . . . . .                                   | 775 |
| ranlib/include/ <b>RanlibCPP.h</b> . . . . .                                | 778 |
| SbgpFG/include/ <b>SubgroupFG.h</b> . . . . .                               | 779 |
| SbgpFG/include/ <b>WhiteheadGraph.h</b> . . . . .                           | 725 |
| StringSimilarity/include/ <b>Levenstein.h</b> . . . . .                     | 780 |
| StringSimilarity/include/ <b>MotivePatternWrapper.h</b> . . . . .           | 781 |
| StringSimilarity/include/ <b>PairDistanceTest.h</b> . . . . .               | 782 |
| StringSimilarity/include/ <b>SimilarityMeasures.h</b> . . . . .             | 783 |
| StringSimilarity/include/ <b>StringScramblers.h</b> . . . . .               | 785 |
| TheGrigorchukGroup/include/ <b>TheGrigorchukGroupAlgorithms.h</b> . . . . . | 786 |
| ThompsonGroup/include/ <b>ThompsonGroupFNormalForm.h</b> . . . . .          | 787 |



## **Chapter 7**

# **Directory Documentation**

### **7.1 Alphabet/ Directory Reference**

#### **Directories**

- directory **include**

## 7.2 BraidGroup/ Directory Reference

### Directories

- directory **include**

## 7.3 CryptoAAG/ Directory Reference

### Directories

- directory **include**

## 7.4 CryptoAE/ Directory Reference

### Directories

- directory **include**



## 7.5 CryptoKL/ Directory Reference

### Directories

- directory **include**

## 7.6 CryptoShftConj/ Directory Reference

### Directories

- directory **include**

## 7.7 CryptoTripleDecomposition/ Directory Reference

### Directories

- directory **include**

## 7.8 Elt/ Directory Reference

### Directories

- directory **include**

## 7.9 Equation/ Directory Reference

### Directories

- directory **include**

## 7.10 Experiments/ Directory Reference

### Directories

- directory **include**

## 7.11 FreeGroup/ Directory Reference

### Directories

- directory **include**

## 7.12 FreeMetabelianGroup/ Directory Reference

### Directories

- directory **include**



## 7.13 general/ Directory Reference

### Directories

- directory **include**

## 7.14 Graph/ Directory Reference

### Directories

- directory **include**

## 7.15 Graphics/ Directory Reference

### Directories

- directory **include**

## 7.16 Group/ Directory Reference

### Directories

- directory **include**

## 7.17 HigmanGroup/ Directory Reference

### Directories

- directory **include**

## 7.18 ThompsonGroup/include/ Directory Reference

### Files

- file `ThompsonGroupFNormalForm.h`

## 7.19 TheGrigorchukGroup/include/ Directory Reference

### Files

- file TheGrigorchukGroupAlgorithms.h

## 7.20 StringSimilarity/include/ Directory Reference

### Files

- file **Levenstein.h**
- file **MotivePatternWrapper.h**
- file **PairDistanceTest.h**
- file **SimilarityMeasures.h**
- file **StringScramblers.h**



## 7.21 ranlib/include/ Directory Reference

### Files

- file **cdflib.h**
- file **ranlib.h**
- file **RanlibCPP.h**

## 7.22 Maps/include/ Directory Reference

### Files

- file **Map.h**
- file **WhiteheadAutoSet.h**

## 7.23 HigmanGroup/include/ Directory Reference

### Files

- file **BaumslagGersten.h**
- file **PowerCircuit.h**
- file **PowerCircuitCompMatrix.h**
- file **PowerCircuitGraph.h**
- file **Sign.h**
- file **SignMatrix.h**

## 7.24 Group/include/ Directory Reference

### Files

- file **AdvDehnAlgorithm.h**
- file **FPGroup.h**

## 7.25 Graphics/include/ Directory Reference

### Files

- file **AImage.h**
- file **PDFgraphing.h**
- file **WordDraw.h**

## 7.26 Graph/include/ Directory Reference

### Files

- file **FSA.h**
- file **FSARep.h**
- file **Graph.h**
- file **GraphAlgorithms.h**
- file **GraphConcept.h**
- file **GraphConceptAlgorithms.h**
- file **GraphDrawingAttributes.h**
- file **GraphRep.h**
- file **GraphType.h**
- file **RandomFSA.h**

## 7.27 general/include/ Directory Reference

### Files

- file **BalancedTree.h**
- file **ConfigFile.h**
- file **dump.h**
- file **errmsgs.h**
- file **FormatOutput.h**
- file **ObjectOf.h**
- file **Permutation.h**
- file **PermutationEnumerator.h**
- file **ProgressBar.h**
- file **RefCounter.h**
- file **tuples.h**
- file **VectorEnumerator.h**

## 7.28 FreeMetabelianGroup/include/ Directory Reference

### Files

- file `FreeMetabelianGroupAlgorithms.h`



## 7.29 SbgpFG/include/ Directory Reference

### Files

- file **SubgroupFG.h**
- file **WhiteheadGraph.h**

## 7.30 FreeGroup/include/ Directory Reference

### Files

- file **FreeGroup.h**
- file **StraightLineProgramWord.h**
- file **WhiteheadGraph.h**

## 7.31 Experiments/include/ Directory Reference

### Files

- file **AAGKeyPairGenerator.h**
- file **DCBraidReduction.h**
- file **DDL.h**
- file **FRDFIT.h**

## 7.32 Equation/include/ Directory Reference

### Files

- file **Equation.h**
- file **QuadEquatTransformationGraph.h**

## 7.33 Elt/include/ Directory Reference

### Files

- file **PowerWord.h**
- file **PowerWordIterator.h**
- file **PowerWordRep.h**
- file **Word.h**
- file **WordIterator.h**
- file **WordRep.h**

## 7.34 CryptoTripleDecomposition/include/ Directory Reference

### Files

- file TripleDecompositionKeyGeneration.h

## 7.35 CryptoShftConj/include/ Directory Reference

### Files

- file **ShftConjKeyGeneration.h**
- file **ShftConjKeyGenerationGarside.h**

## 7.36 CryptoKL/include/ Directory Reference

### Files

- file **KLKeyGeneration.h**



## 7.37 CryptoAE/include/ Directory Reference

### Files

- file **AEProtocol.h**
- file **TTPAttack.h**

## 7.38 CryptoAAG/include/ Directory Reference

### Files

- file **AAGChallengeGeneration.h**
- file **AAGKeyGeneration.h**
- file **LengthAttack.h**
- file **MajorDump.h**

## 7.39 BraidGroup/include/ Directory Reference

### Files

- file **BKLLeftNormalForm.h**
- file **BKLRightNormalForm.h**
- file **BraidGroup.h**
- file **DehornoyForm.h**
- file **DehornoyForm\_old.h**
- file **LinkedBraidStructure.h**
- file **LinkedBraidStructure\_old.h**
- file **ShortBraidForm.h**
- file **ShortBraidForm\_old.h**
- file **ThLeftNormalForm.h**
- file **ThRightNormalForm.h**
- file **ThRightNormalFormAlgorithms.h**

## 7.40 Alphabet/include/ Directory Reference

### Files

- file **Alphabet.h**
- file **AlphabetBisonGrammar.h**
- file **Parser.h**
- file **WordBisonGrammar.h**

## 7.41 Maps/ Directory Reference

### Directories

- directory **include**

## 7.42 ranlib/ Directory Reference

### Directories

- directory **include**

## 7.43 SbgpFG/ Directory Reference

### Directories

- directory **include**

## 7.44 StringSimilarity/ Directory Reference

### Directories

- directory **include**



## 7.45 TheGrigorchukGroup/ Directory Reference

### Directories

- directory **include**

## 7.46 ThompsonGroup/ Directory Reference

### Directories

- directory **include**

## Chapter 8

# Namespace Documentation

### 8.1 `__gnu_cxx` Namespace Reference

#### Classes

- struct `map_hash`

## 8.2 AAGChallenge Namespace Reference

### Functions

- `vector< Word > getDpSubgroup (int N, int Pgens, const vector< Word > &Prelts, const pair< vector< Word >, vector< Word > > &genComps, int conjLen)`
- `Word getDelta (int i, int c)`
- `pair< vector< Word >, vector< Word > > getSgGenComponentsRandom (int Pgens, int c, int k, int len)`
- `pair< vector< Word >, vector< Word > > getSgGenComponentsSquares (int Pgens, int c, int k)`
- `Word randomSubgroupWord (int N, const vector< Word > &sg)`
- `Word specialSubgroupWord (const Word &a, const Word &t, int n)`
- `Word specialSubgroupWordRandom (const Word &a, const Word &t, int len)`
- `vector< Word > generateSubgroup (int c, int k, int comp_len)`
- `Word generateKeyDecomp (int n)`

## 8.2.1 Function Documentation

- 8.2.1.1 Word AAGChallenge::generateKeyDecomp (int *n*)
- 8.2.1.2 vector<Word> AAGChallenge::generateSubgroup (int *c*, int *k*, int *comp\_len*)
- 8.2.1.3 Word AAGChallenge::getDelta (int *i*, int *c*)
- 8.2.1.4 vector<Word> AAGChallenge::getDpSubgroup (int *N*, int *Pgens*, const vector< Word > & *Prelts*, const pair< vector< Word >, vector< Word > > & *genComps*, int *conjLen*)
- 8.2.1.5 pair< vector<Word>,vector<Word> > AAGChallenge::getSgGenComponentsRandom (int *Pgens*, int *c*, int *k*, int *len*)
- 8.2.1.6 pair< vector<Word>,vector<Word> > AAGChallenge::getSgGenComponentsSquares (int *Pgens*, int *c*, int *k*)
- 8.2.1.7 Word AAGChallenge::randomSubgroupWord (int *N*, const vector< Word > & *sg*)
- 8.2.1.8 Word AAGChallenge::specialSubgroupWord (const Word & *a*, const Word & *t*, int *n*)
- 8.2.1.9 Word AAGChallenge::specialSubgroupWordRandom (const Word & *a*, const Word & *t*, int *len*)

## 8.3 BG Namespace Reference

### Classes

- struct **BGMonomial**

### Functions

- bool **solveWPinBG** (**PowerCircuit** \*pc, std::list< **BGMonomial** > &input)
- bool **solveWPinBG** (**PowerCircuit** \*pc, std::string input)

#### 8.3.1 Function Documentation

##### 8.3.1.1 bool BG::solveWPinBG (**PowerCircuit** \* *pc*, std::string *input*)

The input is a string consisting of the letters a, b, t and their inverses A, B, T. An empty instance of PowerCircuit is needed, too.

##### 8.3.1.2 bool BG::solveWPinBG (**PowerCircuit** \* *pc*, std::list< **BGMonomial** > & *input*)

## 8.4 Graphs Namespace Reference

### Classes

- class **GraphConceptRep**  
*Representation class for graph types.*
- class **GraphConcept**  
*The main class for graph types.*
- struct **FoldDetails**

### Functions

- template<class VertexType , class EdgeType >  
ostream & **operator**<< (ostream &os, const **GraphConcept**< VertexType, EdgeType > &G)  
*Output the graph.*
- template<class Graph >  
map< int, typename **Graph::edge\_type** > **getGeodesicTree\_in** (const **Graph** &graph, int init\_v)  
*Compute directed geodesic tree "inward" to the vertex init\_v.*
- template<class Graph >  
map< int, typename **Graph::edge\_type** > **getGeodesicTree\_out** (const **Graph** &graph, int init\_v)  
*Compute geodesic directed tree starting the vertex init\_v.*
- template<class Graph >  
map< int, int > **getDistances\_out** (const **Graph** &graph, int init\_v)  
*Compute distances from the vertex init\_v.*
- template<class Graph >  
map< int, int > **getDistances\_in** (const **Graph** &graph, int init\_v)  
*Compute distances to the vertex init\_v.*
- template<class edge\_type >  
pair< bool, list< edge\_type > > **readoffGeodesicTree** (const map< int, edge\_type > &tree, int v)  
*Function finds a path in the tree starting from the state st\_num to the root.*

- `template<class LabelledGraph , class ConstIterator >`  
`int trace (const LabelledGraph &LG, int init_v, ConstIterator B, ConstIterator E)`
- `template<class LabelledGraph , class ConstIterator >`  
`pair< bool, list< typename LabelledGraph::edge_type > > trace_path (const LabelledGraph &LG, int init_v, ConstIterator B, ConstIterator E)`
- `template<class LabelledGraph >`  
`void fold (LabelledGraph &G, set< int > candidates, list< FoldDetails< typename LabelledGraph::vertex_type, typename LabelledGraph::edge_type > > *details=NULL)`

*Fold a labelled graph. To apply this function the graph edges must have theLabel defined.*

- `template<class LabelledGraph >`  
`void fold (const LabelledGraph &G, int candidate, list< FoldDetails< typename LabelledGraph::vertex_type, typename LabelledGraph::edge_type > > *details=NULL)`
- `template<class LabelledGraph >`  
`void fold (const LabelledGraph &G, list< FoldDetails< typename LabelledGraph::vertex_type, typename LabelledGraph::edge_type > > *details=NULL)`
- `template<class LabelledGraph , class FoldDetailsConstIterator >`  
`void unfold (LabelledGraph &G, FoldDetailsConstIterator B, FoldDetailsConstIterator E)`

*Revert the folding.*

- `template<class LabelledGraph , class FoldDetailsConstIterator >`  
`void liftup (const LabelledGraph &graph, int init_vertex, list< typename LabelledGraph::edge_type > &path, FoldDetailsConstIterator B, FoldDetailsConstIterator E)`

*Revert the folding.*

- `template<class edge_type >`  
`void reduce_path (int init_vertex, list< edge_type > &path)`

*Reduce a path (remove all pairs of consequent inverse edges).*

### 8.4.1 Detailed Description

EdgeType must support: `e.inverse( v )`, where `v` is the number of the origin



## 8.4.2 Function Documentation

**8.4.2.1** `template<class LabelledGraph > void Graphs::fold  
(const LabelledGraph & G, list< FoldDetails< typename  
LabelledGraph::vertex_type, typename LabelledGraph::edge_type > >  
* details = NULL) [inline]`

Definition at line 457 of file GraphConceptAlgorithms.h.

References fold().

**8.4.2.2** `template<class LabelledGraph > void Graphs::fold (const  
LabelledGraph & G, int candidate, list< FoldDetails< typename  
LabelledGraph::vertex_type, typename LabelledGraph::edge_type > >  
* details = NULL) [inline]`

Definition at line 448 of file GraphConceptAlgorithms.h.

References fold().

**8.4.2.3** `template<class LabelledGraph > void Graphs::fold (LabelledGraph  
& G, set< int > candidates, list< FoldDetails< typename  
LabelledGraph::vertex_type, typename LabelledGraph::edge_type > >  
* details = NULL) [inline]`

Fold a labelled graph. To apply this function the graph edges must have theLabel defined. The graph is not folded if there is a vertex and two different edges leaving it equally labelled. In that event "fold" pinches end-points of these edges.

Definition at line 404 of file GraphConceptAlgorithms.h.

Referenced by fold().

**8.4.2.4** `template<class Graph > map< int , int > Graphs::getDistances_in  
(const Graph & graph, int init_v) [inline]`

Compute distances to the vertex init\_v.

Definition at line 213 of file GraphConceptAlgorithms.h.

**8.4.2.5** `template<class Graph > map< int , int > Graphs::getDistances_out  
(const Graph & graph, int init_v) [inline]`

Compute distances from the vertex init\_v.

Definition at line 155 of file GraphConceptAlgorithms.h.

**8.4.2.6** `template<class Graph > map< int , typename Graph::edge_type  
> Graphs::getGeodesicTree_in (const Graph & graph, int init_v)  
[inline]`

Compute directed geodesic tree "inward" to the vertex *init\_v*. Returns a list of edges that form a a geodesic subtree of a graph directed to the vertex *init\_v*. If *init\_v* is reachable from *V* if and only if *result[V]* is defined.

Definition at line 33 of file *GraphConceptAlgorithms.h*.

**8.4.2.7** `template<class Graph > map< int , typename Graph::edge_type >  
Graphs::getGeodesicTree_out (const Graph & graph, int init_v)  
[inline]`

Compute geodesic directed tree starting the vertex *init\_v*. Returns a list of edges that form a a geodesic subtree of a graph directed to the vertex *init\_v*. If *init\_v* is reachable from *V* if and only if *result[V]* is defined.

Definition at line 95 of file *GraphConceptAlgorithms.h*.

**8.4.2.8** `template<class LabelledGraph , class FoldDetailsConstIterator  
> void Graphs::liftup (const LabelledGraph & graph, int  
init_vertex, list< typename LabelledGraph::edge_type > & path,  
FoldDetailsConstIterator B, FoldDetailsConstIterator E) [inline]`

Revert the folding.

Definition at line 498 of file *GraphConceptAlgorithms.h*.

References *reduce\_path()*.

**8.4.2.9** `template<class VertexType , class EdgeType > ostream&  
Graphs::operator<< (ostream & os, const GraphConcept<  
VertexType, EdgeType > & G) [inline]`

Output the graph. To use this function a function *ostream& operator << ( ostream& os , const EdgeType& E )* must be defined. Otherwise you will get an error when linking.

Definition at line 346 of file *GraphConcept.h*.

**8.4.2.10** `template<class edge_type > pair< bool , list< edge_type > >  
Graphs::readoffGeodesicTree (const map< int, edge_type > & tree,  
int v) [inline]`

Function finds a path in the tree starting from the state *st\_num* to the root.

Definition at line 271 of file GraphConceptAlgorithms.h.

**8.4.2.11** `template<class edge_type > void Graphs::reduce_path (int init_vertex,  
list< edge_type > &path) [inline]`

Reduce a path (remove all pairs of consequent inverse edges).

Definition at line 574 of file GraphConceptAlgorithms.h.

Referenced by `liftup()`.

**8.4.2.12** `template<class LabelledGraph , class ConstIterator > int  
Graphs::trace (const LabelledGraph & LG, int init_v, ConstIterator  
B, ConstIterator E) [inline]`

Definition at line 294 of file GraphConceptAlgorithms.h.

**8.4.2.13** `template<class LabelledGraph , class ConstIterator > pair< bool ,  
list< typename LabelledGraph::edge_type > > Graphs::trace_path  
(const LabelledGraph & LG, int init_v, ConstIterator B,  
ConstIterator E) [inline]`

Definition at line 333 of file GraphConceptAlgorithms.h.

**8.4.2.14** `template<class LabelledGraph , class FoldDetailsConstIterator > void  
Graphs::unfold (LabelledGraph & G, FoldDetailsConstIterator B,  
FoldDetailsConstIterator E) [inline]`

Revert the folding.

Definition at line 480 of file GraphConceptAlgorithms.h.

## 8.5 msgs Namespace Reference

Errors output handling.

### Functions

- void **error** (const char \*msg)  
*Print a message to the standard error output.*
- void **error** (const string &msg)  
*Print a message to the standard error output.*
- bool **error** (bool exp, const char \*msg)  
*Conditional output of an error message to the standard error output.*
- void **warn** (const char \*msg)  
*Print a message to the standard error output.*

### 8.5.1 Detailed Description

Errors output handling.

### 8.5.2 Function Documentation

#### 8.5.2.1 bool msgs::error (bool exp, const char \* msg) [inline]

Conditional output of an error message to the standard error output. Prints an error message if `exp` is not `true`. This function will go into infinite loop (freeze) so one can get a stack backtrace when debugging.

#### Parameters:

- exp* - boolean expression.  
*msg* - the error message.

Definition at line 49 of file `errormsgs.h`.

#### 8.5.2.2 void msgs::error (const string & msg) [inline]

Print a message to the standard error output. This function will go into infinite loop (freeze) so one can get a stack backtrace when debugging.

**Parameters:**

*msg* - the error message.

Definition at line 35 of file errormsgs.h.

**8.5.2.3 void msgs::error (const char \* *msg*) [inline]**

Print a message to the standard error output. This function will go into infinite loop (freeze) so one can get a stack backtrace when debugging.

**Parameters:**

*msg* - the error message.

Definition at line 23 of file errormsgs.h.

Referenced by Map::generatingImages(), WhiteheadSimpleGraph::getCount(), Map::Map(), and Map::setGeneratingImages().

**8.5.2.4 void msgs::warn (const char \* *msg*) [inline]**

Print a message to the standard error output.

**Parameters:**

*msg* - the error message.

Definition at line 65 of file errormsgs.h.

## 8.6 PC Namespace Reference

### Classes

- class **Node**
- class **Marking**
- class **PowerCircuit**
- class **PowerCircuitCompMatrix**
- class **PowerCircuitGraph**
- struct **Row**
- struct **Col**
- class **SignMatrix**

### Typedefs

- typedef char **Sign**

### Functions

- void **addSigns** (const **Sign** &op1, const **Sign** &op2, **Sign** &result, **Sign** &carry)
- int **compareSigns** (const **Sign** &op1, const **Sign** &op2)
- int **signToInt** (const **Sign** &s)
- **Sign** **negateSign** (const **Sign** &s)
- std::string **signToString** (const **Sign** &s)

### Variables

- const **Sign** **ZERO** = 0
- const **Sign** **PLUS** = 1
- const **Sign** **MINUS** = 2
- const **Row** **UNDEFINED\_ROW**
- const **Col** **UNDEFINED\_COL**

#### 8.6.1 Detailed Description

This package implements power circuits in two different ways. The first one uses compact representation of markings as defined in [1]. The tree containing all markings is stored not as a graph but as a matrix to improve access time. This implementation is asymptotically optimal.

The other implementation (**PowerCircuitGraph** (p.418)) stores the graph by adjacency lists and resembles the less technical approach chosen in [2]. Markings, too,

are stored as lists. This implementation does not make use of compact representations. Thus, it is not theoretically optimal, but usually faster for typical inputs as it uses a lot of overhead.

Both power-circuit-classes are derived from the abstract class **PowerCircuit** (p. 395) and can be used interchangeably.

To perform calculations with power circuits, create an instance of one of the above classes. The **PowerCircuit** (p. 395) class provides methods to create markings, perform reduction, insert edges into the graph (connect), and other tasks.

Markings work like references to the respective internal representations. Thus the assignment operator does not create a deep copy of a marking. However all arithmetic operations create new markings, so you can keep the old ones. It is also not necessary (and not possible) to delete markings (in the **PowerCircuitGraph** (p. 418) class once there is no marking pointing onto an internal marking, its memory is automatically deallocated). Just make sure that you create markings only for the time you really need them.

To get an idea of how to use power circuits, you might want to take a look at the examples included in this package, where power circuits are used to solve the word problem in certain groups.

In order to use the draw method you need graphviz installed on your computer (download from <http://www.graphviz.org/>). This method creates a postscript file with a graphical depiction of the graph.

The documentation is created with doxygen (see <http://www.stack.nl/~dimitri/doxygen/index.html>). To create the documentation in html and latex format use "doxygen". You can change the settings for the documentation in the file "Doxyfile". If "Doxyfile" does not exist, use "doxygen -g" to create it.

#### Authors:

Juern Laun  
Armin Weiss

A lot of the theoretical background of power circuits is due to Sasha Ushakov. We thank him for his helpful advice, and also for the invitation to the Stevens Institute where parts of this implementation were developed.

[1] V. Diekert, J. Laun, A. Ushakov. Efficient algorithms for highly compressed data: The **Word** (p. 642) Problem in Higman's Group is in P. Preprint, 2011. <http://arxiv.org/abs/1103.1232>

[2] V. Diekert, J. Laun, A. Ushakov. Efficient algorithms for highly compressed data: The **Word** (p. 642) Problem in Higman's Group is in P. Conference version, to appear, 2011.

## 8.6.2 Typedef Documentation

### 8.6.2.1 typedef char PC::Sign

Type for storing signs, i.e., members of the set  $\{-1, 0, +1\}$ .

Definition at line 19 of file Sign.h.

## 8.6.3 Function Documentation

**8.6.3.1** void PC::addSigns (const Sign & *op1*, const Sign & *op2*, Sign & *result*, Sign & *carry*)

**8.6.3.2** int PC::compareSigns (const Sign & *op1*, const Sign & *op2*)

**8.6.3.3** Sign PC::negateSign (const Sign & *s*)

**8.6.3.4** int PC::signToInt (const Sign & *s*)

**8.6.3.5** std::string PC::signToString (const Sign & *s*)

## 8.6.4 Variable Documentation

**8.6.4.1** const Sign PC::MINUS = 2

Definition at line 22 of file Sign.h.

Referenced by PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::printMatrix().

**8.6.4.2** const Sign PC::PLUS = 1

Definition at line 21 of file Sign.h.

Referenced by PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::printMatrix().

**8.6.4.3** const Col PC::UNDEFINED\_COL

Definition at line 27 of file SignMatrix.h.

**8.6.4.4** const Row PC::UNDEFINED\_ROW

Definition at line 26 of file SignMatrix.h.



**8.6.4.5   const Sign PC::ZERO = 0**

Definition at line 20 of file Sign.h.

## 8.7 RMap Namespace Reference

Namespace for random map generating methods.

### Functions

- **Map getRandomWhiteheadAuto** (int  $n$ )  
*Returns a randomly generated Whitehead automorphism.*
- **Map getRandomAuto** (int  $n$ , int  $l$ )  
*Returns a randomly generated automorphism of a given length.*

### 8.7.1 Detailed Description

Namespace for random map generating methods.

### 8.7.2 Function Documentation

#### 8.7.2.1 Map RMap::getRandomAuto (int $n$ , int $l$ )

Returns a randomly generated automorphism of a given length.

**Parameters:**

- $n$  - the rank of a free group
- $l$  - the number of Whitehead automorphisms

#### 8.7.2.2 Map RMap::getRandomWhiteheadAuto (int $n$ )

Returns a randomly generated Whitehead automorphism.

**Parameters:**

- $n$  - the rank of a free group

**Returns:**

- returns a randomly generated Whitehead automorphism

## 8.8 WhiteheadAutoSet Namespace Reference

### Functions

- **Word reduceBy** (const **Word** &*w*, const **SetOfMaps** &*theSet*)  
*Reduce the length of a given word by automorphisms form a set.*

### 8.8.1 Function Documentation

#### 8.8.1.1 Word **WhiteheadAutoSet::reduceBy** (const **Word** & *w*, const **SetOfMaps** & *theSet*)

Reduce the length of a given word by automorphisms form a set. Tries to reduce the length of a given word by applying automorphisms from a set. Will return if a shorter word is found

Note, it does not return the minimal length word.

#### Parameters:

- w* - the initial word to be reduced.
- theSet* - set of automorphisms to reduce by.

#### Returns:

- a shorter word if found, original word otherwise.

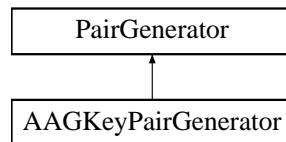


## Chapter 9

# Class Documentation

### 9.1 AAGKeyPairGenerator Class Reference

`#include <AAGKeyPairGenerator.h>`Inheritance diagram for AAGKeyPairGenerator::



#### Public Member Functions

- **AAGKeyPairGenerator** (int n, int min\_len, int max\_len, int n\_sg\_gens, int a\_decomp, int b\_decomp)
- pair< **Word**, **Word** > **getFalsePair** ()  
*Returns a pair of words that are not similar under specified criteria.*
- pair< **Word**, **Word** > **getTruePair** ()  
*Returns a pair of words similar under specified criteria.*

#### Private Attributes

- int **nGens**
- int **minLen**

- int **maxLen**
- int **nSgGens**
- int **AliceDecompositionLength**
- int **BobDecompositionLength**

### 9.1.1 Detailed Description

Definition at line 20 of file AAGKeyPairGenerator.h.

### 9.1.2 Constructor & Destructor Documentation

**9.1.2.1** `AAGKeyPairGenerator::AAGKeyPairGenerator (int n, int min_len, int max_len, int n_sg_gens, int a_decomp, int b_decomp)` `[inline]`

Definition at line 23 of file AAGKeyPairGenerator.h.

### 9.1.3 Member Function Documentation

**9.1.3.1** `pair<Word,Word> AAGKeyPairGenerator::getFalsePair ()`  
`[inline, virtual]`

Returns a pair of words that are not similar under specified criteria.

Implements **PairGenerator** (p. 353).

Definition at line 31 of file AAGKeyPairGenerator.h.

References `AliceDecompositionLength`, `BobDecompositionLength`, `AAGProtocolInstance::getAliceConjSbgp()`, `AAGProtocolInstance::getAlicePublicSbgp()`, `maxLen`, `minLen`, `nGens`, and `nSgGens`.

**9.1.3.2** `pair<Word,Word> AAGKeyPairGenerator::getTruePair ()`  
`[inline, virtual]`

Returns a pair of words similar under specified criteria.

Implements **PairGenerator** (p. 353).

Definition at line 61 of file AAGKeyPairGenerator.h.

References `AliceDecompositionLength`, `BobDecompositionLength`, `AAGProtocolInstance::getAliceConjSbgp()`, `AAGProtocolInstance::getAlicePublicSbgp()`, `maxLen`, `minLen`, `nGens`, and `nSgGens`.

## 9.1.4 Member Data Documentation

### 9.1.4.1 `int AAGKeyPairGenerator::AliceDecompositionLength` `[private]`

Definition at line 96 of file AAGKeyPairGenerator.h.

Referenced by `getFalsePair()`, and `getTruePair()`.

### 9.1.4.2 `int AAGKeyPairGenerator::BobDecompositionLength` `[private]`

Definition at line 97 of file AAGKeyPairGenerator.h.

Referenced by `getFalsePair()`, and `getTruePair()`.

### 9.1.4.3 `int AAGKeyPairGenerator::maxLen` `[private]`

Definition at line 94 of file AAGKeyPairGenerator.h.

Referenced by `getFalsePair()`, and `getTruePair()`.

### 9.1.4.4 `int AAGKeyPairGenerator::minLen` `[private]`

Definition at line 93 of file AAGKeyPairGenerator.h.

Referenced by `getFalsePair()`, and `getTruePair()`.

### 9.1.4.5 `int AAGKeyPairGenerator::nGens` `[private]`

Definition at line 92 of file AAGKeyPairGenerator.h.

Referenced by `getFalsePair()`, and `getTruePair()`.

### 9.1.4.6 `int AAGKeyPairGenerator::nSgGens` `[private]`

Definition at line 95 of file AAGKeyPairGenerator.h.

Referenced by `getFalsePair()`, and `getTruePair()`.

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

- `Experiments/include/AAGKeyPairGenerator.h`

## 9.2 AAGProtocolInstance Class Reference

```
#include <AAGKeyGeneration.h>
```

### Public Member Functions

- **AAGProtocolInstance** (int N, const vector< **Word** > &aSbgp, const vector< **Word** > &bSbgp, const **Word** &aDecomp, const **Word** &bDecomp, bool useForm=true)
- vector< **Word** > **getAlicePublicSbgp** () const
- vector< **Word** > **getBobPublicSbgp** () const
- vector< **Word** > **getAliceConjSbgp** () const
- vector< **Word** > **getBobConjSbgp** () const
- **Word** **getAliceKey** () const
- **Word** **getBobKey** () const
- **Word** **getSharedKey** () const
- void **printStats** (ostream &out)

### Static Public Member Functions

- static **AAGProtocolInstance** **random** (int N, int num\_gens, int min\_len, int max\_len, int AliceDecompositionLength, int BobDecompositionLength)
- static **AAGProtocolInstance** **challenge** (int N, int sg\_conj\_len, int c, int k, int key\_len)

### Static Private Member Functions

- static **Word** **generateHardProductOfGenerators** (int num\_gens, int product\_length)

*Function generates a hard key.*

### Private Attributes

- vector< **Word** > **AlicePublicSbgp**
- vector< **Word** > **BobPublicSbgp**
- **Word** **AliceKeyDecomposition**
- **Word** **BobKeyDecomposition**
- **Word** **AliceKey**
- **Word** **BobKey**
- vector< **Word** > **AliceConjugatedSbgp**
- vector< **Word** > **BobConjugatedSbgp**
- **Word** **theSharedKey**



### 9.2.1 Detailed Description

Definition at line 23 of file AAGKeyGeneration.h.

### 9.2.2 Constructor & Destructor Documentation

**9.2.2.1** `AAGProtocolInstance::AAGProtocolInstance (int N, const vector< Word > & aSbgp, const vector< Word > & bSbgp, const Word & aDecomp, const Word & bDecomp, bool useForm = true)`

### 9.2.3 Member Function Documentation

**9.2.3.1** `static AAGProtocolInstance AAGProtocolInstance::challenge (int N, int sg_conj_len, int c, int k, int key_len) [static]`

**9.2.3.2** `static Word AAGProtocolInstance::generateHardProductOfGenerators (int num_gens, int product_length) [static, private]`

Function generates a hard key.

**9.2.3.3** `vector< Word > AAGProtocolInstance::getAliceConjSbgp () const [inline]`

Definition at line 54 of file AAGKeyGeneration.h.

References AliceConjugatedSbgp.

Referenced by AAGKeyPairGenerator::getFalsePair(), and AAGKeyPairGenerator::getTruePair().

**9.2.3.4** `Word AAGProtocolInstance::getAliceKey () const [inline]`

Definition at line 57 of file AAGKeyGeneration.h.

References AliceKey.

**9.2.3.5** `vector< Word > AAGProtocolInstance::getAlicePublicSbgp () const [inline]`

Definition at line 52 of file AAGKeyGeneration.h.

References AlicePublicSbgp.

Referenced by AAGKeyPairGenerator::getFalsePair(), and AAGKeyPairGenerator::getTruePair().

### 9.2.3.6 `vector< Word > AAGProtocolInstance::getBobConjSbgp () const` `[inline]`

Definition at line 55 of file AAGKeyGeneration.h.

References BobConjugatedSbgp.

### 9.2.3.7 `Word AAGProtocolInstance::getBobKey () const` `[inline]`

Definition at line 58 of file AAGKeyGeneration.h.

References BobKey.

### 9.2.3.8 `vector< Word > AAGProtocolInstance::getBobPublicSbgp () const` `[inline]`

Definition at line 53 of file AAGKeyGeneration.h.

References BobPublicSbgp.

### 9.2.3.9 `Word AAGProtocolInstance::getSharedKey () const` `[inline]`

Definition at line 59 of file AAGKeyGeneration.h.

References theSharedKey.

### 9.2.3.10 `void AAGProtocolInstance::printStats (ostream & out)`

### 9.2.3.11 `static AAGProtocolInstance AAGProtocolInstance::random (int N, int num_gens, int min_len, int max_len, int AliceDecompositionLength, int BobDecompositionLength)` `[static]`

## 9.2.4 Member Data Documentation

### 9.2.4.1 `vector< Word > AAGProtocolInstance::AliceConjugatedSbgp` `[private]`

Definition at line 99 of file AAGKeyGeneration.h.

Referenced by getAliceConjSbgp().

### 9.2.4.2 `Word AAGProtocolInstance::AliceKey` `[private]`

Definition at line 96 of file AAGKeyGeneration.h.

Referenced by `getAliceKey()`.

#### 9.2.4.3 Word AAGProtocolInstance::AliceKeyDecomposition [private]

Definition at line 93 of file `AAGKeyGeneration.h`.

#### 9.2.4.4 vector< Word > AAGProtocolInstance::AlicePublicSbgp [private]

Definition at line 90 of file `AAGKeyGeneration.h`.

Referenced by `getAlicePublicSbgp()`.

#### 9.2.4.5 vector< Word > AAGProtocolInstance::BobConjugatedSbgp [private]

Definition at line 100 of file `AAGKeyGeneration.h`.

Referenced by `getBobConjSbgp()`.

#### 9.2.4.6 Word AAGProtocolInstance::BobKey [private]

Definition at line 97 of file `AAGKeyGeneration.h`.

Referenced by `getBobKey()`.

#### 9.2.4.7 Word AAGProtocolInstance::BobKeyDecomposition [private]

Definition at line 94 of file `AAGKeyGeneration.h`.

#### 9.2.4.8 vector< Word > AAGProtocolInstance::BobPublicSbgp [private]

Definition at line 91 of file `AAGKeyGeneration.h`.

Referenced by `getBobPublicSbgp()`.

#### 9.2.4.9 Word AAGProtocolInstance::theSharedKey [private]

Definition at line 102 of file `AAGKeyGeneration.h`.

Referenced by `getSharedKey()`.

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

- `CryptoAAG/include/AAGKeyGeneration.h`

## 9.3 AdvDehnAlgorithm Class Reference

```
#include <AdvDehnAlgorithm.h>
```

### Public Member Functions

- **AdvDehnAlgorithm** (const **FPGroup** &G, const **Word** &w)
- **AdvDehnAlgorithm** (const **FPGroup** &G, const set< **Word** > &gens, const **Word** &w)
- const **IntLabeledGraph** & **getFSA** () const
- bool **builtup** (set< **Word** > \*conj=0, int coset\_limit=100000)
- bool **isLoop** (const **Word** &w) const

### Private Member Functions

- void **addCycle** (const **Word** &w, int origin)

### Private Attributes

- const **FPGroup** **theGroup**
- const **Word** **theWord**
- set< int > **checkedStates**
- **IntLabeledGraph** **theFSA**

#### 9.3.1 Detailed Description

Requires a revision. set< int > checkedStates; must be replaced with int lastChecked-Vertex;

Definition at line 24 of file AdvDehnAlgorithm.h.

## 9.3.2 Constructor & Destructor Documentation

**9.3.2.1** AdvDehnAlgorithm::AdvDehnAlgorithm (const FPGroup & *G*, const Word & *w*)

**9.3.2.2** AdvDehnAlgorithm::AdvDehnAlgorithm (const FPGroup & *G*, const set< Word > & *gens*, const Word & *w*)

## 9.3.3 Member Function Documentation

**9.3.3.1** void AdvDehnAlgorithm::addCycle (const Word & *w*, int *origin*)  
[private]

**9.3.3.2** bool AdvDehnAlgorithm::builtup (set< Word > \* *conj* = 0, int *coset\_limit* = 100000)

**9.3.3.3** const IntLabeledGraph& AdvDehnAlgorithm::getFSA () const  
[inline]

Definition at line 47 of file AdvDehnAlgorithm.h.

References theFSA.

**9.3.3.4** bool AdvDehnAlgorithm::isLoop (const Word & *w*) const

## 9.3.4 Member Data Documentation

**9.3.4.1** set< int > AdvDehnAlgorithm::checkedStates [private]

Definition at line 77 of file AdvDehnAlgorithm.h.

**9.3.4.2** IntLabeledGraph AdvDehnAlgorithm::theFSA [private]

Definition at line 79 of file AdvDehnAlgorithm.h.

Referenced by getFSA().

**9.3.4.3** const FPGroup AdvDehnAlgorithm::theGroup [private]

Definition at line 74 of file AdvDehnAlgorithm.h.

#### 9.3.4.4 `const Word AdvDehnAlgorithm::theWord` `[private]`

Definition at line 75 of file AdvDehnAlgorithm.h.

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

- Group/include/AdvDehnAlgorithm.h

## 9.4 AEKeyExchange Class Reference

```
#include <AEProtocol.h>
```

### Public Member Functions

- **AEKeyExchange** (int *n*, int *p*, const **TPTuple** &ttp)
- **ProdElement** **alicePublicKey** ()
- **ProdElement** **bobPublicKey** ()

### Static Public Member Functions

- static **TPTuple** **generateTuples** (const **TTP\_Conf** &ttp\_conf, const **BSets** &bs)

### Private Member Functions

- **ProdElement** **starMult** (const **ProdElement** &pe, const **BureauGenerator** &bg)
- **ProdElement** **generatePublicKey** (const vector< **Word** > &v, int *r*, int *m*, int *wl*)

### Private Attributes

- int **the\_n**
- int **the\_p**
- **TPTuple** **theTPTuple**
- **MatrixFp** **M0**

#### 9.4.1 Detailed Description

Definition at line 202 of file AEProtocol.h.

#### 9.4.2 Constructor & Destructor Documentation

##### 9.4.2.1 AEKeyExchange::AEKeyExchange (int *n*, int *p*, const TPTuple &ttp) [inline]

Definition at line 205 of file AEProtocol.h.

### 9.4.3 Member Function Documentation

#### 9.4.3.1 ProdElement AEKeyExchange::alicePublicKey () [inline]

Definition at line 213 of file AEProtocol.h.

References generatePublicKey(), theTTPTuple, and TTPTuple::WL.

#### 9.4.3.2 ProdElement AEKeyExchange::bobPublicKey () [inline]

Definition at line 219 of file AEProtocol.h.

References generatePublicKey(), theTTPTuple, and TTPTuple::WR.

#### 9.4.3.3 ProdElement AEKeyExchange::generatePublicKey (const vector< Word > & v, int r, int m, int wl) [private]

Referenced by alicePublicKey(), and bobPublicKey().

#### 9.4.3.4 static TTPTuple AEKeyExchange::generateTuples (const TTP\_Conf & ttp\_conf, const BSets & bs) [static]

#### 9.4.3.5 ProdElement AEKeyExchange::starMult (const ProdElement & pe, const BureauGenerator & bg) [private]

### 9.4.4 Member Data Documentation

#### 9.4.4.1 MatrixFp AEKeyExchange::M0 [private]

Definition at line 237 of file AEProtocol.h.

#### 9.4.4.2 int AEKeyExchange::the\_n [private]

Definition at line 234 of file AEProtocol.h.

#### 9.4.4.3 int AEKeyExchange::the\_p [private]

Definition at line 235 of file AEProtocol.h.

#### 9.4.4.4 TTPTuple AEKeyExchange::theTTPTuple [private]

Definition at line 236 of file AEProtocol.h.



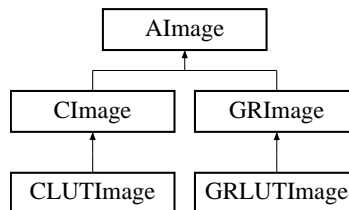
Referenced by `alicePublicKey()`, and `bobPublicKey()`.

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

- `CryptoAE/include/AEProtocol.h`

## 9.5 AImage Class Reference

`#include <AImage.h>` Inheritance diagram for AImage::



### Public Member Functions

- **AImage** ()
- **AImage** (int w, int h)
- **AImage** (const string &in\_file\_name)
- **AImage** (const **AImage** &i)
- **AImage** (const **AImage** &image, int tlx, int tly, int brx, int bry, bool zero\_padding)
- int **getSize** () const
- int **getWidth** () const
- int **getHeight** () const
- virtual void **saveTo** (const string &file\_name)=0
- virtual **IMAGE\_TYPE** **getType** () const =0

### Protected Attributes

- int **width**
- int **height**
- int **size**

#### 9.5.1 Detailed Description

Definition at line 51 of file AImage.h.

#### 9.5.2 Constructor & Destructor Documentation

##### 9.5.2.1 AImage::AImage () [inline]

Definition at line 55 of file AImage.h.

**9.5.2.2 AImage::AImage (int *w*, int *h*) [inline]**

Definition at line 58 of file AImage.h.

**9.5.2.3 AImage::AImage (const string & *in\_file\_name*) [inline]**

Definition at line 62 of file AImage.h.

**9.5.2.4 AImage::AImage (const AImage & *i*) [inline]**

Definition at line 65 of file AImage.h.

**9.5.2.5 AImage::AImage (const AImage & *image*, int *tlx*, int *tly*, int *brx*, int *bry*, bool *zero\_padding*) [inline]**

Definition at line 68 of file AImage.h.

**9.5.3 Member Function Documentation****9.5.3.1 int AImage::getHeight () const [inline]**

Definition at line 79 of file AImage.h.

References height.

Referenced by WordDraw::drawVerticalGrid(), and WordDraw::WordDraw().

**9.5.3.2 int AImage::getSize () const [inline]**

Definition at line 75 of file AImage.h.

References size.

**9.5.3.3 virtual IMAGE\_TYPE AImage::getType () const [pure virtual]**

Implemented in **GRIImage** (p. 264), and **CImage** (p. 155).

**9.5.3.4 int AImage::getWidth () const [inline]**

Definition at line 77 of file AImage.h.

References width.

Referenced by WordDraw::WordDraw().

#### **9.5.3.5 virtual void AImage::saveTo (const string &file\_name) [pure virtual]**

Implemented in **GRIImage** (p. 264), and **CImage** (p. 156).

### **9.5.4 Member Data Documentation**

#### **9.5.4.1 int AImage::height [protected]**

Definition at line 91 of file AImage.h.

Referenced by getHeight().

#### **9.5.4.2 int AImage::size [protected]**

Definition at line 93 of file AImage.h.

Referenced by getSize(), and GRIImage::GRIImage().

#### **9.5.4.3 int AImage::width [protected]**

Definition at line 89 of file AImage.h.

Referenced by getWidth().

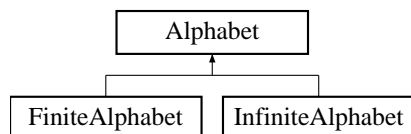
The documentation for this class was generated from the following file:

- Graphics/include/**AImage.h**

## 9.6 Alphabet Class Reference

Implements an abstract interface for all alphabet realizations.

#include <Alphabet.h> Inheritance diagram for Alphabet::



### Public Member Functions

- virtual int **getNum** (const string &letter) const =0  
*Interface for conversion from the letter name to an integer.*
- virtual string **getLetter** (int index) const =0  
*Interface for conversion from an integer index into the corresponding letter name.*
- void **printWord** (ostream &out, const **Word** &w) const  
*Output a word in the alphabet letters.*
- **Word readWord** (istream &in) const  
*Read a word from a stream in the alphabet letters.*
- void **printVector** (ostream &out, const vector< **Word** > &v) const  
*Output a vector of words in the alphabet letters.*
- vector< **Word** > **readVector** (istream &in) const  
*Read a vector of words from a stream in the alphabet letters.*

### 9.6.1 Detailed Description

Implements an abstract interface for all alphabet realizations.

Definition at line 34 of file Alphabet.h.

## 9.6.2 Member Function Documentation

### 9.6.2.1 `virtual string Alphabet::getLetter (int index) const [pure virtual]`

Interface for conversion from an integer index into the corresponding letter name.

Implemented in **InfiniteAlphabet** (p. 273).

### 9.6.2.2 `virtual int Alphabet::getNum (const string & letter) const [pure virtual]`

Interface for conversion from the letter name to an integer.

Implemented in **InfiniteAlphabet** (p. 273).

### 9.6.2.3 `void Alphabet::printVector (ostream & out, const vector< Word > & v) const`

Output a vector of words in the alphabet letters.

### 9.6.2.4 `void Alphabet::printWord (ostream & out, const Word & w) const`

Output a word in the alphabet letters.

Referenced by `Word::printOn()`.

### 9.6.2.5 `vector<Word> Alphabet::readVector (istream & in) const`

Read a vector of words from a stream in the alphabet letters.

### 9.6.2.6 `Word Alphabet::readWord (istream & in) const`

Read a word from a stream in the alphabet letters.

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

- `Alphabet/include/Alphabet.h`

## 9.7 AParser Class Reference

```
#include <Parser.h>
```

### Public Member Functions

- **AParser** (istream &in)
- **~AParser** ()
- void **parse** ()
- **AInputType** **getType** () const
- **FiniteAlphabet** **getAlphabet** () const

### Private Attributes

- class AlphabetFlexLexer \* **localFlexLexer**
- **AInputType** **theType**
- **FiniteAlphabet** \* **theAlphabet**

#### 9.7.1 Detailed Description

Definition at line 32 of file Parser.h.

#### 9.7.2 Constructor & Destructor Documentation

**9.7.2.1** AParser::AParser (istream & *in*)

**9.7.2.2** AParser::~~AParser ()

#### 9.7.3 Member Function Documentation

**9.7.3.1** FiniteAlphabet AParser::getAlphabet () const

**9.7.3.2** AInputType AParser::getType () const

**9.7.3.3** void AParser::parse ()

#### 9.7.4 Member Data Documentation

**9.7.4.1** class AlphabetFlexLexer\* AParser::localFlexLexer [private]

Definition at line 41 of file Parser.h.

#### 9.7.4.2 FiniteAlphabet\* AParser::theAlphabet [private]

Definition at line 43 of file Parser.h.

#### 9.7.4.3 AInputType AParser::theType [private]

Definition at line 42 of file Parser.h.

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

- Alphabet/include/Parser.h



## 9.8 StraightLineProgramWord::Assertion Struct Reference

Structure used in function **equal()** (p. 512) only.

### Public Member Functions

- **Assertion** (bool b, int v1, int v2, **LongInteger** l)
- bool **operator<** (const **Assertion** &A) const
- bool **similar** (const **Assertion** &A) const

### Public Attributes

- bool **theBase1**  
*true if and only if the vertex 1 comes from the first SLP*
- int **theVertex1**  
*the number of the vertex 1*
- int **theVertex2**  
*the number of the vertex 2*
- **LongInteger** **theLength**  
*shift in theVertex 1*

### Friends

- ostream & **operator<<** (ostream &os, const **Assertion** &A)

#### 9.8.1 Detailed Description

Structure used in function **equal()** (p. 512) only. This structure is used for comparison of 2 straight line program words (as elements of a semigroup). **theVertex1** is a number of a vertex in the first graph, **theVertex2** is a number of a vertex in the second graph, **theBase1** is true if **theVertex1** is a base vertex, **theLength** is the distance between the vertices.

Definition at line 86 of file StraightLineProgramWord.h.

## 9.8.2 Constructor & Destructor Documentation

### 9.8.2.1 `StraightLineProgramWord::Assertion::Assertion (bool b, int v1, int v2, LongInteger l) [inline]`

Definition at line 88 of file `StraightLineProgramWord.h`.

## 9.8.3 Member Function Documentation

### 9.8.3.1 `bool StraightLineProgramWord::Assertion::operator< (const Assertion & A) const [inline]`

Definition at line 90 of file `StraightLineProgramWord.h`.

References `theBase1`, `theLength`, `theVertex1`, and `theVertex2`.

### 9.8.3.2 `bool StraightLineProgramWord::Assertion::similar (const Assertion & A) const [inline]`

Definition at line 108 of file `StraightLineProgramWord.h`.

References `theBase1`, `theVertex1`, and `theVertex2`.

## 9.8.4 Friends And Related Function Documentation

### 9.8.4.1 `ostream& operator<< (ostream & os, const Assertion & A) [friend]`

Definition at line 112 of file `StraightLineProgramWord.h`.

## 9.8.5 Member Data Documentation

### 9.8.5.1 `bool StraightLineProgramWord::Assertion::theBase1`

true if and only if the vertex 1 comes from the first SLP

Definition at line 118 of file `StraightLineProgramWord.h`.

Referenced by `operator<()`, and `similar()`.

### 9.8.5.2 `LongInteger StraightLineProgramWord::Assertion::theLength`

shift in theVertex 1

Definition at line 127 of file StraightLineProgramWord.h.

Referenced by operator<().

#### 9.8.5.3 int StraightLineProgramWord::Assertion::theVertex1

the number of the vertex 1

Definition at line 121 of file StraightLineProgramWord.h.

Referenced by operator<(), and similar().

#### 9.8.5.4 int StraightLineProgramWord::Assertion::theVertex2

the number of the vertex 2

Definition at line 124 of file StraightLineProgramWord.h.

Referenced by operator<(), and similar().

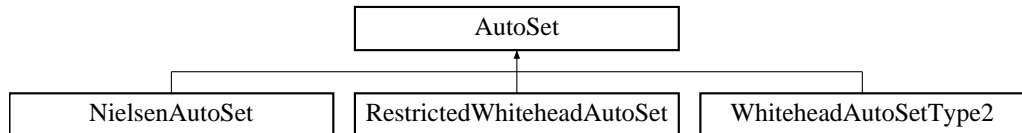
The documentation for this struct was generated from the following file:

- FreeGroup/include/StraightLineProgramWord.h

## 9.9 AutoSet Class Reference

Abstract interface for a set of Maps (Automorphisms).

#include <WhiteheadAutoSet.h> Inheritance diagram for AutoSet::



### Public Member Functions

- virtual const **SetOfMaps** & **getSet** () const =0  
*Returns the corresponding set of maps.*

#### 9.9.1 Detailed Description

Abstract interface for a set of Maps (Automorphisms).

Definition at line 43 of file WhiteheadAutoSet.h.

#### 9.9.2 Member Function Documentation

##### 9.9.2.1 virtual const SetOfMaps& AutoSet::getSet () const [pure virtual]

Returns the corresponding set of maps.

Implemented in **NielsenAutoSet** (p. 342), **RestrictedWhiteheadAutoSet** (p. 470), and **WhiteheadAutoSetType2** (p. 626).

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

- Maps/include/WhiteheadAutoSet.h

## 9.10 `BalancedTree< Obj >` Class Template Reference

Class **BalancedTree** (p. 111) (container class intended to keep an ordered sequence of objects, supports fast  $n \log n$  insertions).

```
#include <BalancedTree.h>
```

### Classes

- struct **BTNode**

### Public Member Functions

- **BalancedTree** ()  
*Default constructor.*
- template<class ConstObjIter >  
**BalancedTree** (const ConstObjIter &B, const ConstObjIter &E)  
*Constructor. Creates a tree for a sequence in range [B,E).*
- **~BalancedTree** ()  
*Destructor recursively frees memory.*
- int **size** () const
- template<class ConstObjIter >  
void **insert** (int pos, const ConstObjIter &B, const ConstObjIter &E)  
*Insert a sequence of objects bounded by [B,E) into a position pos.*
- void **insert** (int pos, const Obj &obj)  
*Insert an object obj into position pos.*
- list< Obj > **getList** () const  
*Get an ordered list of objects stored in a balanced tree.*

### Private Member Functions

- void **getList** (**BTNode** \*node, list< Obj > &result) const  
*Get an ordered list of objects in a subtree with a root node.*
- void **insert** (**BTNode** \*parent, **BTNode** \*node, int offset, const Obj &obj)  
*Insert an object obj into a subtree with a root node at the offset position offset.*

- void **rotateLeft** (BTNode \*parent, BTNode \*child)  
*Left rotation.*
- void **rotateRight** (BTNode \*parent, BTNode \*child)  
*Right rotation.*

## Private Attributes

- BTNode \* **theRoot**

### 9.10.1 Detailed Description

**template<class Obj> class BalancedTree< Obj >**

Class **BalancedTree** (p. 111) (container class intended to keep an ordered sequence of objects, supports fast  $n \log n$  insertions). Typical usage for this class is generation of trivial elements of groups, where a lot of insertions of relators is performed, i.e., start from a trivial word and consequently insert relators (with their inverses and cyclic permutations). The result will be a word representing the identity of the group. The main feature of this class - fast  $n \log n$  asymptotic insertions which cannot be achieved using standard vector<...> and list<...>.

Definition at line 34 of file BalancedTree.h.

### 9.10.2 Constructor & Destructor Documentation

**9.10.2.1** **template<class Obj > BalancedTree< Obj >::BalancedTree ()**  
**[inline]**

Default constructor.

Definition at line 66 of file BalancedTree.h.

**9.10.2.2** **template<class Obj > template<class ConstObjIter > BalancedTree< Obj >::BalancedTree (const ConstObjIter & B, const ConstObjIter & E) [inline]**

Constructor. Creates a tree for a sequence in range [B,E).

Definition at line 68 of file BalancedTree.h.

References **BalancedTree< Obj >::insert()**.

### 9.10.2.3 `template<class Obj > BalancedTree< Obj >::~~BalancedTree ()` `[inline]`

Destructor recursively frees memory.

Definition at line 72 of file `BalancedTree.h`.

References `BalancedTree< Obj >::theRoot`.

## 9.10.3 Member Function Documentation

### 9.10.3.1 `template<class Obj > void BalancedTree< Obj >::getList (BTNode * node, list< Obj > &result) const [inline, private]`

Get an ordered list of objects in a subtree with a root node.

Definition at line 128 of file `BalancedTree.h`.

References `BalancedTree< Obj >::getList()`, `BalancedTree< Obj >::BTNode::subtree1`, `BalancedTree< Obj >::BTNode::subtree2`, and `BalancedTree< Obj >::BTNode::theObject`.

### 9.10.3.2 `template<class Obj > list< Obj > BalancedTree< Obj >::getList () const [inline]`

Get an ordered list of objects stored in a balanced tree.

Definition at line 113 of file `BalancedTree.h`.

References `BalancedTree< Obj >::theRoot`.

Referenced by `BalancedTree< Obj >::getList()`.

### 9.10.3.3 `template<class Obj > void BalancedTree< Obj >::insert (BTNode * parent, BTNode *node, int offset, const Obj &obj) [inline, private]`

Insert an onject `obj` into a subtree with a root node at the offset position `offset`.

Definition at line 136 of file `BalancedTree.h`.

References `BalancedTree< Obj >::BTNode::height1`, `BalancedTree< Obj >::BTNode::height2`, `BalancedTree< Obj >::insert()`, `BalancedTree< Obj >::rotateLeft()`, `BalancedTree< Obj >::rotateRight()`, `BalancedTree< Obj >::BTNode::subtree1`, `BalancedTree< Obj >::BTNode::subtree2`, `BalancedTree< Obj >::BTNode::weight1`, and `BalancedTree< Obj >::BTNode::weight2`.

### 9.10.3.4 **template<class Obj > void BalancedTree< Obj >::insert (int pos, const Obj & obj) [inline]**

Insert an object obj into position pos.

Definition at line 105 of file BalancedTree.h.

References BalancedTree< Obj >::insert(), and BalancedTree< Obj >::theRoot.

### 9.10.3.5 **template<class Obj > template<class ConstObjIter > void BalancedTree< Obj >::insert (int pos, const ConstObjIter & B, const ConstObjIter & E) [inline]**

Insert a sequence of objects bounded by [B,E) into a position pos.

Definition at line 98 of file BalancedTree.h.

Referenced by BalancedTree< Obj >::BalancedTree(), and BalancedTree< Obj >::insert().

### 9.10.3.6 **template<class Obj > void BalancedTree< Obj >::rotateLeft (BTNode \* parent, BTNode \* child) [inline, private]**

Left rotation.

Definition at line 168 of file BalancedTree.h.

References BalancedTree< Obj >::BTNode::height1, BalancedTree< Obj >::BTNode::height2, BalancedTree< Obj >::BTNode::subtree1, BalancedTree< Obj >::BTNode::subtree2, BalancedTree< Obj >::theRoot, BalancedTree< Obj >::BTNode::weight1, and BalancedTree< Obj >::BTNode::weight2.

Referenced by BalancedTree< Obj >::insert().

### 9.10.3.7 **template<class Obj > void BalancedTree< Obj >::rotateRight (BTNode \* parent, BTNode \* child) [inline, private]**

Right rotation.

Definition at line 217 of file BalancedTree.h.

References BalancedTree< Obj >::BTNode::height1, BalancedTree< Obj >::BTNode::height2, BalancedTree< Obj >::BTNode::subtree1, BalancedTree< Obj >::BTNode::subtree2, BalancedTree< Obj >::theRoot, BalancedTree< Obj >::BTNode::weight1, and BalancedTree< Obj >::BTNode::weight2.

Referenced by BalancedTree< Obj >::insert().



### 9.10.3.8 `template<class Obj > int BalancedTree< Obj >::size () const` `[inline]`

Definition at line 83 of file `BalancedTree.h`.

References `BalancedTree< Obj >::theRoot`, `BalancedTree< Obj >::BTNode::weight1`, and `BalancedTree< Obj >::BTNode::weight2`.

## 9.10.4 Member Data Documentation

### 9.10.4.1 `template<class Obj > BTNode* BalancedTree< Obj >::theRoot` `[private]`

Definition at line 274 of file `BalancedTree.h`.

Referenced by `BalancedTree< Obj >::getList()`, `BalancedTree< Obj >::insert()`, `BalancedTree< Obj >::rotateLeft()`, `BalancedTree< Obj >::rotateRight()`, `BalancedTree< Obj >::size()`, and `BalancedTree< Obj >::~~BalancedTree()`.

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

- `general/include/BalancedTree.h`

## 9.11 BG::BGMonomial Struct Reference

```
#include <BaumslagGersten.h>
```

### Public Types

- enum { **BSat**, **b** }

### Public Attributes

- enum BG::BGMonomial:: { ... } **type**
- **Marking U**
- **Marking X**
- **Marking K**
- int **expb**

#### 9.11.1 Detailed Description

This data type is used to represent the input of the procedure solving the word problem in  $BG(1,2)$ . A word is given as a sequence of BGMonomials, i.e., letters  $b$ ,  $b^{-1}$  and elements of the Baumslag-Solitar group  $BS(1,2)=\langle a,t \mid a^t=a^2 \rangle$ , where the latter are represented by **triple** (p. 577) markings  $(U,X,K)$  in a common power circuit, where: i) all  $U$ ,  $X$ , and  $K$  must have disjoint supports, ii) all  $U$  must be sources, iii) all incoming arcs to  $X$  and  $K$  must originate in the corresponding  $U$ , and iv) arcs from  $U$  to  $X$  must have the opposite sign of the corresponding node-signs in  $X$ .

Definition at line 41 of file BaumslagGersten.h.

#### 9.11.2 Member Enumeration Documentation

##### 9.11.2.1 anonymous enum

**Enumerator:**

*BSat*

*b*

Definition at line 43 of file BaumslagGersten.h.

### 9.11.3 Member Data Documentation

#### 9.11.3.1 int BG::BGMonomial::expb

Definition at line 47 of file BaumslagGersten.h.

#### 9.11.3.2 Marking BG::BGMonomial::K

Definition at line 46 of file BaumslagGersten.h.

#### 9.11.3.3 enum { ... } BG::BGMonomial::type

#### 9.11.3.4 Marking BG::BGMonomial::U

Definition at line 46 of file BaumslagGersten.h.

#### 9.11.3.5 Marking BG::BGMonomial::X

Definition at line 46 of file BaumslagGersten.h.

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

- HigmanGroup/include/BaumslagGersten.h

## 9.12 BKLeftNormalForm Class Reference

Birman-Ko-Lee left normal form.

```
#include <BKLeftNormalForm.h>
```

### Public Types

- typedef **triple**< int, int, list< **Permutation** > > **NF**

*Presentation of a normal form.*

### Public Member Functions

- **BKLeftNormalForm** ()

*Create trivial normal form.*

- **BKLeftNormalForm** (const **BKLeftNormalForm** &bkl)

*Copy constructor.*

- **BKLeftNormalForm** (int rank, int p, const list< **Permutation** > &d)

*Create normal form by its presentation (no check that given presentation is correct, use adjustDecomposition if the presentation is incorrect).*

- **BKLeftNormalForm** (const **NF** &bkl)

*Create normal form by its presentation (no check that given presentation is correct, use adjustDecomposition if the presentation is incorrect).*

- **BKLeftNormalForm** (const **BraidGroup** &G, const **Word** &w)

*Create a normal form of a braid word.*

- **BKLeftNormalForm** & operator= (const **BKLeftNormalForm** &bkl)

*Assignment operator.*

- **BKLeftNormalForm** & operator= (const **NF** &bkl)

*Assignment operator.*

- **BKLeftNormalForm** & operator\*= (const **BKLeftNormalForm** &bkl)
- **BKLeftNormalForm** operator\* (const **BKLeftNormalForm** &bkl) const
- **BKLeftNormalForm** operator- () const
- bool operator== (const **BKLeftNormalForm** &bkl) const
- bool operator!= (const **BKLeftNormalForm** &bkl) const

- **bool operator<** (const **BKLeftNormalForm** &bkl) const
- **int getPower** () const  
*Get the power of omega.*
- **const list< Permutation > & getDecomposition** () const  
*Get the decomposition.*
- **operator NF** () const  
*Get the presentation of the normal form.*
- **bool isTrivial** () const  
*Check if the normal form os trivial.*
- **Word getWord** () const  
*Return a word represented by the normal form.*
- **pair< bool, Word > areConjugate** (const **BKLeftNormalForm** &bkl) const  
*Check if two normal forms are conjugate.*
- **pair< BKLeftNormalForm, BKLeftNormalForm > cycle** () const  
*Cycle operation.*
- **pair< BKLeftNormalForm, BKLeftNormalForm > decycle** () const  
*Decycle operation.*
- **pair< BKLeftNormalForm, BKLeftNormalForm > findSSSRepresentative** () const  
*Find a representative of the supper summit set.*
- **void setPower** (int p)  
*Set the power of  $\Delta$ . The result is always a correct normal form.*
- **void setDecomposition** (const list< **Permutation** > &d)  
*Set the decomposition  $(\xi_1, \dots, \xi_m)$ . The result might be an incorrect form (not satisfying some greedy conditions of this BKL form).*

## Static Public Member Functions

- **static Permutation getTinyTwistPermutation** (int theIndex)  
*Returns a cyclic permutation  $\delta = (1, 2, 3, \dots, n - 1, 0)$ .*

## Private Types

- enum **transformationResult** { **TWO\_MULTIPLIERS**, **ONE\_MULTIPLIER**, **NO\_CHANGE** }

*The result of the transformation.*

## Private Member Functions

- **NF inverse** () const  
*Invert the normal form.*
- **NF multiply** (const **BKLLeftNormalForm** &bkl) const  
*Multiply the normal form by another normal form on the right.*

## Static Private Member Functions

- static **transformationResult transform** (int theIndex, **Permutation** &p1, **Permutation** &p2)  
*Main function for computing the normal form of the word.*
- static set< **Permutation** > **getSimpleConjugators** (const **NF** &bkl)  
*Function returns a set of permutation braids (well, permutations) such that ... Must be checked.*
- static set< **Permutation** > **getSimpleSummitConjugators** (const **NF** &bkl)  
*Function returns a set of permutation braids (well, permutations) such that ... Must be checked.*
- static void **adjustDecomposition** (int rank, int &power, list< **Permutation** > &decomp)  
*Function adjusting the decomposition in a normal form.*

## Private Attributes

- int **theRank**  
*The rank of the braid group (number of strands).*
- int **theOmegaPower**  
*Power of omega.*

- list< **Permutation** > **theDecomposition**

*Sequence of permutations.*

### 9.12.1 Detailed Description

Birman-Ko-Lee left normal form. The standard presentation for BKL is a pair  $(p, (\xi_1, \dots, \xi_m))$ , where  $p \in Z$  and each  $\xi_i \in S_n$  is a permutation.

Definition at line 34 of file BKLeftNormalForm.h.

### 9.12.2 Member Typedef Documentation

#### 9.12.2.1 typedef triple< int , int , list< **Permutation** > > BKLeftNormalForm::NF

Presentation of a normal form. The first component specifies the rank of a braid group. The second component specifies the power of the twist. The third component specifies the list of braid permutations.

Definition at line 45 of file BKLeftNormalForm.h.

### 9.12.3 Member Enumeration Documentation

#### 9.12.3.1 enum BKLeftNormalForm::transformationResult [private]

The result of the transformation.

**Enumerator:**

*TWO\_MULTIPLIERS*  
*ONE\_MULTIPLIER*  
*NO\_CHANGE*

Definition at line 192 of file BKLeftNormalForm.h.

### 9.12.4 Constructor & Destructor Documentation

#### 9.12.4.1 BKLeftNormalForm::BKLeftNormalForm () [inline]

Create trivial normal form.

Definition at line 55 of file BKLeftNormalForm.h.

#### 9.12.4.2 **BKLeftNormalForm::BKLeftNormalForm (const BKLeftNormalForm & *bkl*) [inline]**

Copy constructor.

Definition at line 60 of file BKLeftNormalForm.h.

#### 9.12.4.3 **BKLeftNormalForm::BKLeftNormalForm (int *rank*, int *p*, const list< Permutation > & *d*) [inline]**

Create normal form by its presentation (no check that given presentation is correct, use adjustDecomposition if the presentation is incorrect).

Definition at line 66 of file BKLeftNormalForm.h.

#### 9.12.4.4 **BKLeftNormalForm::BKLeftNormalForm (const NF & *bkl*) [inline]**

Create normal form by its presentation (no check that given presentation is correct, use adjustDecomposition if the presentation is incorrect).

Definition at line 72 of file BKLeftNormalForm.h.

#### 9.12.4.5 **BKLeftNormalForm::BKLeftNormalForm (const BraidGroup & *G*, const Word & *w*)**

Create a normal form of a braid word.

### 9.12.5 Member Function Documentation

#### 9.12.5.1 **static void BKLeftNormalForm::adjustDecomposition (int *rank*, int & *power*, list< Permutation > & *decomp*) [static, private]**

Function adjusting the decomposition in a normal form.

#### 9.12.5.2 **pair< bool , Word > BKLeftNormalForm::areConjugate (const BKLeftNormalForm & *bkl*) const**

Check if two normal forms are conjugate.



**9.12.5.3** `pair< BKLeftNormalForm , BKLeftNormalForm >  
BKLeftNormalForm::cycle () const`

Cycle operation.

**9.12.5.4** `pair< BKLeftNormalForm , BKLeftNormalForm >  
BKLeftNormalForm::decycle () const`

Decycle operation.

**9.12.5.5** `pair< BKLeftNormalForm , BKLeftNormalForm >  
BKLeftNormalForm::findSSSRepresentative () const`

Find a representative of the supper summit set.

**9.12.5.6** `const list< Permutation >& BKLeftNormalForm::getDecomposition  
() const [inline]`

Get the decomposition.

Definition at line 126 of file BKLeftNormalForm.h.

References theDecomposition.

**9.12.5.7** `int BKLeftNormalForm::getPower () const [inline]`

Get the power of omega.

Definition at line 124 of file BKLeftNormalForm.h.

References theOmegaPower.

**9.12.5.8** `static set<Permutation> BKLeftNormal-  
Form::getSimpleConjugators (const NF & bkl) [static,  
private]`

Function returns a set of permutation braids (well, permutations) such that ... Must be checked.

**9.12.5.9 static set<Permutation> BKLeftNormalForm::getSimpleSummitConjugators (const NF & *bkl*) [static, private]**

Function returns a set of permutation braids (well, permutations) such that ... Must be checked.

**9.12.5.10 static Permutation BKLeftNormalForm::getTinyTwistPermutation (int *theIndex*) [inline, static]**

Returns a cyclic permutation  $\delta = (1, 2, 3, \dots, n-1, 0)$ .

Definition at line 135 of file BKLeftNormalForm.h.

**9.12.5.11 Word BKLeftNormalForm::getWord () const**

Return a word represented by the normal form.

**9.12.5.12 NF BKLeftNormalForm::inverse () const [private]**

Invert the normal form.

**9.12.5.13 bool BKLeftNormalForm::isTrivial () const [inline]**

Check if the normal form os trivial.

Definition at line 132 of file BKLeftNormalForm.h.

References theDecomposition, and theOmegaPower.

**9.12.5.14 NF BKLeftNormalForm::multiply (const BKLeftNormalForm & *bkl*) const [private]**

Multiply the normal form by another normal form on the right.

**9.12.5.15 BKLeftNormalForm::operator NF () const [inline]**

Get the presentation of the normal form.

Definition at line 129 of file BKLeftNormalForm.h.

References theDecomposition, theOmegaPower, and theRank.

**9.12.5.16** `bool BKLeftNormalForm::operator!= (const BKLeftNormalForm & bkl) const`

**9.12.5.17** `BKLeftNormalForm BKLeftNormalForm::operator* (const BKLeftNormalForm & bkl) const`

**9.12.5.18** `BKLeftNormalForm& BKLeftNormalForm::operator*= (const BKLeftNormalForm & bkl)`

**9.12.5.19** `BKLeftNormalForm BKLeftNormalForm::operator- () const`

**9.12.5.20** `bool BKLeftNormalForm::operator< (const BKLeftNormalForm & bkl) const`

**9.12.5.21** `BKLeftNormalForm& BKLeftNormalForm::operator= (const NF & bkl) [inline]`

Assignment operator.

Definition at line 91 of file BKLeftNormalForm.h.

References `triple< T1, T2, T3 >::first`, `triple< T1, T2, T3 >::second`, `theDecomposition`, `theOmegaPower`, `theRank`, and `triple< T1, T2, T3 >::third`.

**9.12.5.22** `BKLeftNormalForm& BKLeftNormalForm::operator= (const BKLeftNormalForm & bkl) [inline]`

Assignment operator.

Definition at line 83 of file BKLeftNormalForm.h.

References `theDecomposition`, `theOmegaPower`, and `theRank`.

**9.12.5.23** `bool BKLeftNormalForm::operator== (const BKLeftNormalForm & bkl) const`

**9.12.5.24** `void BKLeftNormalForm::setDecomposition (const list< Permutation > & d) [inline]`

Set the decomposition  $(\xi_1, \dots, \xi_m)$ . The result might be an incorrect form (not satisfying some greedy conditions of this BKL form).

Definition at line 171 of file BKLeftNormalForm.h.

References `theDecomposition`.

**9.12.5.25 void BKLeftNormalForm::setPower (int *p*) [inline]**

Set the power of  $\Delta$ . The result is always a correct normal form.

Definition at line 167 of file BKLeftNormalForm.h.

References theOmegaPower.

**9.12.5.26 static transformationResult BKLeftNormalForm::transform (int *theIndex*, Permutation & *p1*, Permutation & *p2*) [static, private]**

Main function for computing the normal form of the word.

**9.12.6 Member Data Documentation****9.12.6.1 list< Permutation > BKLeftNormalForm::theDecomposition [private]**

Sequence of permutations.

Definition at line 227 of file BKLeftNormalForm.h.

Referenced by getDecomposition(), isTrivial(), operator NF(), operator=(), and setDecomposition().

**9.12.6.2 int BKLeftNormalForm::theOmegaPower [private]**

Power of omega.

Definition at line 224 of file BKLeftNormalForm.h.

Referenced by getPower(), isTrivial(), operator NF(), operator=(), and setPower().

**9.12.6.3 int BKLeftNormalForm::theRank [private]**

The rank of the braid group (number of strands).

Definition at line 221 of file BKLeftNormalForm.h.

Referenced by operator NF(), and operator=().

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

- BraidGroup/include/BKLeftNormalForm.h

## 9.13 BKLRightNormalForm Class Reference

(Contains errors!!! Not to be used yet) Birman-Ko-Lee right normal form.

```
#include <BKLRightNormalForm.h>
```

### Public Types

- typedef **triple**< int, int, list< **Permutation** > > **NF**

*Presentation of a normal form.*

### Public Member Functions

- **BKLRightNormalForm** ()  
*Create trivial normal form.*
- **BKLRightNormalForm** (const **BKLRightNormalForm** &bkl)  
*Copy constructor.*
- **BKLRightNormalForm** (int rank, int power, const list< **Permutation** > &decomp)  
*Create normal form by its presentation (no check that given presentation is correct, use adjustDecomposition if the presentation is incorrect).*
- **BKLRightNormalForm** (const **BraidGroup** &G, const **Word** &w)  
*Create a normal form of a braid word.*
- **BKLRightNormalForm** & operator= (const **BKLRightNormalForm** &bkl)  
*Assignment operator.*
- **BKLRightNormalForm** & operator= (const **NF** &bkl)  
*Assignment operator.*
- **BKLRightNormalForm** operator- () const  
*Invert a normal form.*
- **BKLRightNormalForm** operator\* (const **BKLRightNormalForm** &bkl) const  
*Multiply two normal forms.*
- bool operator== (const **BKLRightNormalForm** &bkl) const

- bool **operator!=** (const **BKLRightNormalForm** &bkl) const
- bool **operator<** (const **BKLRightNormalForm** &bkl) const
- int **getPower** () const  
*Get the power of omega.*
- const list< **Permutation** > & **getDecomposition** () const  
*Get the decomposition.*
- **operator NF** () const  
*Get the presentation of the normal form.*
- bool **isTrivial** () const  
*Check if the normal form os trivial.*
- **Word** **getWord** () const  
*Return a word represented by the normal form.*
- void **setPower** (int p)  
*Set the power of  $\Delta$ . The result is always a correct normal form.*
- void **setDecomposition** (const list< **Permutation** > &d)  
*Set the decomposition  $(\xi_1, \dots, \xi_m)$ . The result might be an incorrect form (not satisfying some greedy conditions of this BKL form).*

## Static Public Member Functions

- static **Permutation** **getTinyTwistPermutation** (int theIndex)  
*Returns a cyclic permutation  $\delta = (1, 2, 3, \dots, n - 1, 0)$ .*
- static void **adjustDecomposition** (int rank, int &power, list< **Permutation** > &decomp)  
*Function adjusting the decomposition in a normal form.*

## Private Types

- enum **transformationResult** { **TWO\_MULTIPLIERS**, **ONE\_MULTIPLIER**, **NO\_CHANGE** }  
*The result of the transformation.*

## Private Member Functions

- **BKLRightNormalForm** **inverse** () const

*Invert the normal form.*

- **BKLRightNormalForm** **multiply** (const **BKLRightNormalForm** &bkl) const

*Multiply the normal form by another normal form on the right.*

## Static Private Member Functions

- static **transformationResult** **transform** (int theIndex, **Permutation** &p1, **Permutation** &p2)

*Main function for computing the normal form of the word.*

## Private Attributes

- int **theRank**

*The rank of the braid group (number of strands).*

- int **theOmegaPower**

*Power of omega.*

- list< **Permutation** > **theDecomposition**

*Sequence of permutations.*

### 9.13.1 Detailed Description

(Contains errors!!! Not to be used yet) Birman-Ko-Lee right normal form. The standard presentation for BKL is a pair  $(p, (\xi_1, \dots, \xi_m))$ , where  $p \in Z$  and each  $\xi_i \in S_n$  is a permutation.

Definition at line 37 of file BKLRightNormalForm.h.

## 9.13.2 Member Typedef Documentation

### 9.13.2.1 `typedef triple< int , int , list< Permutation > > BKLRightNormalForm::NF`

Presentation of a normal form. The first component specifies the rank of a braid group. The second component specifies the power of the twist. The third component specifies the list of braid permutations.

Definition at line 47 of file BKLRightNormalForm.h.

## 9.13.3 Member Enumeration Documentation

### 9.13.3.1 `enum BKLRightNormalForm::transformationResult [private]`

The result of the transformation.

**Enumerator:**

*TWO\_MULTIPLIERS*  
*ONE\_MULTIPLIER*  
*NO\_CHANGE*

Definition at line 193 of file BKLRightNormalForm.h.

## 9.13.4 Constructor & Destructor Documentation

### 9.13.4.1 `BKLRightNormalForm::BKLRightNormalForm () [inline]`

Create trivial normal form.

Definition at line 58 of file BKLRightNormalForm.h.

### 9.13.4.2 `BKLRightNormalForm::BKLRightNormalForm (const BKLRightNormalForm & bkl) [inline]`

Copy constructor.

Definition at line 66 of file BKLRightNormalForm.h.

### 9.13.4.3 `BKLRightNormalForm::BKLRightNormalForm (int rank, int power, const list< Permutation > & decomp) [inline]`

Create normal form by its presentation (no check that given presentation is correct, use adjustDecomposition if the presentation is incorrect).



Definition at line 76 of file BKLRightNormalForm.h.

#### 9.13.4.4 BKLRightNormalForm::BKLRightNormalForm (const BraidGroup & *G*, const Word & *w*)

Create a normal form of a braid word.

### 9.13.5 Member Function Documentation

#### 9.13.5.1 static void BKLRightNormalForm::adjustDecomposition (int *rank*, int & *power*, list< Permutation > & *decomp*) [static]

Function adjusting the decomposition in a normal form.

#### 9.13.5.2 const list< Permutation > & BKLRightNormalForm::getDecomposition () const [inline]

Get the decomposition.

Definition at line 138 of file BKLRightNormalForm.h.

References theDecomposition.

#### 9.13.5.3 int BKLRightNormalForm::getPower () const [inline]

Get the power of omega.

Definition at line 134 of file BKLRightNormalForm.h.

References theOmegaPower.

#### 9.13.5.4 static Permutation BKLRightNormalForm::getTinyTwistPermutation (int *theIndex*) [inline, static]

Returns a cyclic permutation  $\delta = (1, 2, 3, \dots, n - 1, 0)$ .

Definition at line 150 of file BKLRightNormalForm.h.

#### 9.13.5.5 Word BKLRightNormalForm::getWord () const

Return a word represented by the normal form.

#### 9.13.5.6 **BKLRightNormalForm BKLRightNormalForm::inverse () const** **[private]**

Invert the normal form.

Referenced by operator-().

#### 9.13.5.7 **bool BKLRightNormalForm::isTrivial () const** **[inline]**

Check if the normal form os trivial.

Definition at line 146 of file BKLRightNormalForm.h.

References theDecomposition, and theOmegaPower.

#### 9.13.5.8 **BKLRightNormalForm BKLRightNormalForm::multiply (const** **BKLRightNormalForm & *bkl*) const** **[private]**

Multiply the normal form by another normal form on the right.

Referenced by operator\*().

#### 9.13.5.9 **BKLRightNormalForm::operator NF () const** **[inline]**

Get the presentation of the normal form.

Definition at line 142 of file BKLRightNormalForm.h.

References theDecomposition, theOmegaPower, and theRank.

#### 9.13.5.10 **bool BKLRightNormalForm::operator!= (const** **BKLRightNormalForm & *bkl*) const**

#### 9.13.5.11 **BKLRightNormalForm BKLRightNormalForm::operator\* (const** **BKLRightNormalForm & *bkl*) const** **[inline]**

Multiply two normal forms.

Definition at line 117 of file BKLRightNormalForm.h.

References multiply().

#### 9.13.5.12 **BKLRightNormalForm BKLRightNormalForm::operator- () const** **[inline]**

Invert a normal form.

Definition at line 113 of file BKLRightNormalForm.h.

References `inverse()`.

**9.13.5.13** `bool BKLRightNormalForm::operator< (const BKLRightNormalForm & bkl) const`

**9.13.5.14** `BKLRightNormalForm& BKLRightNormalForm::operator= (const NF & bkl) [inline]`

Assignment operator.

Definition at line 106 of file BKLRightNormalForm.h.

References `triple< T1, T2, T3 >::first`, `triple< T1, T2, T3 >::second`, `theDecomposition`, `theOmegaPower`, `theRank`, and `triple< T1, T2, T3 >::third`.

**9.13.5.15** `BKLRightNormalForm& BKLRightNormalForm::operator= (const BKLRightNormalForm & bkl) [inline]`

Assignment operator.

Definition at line 99 of file BKLRightNormalForm.h.

References `theDecomposition`, `theOmegaPower`, and `theRank`.

**9.13.5.16** `bool BKLRightNormalForm::operator== (const BKLRightNormalForm & bkl) const`

**9.13.5.17** `void BKLRightNormalForm::setDecomposition (const list< Permutation > & d) [inline]`

Set the decomposition  $(\xi_1, \dots, \xi_m)$ . The result might be an incorrect form (not satisfying some greedy conditions of this BKL form).

Definition at line 182 of file BKLRightNormalForm.h.

References `theDecomposition`.

**9.13.5.18** `void BKLRightNormalForm::setPower (int p) [inline]`

Set the power of  $\Delta$ . The result is always a correct normal form.

Definition at line 179 of file BKLRightNormalForm.h.

References `theOmegaPower`.

#### 9.13.5.19 `static transformationResult BKLRightNormalForm::transform (int theIndex, Permutation & p1, Permutation & p2) [static, private]`

Main function for computing the normal form of the word.

### 9.13.6 Member Data Documentation

#### 9.13.6.1 `list< Permutation > BKLRightNormalForm::theDecomposition [private]`

Sequence of permutations.

Definition at line 223 of file `BKLRightNormalForm.h`.

Referenced by `getDecomposition()`, `isTrivial()`, `operator NF()`, `operator=()`, and `setDecomposition()`.

#### 9.13.6.2 `int BKLRightNormalForm::theOmegaPower [private]`

Power of omega.

Definition at line 219 of file `BKLRightNormalForm.h`.

Referenced by `getPower()`, `isTrivial()`, `operator NF()`, `operator=()`, and `setPower()`.

#### 9.13.6.3 `int BKLRightNormalForm::theRank [private]`

The rank of the braid group (number of strands).

Definition at line 215 of file `BKLRightNormalForm.h`.

Referenced by `operator NF()`, and `operator=()`.

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

- `BraidGroup/include/BKLRightNormalForm.h`

## 9.14 BraidDrawPDF Class Reference

Class for drawing a braid on n strands as a square table.

```
#include <WordDraw.h>
```

### Public Member Functions

- **BraidDrawPDF** (int n, string t=string())  
*Constructor.*
- **~BraidDrawPDF** ()
- void **draw** (const **Word** &w)  
*Draw a braid represented by the word w.*
- void **save** (const char \*f)  
*Write the picture into a file.*
- void **setSS** (int new\_ss)  
*Set the size of the square cells of the table.*

### Private Member Functions

- void **drawGrid** (int nP, int sN)
- int **tableWidth** () const
- int **stripPosition** (int i)
- void **drawCompressedBraid** (const **Word** &theWord, int vert\_offset=0)
- int **posInPage** () const
- int **stripesInPage** () const
- void **drawGenerator** (**Generator** g, int pos, int vert\_offset)
- pair< int, int > **getPageStrip** (int pos) const

### Private Attributes

- int **N**
- **PDFStructure pdf\_out**
- int **ss**
- int **theLength**
- int **betBraids**
- bool **useCircle**
- string **theTitle**

### 9.14.1 Detailed Description

Class for drawing a braid on  $n$  strands as a square table.

Definition at line 188 of file WordDraw.h.

### 9.14.2 Constructor & Destructor Documentation

#### 9.14.2.1 BraidDrawPDF::BraidDrawPDF (int $n$ , string $t = \text{string}()$ ) [inline]

Constructor.

##### Parameters:

$n$  - number of generators (i.e. number of strands -1 )

$t$  - optional title which will appear on each page

Definition at line 197 of file WordDraw.h.

#### 9.14.2.2 BraidDrawPDF::~~BraidDrawPDF () [inline]

Definition at line 208 of file WordDraw.h.

### 9.14.3 Member Function Documentation

#### 9.14.3.1 void BraidDrawPDF::draw (const Word & $w$ ) [inline]

Draw a braid represented by the word  $w$ .

Definition at line 211 of file WordDraw.h.

References drawCompressedBraid(), PDFStructure::newPage(), and pdf\_out.

#### 9.14.3.2 void BraidDrawPDF::drawCompressedBraid (const Word & $theWord$ , int $vert\_offset = 0$ ) [inline, private]

Definition at line 254 of file WordDraw.h.

References Word::begin(), drawGenerator(), drawGrid(), Word::end(),  $N$ , PDFStructure::newPage(), pdf\_out, posInPage(), stripesInPage(), and theLength.

Referenced by draw().

**9.14.3.3 void BraidDrawPDF::drawGenerator (Generator *g*, int *pos*, int *vert\_offset*) [inline, private]**

Definition at line 296 of file WordDraw.h.

References PDFStructure::addObject(), getPageStrip(), PDFPage::lMargin(), pdf\_out, posInPage(), ss, stripPosition(), and useCircle.

Referenced by drawCompressedBraid().

**9.14.3.4 void BraidDrawPDF::drawGrid (int *nP*, int *sN*) [inline, private]**

Definition at line 230 of file WordDraw.h.

References PDFStructure::addObject(), PDFPage::lMargin(), PDFPage::mWidth(), pdf\_out, ss, stripPosition(), tableWidth(), theTitle, and PDFPage::tMargin().

Referenced by drawCompressedBraid().

**9.14.3.5 pair<int,int> BraidDrawPDF::getPageStrip (int *pos*) const [inline, private]**

Definition at line 319 of file WordDraw.h.

References posInPage(), and stripesInPage().

Referenced by drawGenerator().

**9.14.3.6 int BraidDrawPDF::posInPage () const [inline, private]**

Definition at line 293 of file WordDraw.h.

References PDFPage::mWidth(), and ss.

Referenced by drawCompressedBraid(), drawGenerator(), and getPageStrip().

**9.14.3.7 void BraidDrawPDF::save (const char \**f*) [inline]**

Write the picture into a file.

Definition at line 220 of file WordDraw.h.

References pdf\_out, and PDFStructure::save().

#### 9.14.3.8 void BraidDrawPDF::setSS (int *new\_ss*) [inline]

Set the size of the square cells of the table.

Definition at line 225 of file WordDraw.h.

References ss.

#### 9.14.3.9 int BraidDrawPDF::stripesInPage () const [inline, private]

Definition at line 294 of file WordDraw.h.

References PDFPage::mHeight(), N, and ss.

Referenced by drawCompressedBraid(), and getPageStrip().

#### 9.14.3.10 int BraidDrawPDF::stripPosition (int *i*) [inline, private]

Definition at line 247 of file WordDraw.h.

References ss, tableWidth(), theTitle, and PDFPage::tMargin().

Referenced by drawGenerator(), and drawGrid().

#### 9.14.3.11 int BraidDrawPDF::tableWidth () const [inline, private]

Definition at line 245 of file WordDraw.h.

References N, and ss.

Referenced by drawGrid(), and stripPosition().

### 9.14.4 Member Data Documentation

#### 9.14.4.1 int BraidDrawPDF::betBraids [private]

Definition at line 334 of file WordDraw.h.

#### 9.14.4.2 int BraidDrawPDF::N [private]

Definition at line 330 of file WordDraw.h.

Referenced by drawCompressedBraid(), stripesInPage(), and tableWidth().



**9.14.4.3 PDFStructure BraidDrawPDF::pdf\_out [private]**

Definition at line 331 of file WordDraw.h.

Referenced by draw(), drawCompressedBraid(), drawGenerator(), drawGrid(), and save().

**9.14.4.4 int BraidDrawPDF::ss [private]**

Definition at line 332 of file WordDraw.h.

Referenced by drawGenerator(), drawGrid(), posInPage(), setSS(), stripesInPage(), stripPosition(), and tableWidth().

**9.14.4.5 int BraidDrawPDF::theLength [private]**

Definition at line 333 of file WordDraw.h.

Referenced by drawCompressedBraid().

**9.14.4.6 string BraidDrawPDF::theTitle [private]**

Definition at line 336 of file WordDraw.h.

Referenced by drawGrid(), and stripPosition().

**9.14.4.7 bool BraidDrawPDF::useCircle [private]**

Definition at line 335 of file WordDraw.h.

Referenced by drawGenerator().

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

- Graphics/include/**WordDraw.h**

## 9.15 BraidGroup Class Reference

Class **BraidGroup** (p. 140) (defines a representation of a Braid Group)//.

```
#include <BraidGroup.h>
```

### Public Member Functions

- **BraidGroup** (int n)  
*Constructor (creates the braid group on n strands).*
- int **getRank** () const  
*get the rank of the braid group (number of strands)*
- **Word** **twist** (const **Word** &w) const  
*get the braid word called a half-twist (its square generates the center of the braid group)*

### Static Public Member Functions

- static list< **Word** > **getBraidRelators** (int N)  
*Get a list (non-symmetrized) of braid relators.*

### Private Attributes

- int **theRank**

#### 9.15.1 Detailed Description

Class **BraidGroup** (p. 140) (defines a representation of a Braid Group)//. A **Braid-Group** (p. 140) is uniquely defined by its index.

Definition at line 26 of file BraidGroup.h.

#### 9.15.2 Constructor & Destructor Documentation

##### 9.15.2.1 BraidGroup::BraidGroup (int *n*) [inline]

Constructor (creates the braid group on n strands).

Definition at line 36 of file BraidGroup.h.

### 9.15.3 Member Function Documentation

#### 9.15.3.1 `static list< Word > BraidGroup::getBraidRelators (int $N$ )` `[static]`

Get a list (non-symmetrized) of braid relators.

#### 9.15.3.2 `int BraidGroup::getRank () const` `[inline]`

get the rank of the braid group (number of strands)

Definition at line 47 of file BraidGroup.h.

References `theRank`.

#### 9.15.3.3 `Word BraidGroup::twist (const Word & $w$ ) const`

get the braid word called a half-twist (its square generates the center of the braid group)

### 9.15.4 Member Data Documentation

#### 9.15.4.1 `int BraidGroup::theRank` `[private]`

Definition at line 63 of file BraidGroup.h.

Referenced by `getRank()`.

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

- BraidGroup/include/BraidGroup.h

## 9.16 BraidNode Struct Reference

Defines a crossing in a linked braid structure//.

```
#include <LinkedBraidStructure.h>
```

### Public Member Functions

- **BraidNode** (int num=0, bool tp=true, **BraidNode** \*l=NULL, **BraidNode** \*a=NULL, **BraidNode** \*r=NULL, **BraidNode** \*bl=NULL, **BraidNode** \*b=NULL, **BraidNode** \*br=NULL)
- **BraidNode** (bool m=false)

*Default constructor (creates a disconnected node - all pointers are NULL).*

### Static Public Member Functions

- static bool **isHandle** (vector< **BraidNode** \* > &backNodes, int g)
- static void **removeHandle** (vector< **BraidNode** \* > &frontNodes, vector< **BraidNode** \* > &backNodes, int g, list< int > &processedHandle)
- static void **removeNodeTerminalClosure** (vector< **BraidNode** \* > &frontNodes, vector< **BraidNode** \* > &backNodes, int g, **BraidNode** \*node, list< int > &removedClosure)
- static void **removeTerminalNode** (vector< **BraidNode** \* > &frontNodes, vector< **BraidNode** \* > &backNodes, int g, **BraidNode** \*node, list< int > &removedClosure)

### Public Attributes

- **BraidNode** \* **left**  
*pointer to the ahead-left node*
- **BraidNode** \* **ahead**  
*pointer to the ahead node*
- **BraidNode** \* **right**  
*pointer to the ahead-right node*
- **BraidNode** \* **back\_left**  
*pointer to the back-left node*
- **BraidNode** \* **back**

*pointer to the back node*

- **BraidNode \* back\_right**

*pointer to the back-right node*

- **bool type**

*crossing orientation (true=positive, false=negative)*

- **BraidNode \* link**

- **int weight**

- **int theNumber**

- **bool marked**

*Auxiliary flag. Used in `LinkedBraidStructure::getWord( )` (p.307) to mark used crossings.*

### 9.16.1 Detailed Description

Defines a crossing in a linked braid structure//. Represented by the orientation (type) of the crossing and 6 pointers to other crossings

Definition at line 20 of file LinkedBraidStructure.h.

### 9.16.2 Constructor & Destructor Documentation

**9.16.2.1 BraidNode::BraidNode (int num = 0, bool tp = true, BraidNode \* l = NULL, BraidNode \* a = NULL, BraidNode \* r = NULL, BraidNode \* bl = NULL, BraidNode \* b = NULL, BraidNode \* br = NULL) [inline]**

Definition at line 31 of file LinkedBraidStructure.h.

**9.16.2.2 BraidNode::BraidNode (bool m = false) [inline]**

Default constructor (creates a disconnected node - all pointers are NULL).

Definition at line 36 of file LinkedBraidStructure\_old.h.

### 9.16.3 Member Function Documentation

- 9.16.3.1** `static bool BraidNode::isHandle (vector< BraidNode * > & backNodes, int g) [static]`
- 9.16.3.2** `static void BraidNode::removeHandle (vector< BraidNode * > & frontNodes, vector< BraidNode * > & backNodes, int g, list< int > & processedHandle) [static]`
- 9.16.3.3** `static void BraidNode::removeNodeTerminalClosure (vector< BraidNode * > & frontNodes, vector< BraidNode * > & backNodes, int g, BraidNode * node, list< int > & removedClosure) [static]`
- 9.16.3.4** `static void BraidNode::removeTerminalNode (vector< BraidNode * > & frontNodes, vector< BraidNode * > & backNodes, int g, BraidNode * node, list< int > & removedClosure) [static]`

### 9.16.4 Member Data Documentation

#### 9.16.4.1 `BraidNode * BraidNode::ahead`

pointer to the ahead node if  $w = w_1 \circ x_i \circ w_2$  and the node corresponds to  $x_i$  then ahead points to the rightmost  $x_i$  in  $w_1$ .

Definition at line 46 of file LinkedBraidStructure.h.

#### 9.16.4.2 `BraidNode * BraidNode::back`

pointer to the back node if  $w = w_1 \circ x_i \circ w_2$  and the node corresponds to  $x_i$  then ahead points to the leftmost  $x_i$  in  $w_2$ .

Definition at line 50 of file LinkedBraidStructure.h.

#### 9.16.4.3 `BraidNode * BraidNode::back_left`

pointer to the back-left node if  $w = w_1 \circ x_i \circ w_2$  and the node corresponds to  $x_i$  then back\_left points to the leftmost  $x_{i-1}$  in  $w_2$ . Except one case! If between  $x_i$  and  $x_{i-1}$  there is another  $x_i$  then back\_left = NULL.

Definition at line 49 of file LinkedBraidStructure.h.

#### 9.16.4.4 `BraidNode * BraidNode::back_right`

pointer to the back-right node if  $w = w_1 \circ x_i \circ w_2$  and the node corresponds to  $x_i$  then back\_right points to the leftmost  $x_{i+1}$  in  $w_2$ . Except one case! If between  $x_i$  and  $x_{i+1}$

there is another  $x_i$  then `back_right = NULL`.

Definition at line 51 of file `LinkedBraidStructure.h`.

#### 9.16.4.5 `BraidNode * BraidNode::left`

pointer to the ahead-left node if  $w = w_1 \circ x_i \circ w_2$  and the node corresponds to  $x_i$  then `left` points to the rightmost  $x_{i-1}$  in  $w_1$ . Except one case! If between  $x_{i-1}$  and  $x_i$  there is another  $x_i$  then `left = NULL`.

Definition at line 45 of file `LinkedBraidStructure.h`.

#### 9.16.4.6 `BraidNode* BraidNode::link [mutable]`

Definition at line 56 of file `LinkedBraidStructure.h`.

#### 9.16.4.7 `bool BraidNode::marked [mutable]`

Auxiliary flag. Used in `LinkedBraidStructure::getWord( )` (p.307) to mark used crossings.

Definition at line 108 of file `LinkedBraidStructure_old.h`.

#### 9.16.4.8 `BraidNode * BraidNode::right`

pointer to the ahead-right node if  $w = w_1 \circ x_i \circ w_2$  and the node corresponds to  $x_i$  then `right` points to the rightmost  $x_{i+1}$  in  $w_1$ . Except one case! If between  $x_{i+1}$  and  $x_i$  there is another  $x_i$  then `right = NULL`.

Definition at line 47 of file `LinkedBraidStructure.h`.

#### 9.16.4.9 `int BraidNode::theNumber`

Definition at line 62 of file `LinkedBraidStructure.h`.

#### 9.16.4.10 `bool BraidNode::type`

crossing orientation (true=positive, false=negative)

Definition at line 53 of file `LinkedBraidStructure.h`.

#### 9.16.4.11 `int BraidNode::weight [mutable]`

Definition at line 59 of file `LinkedBraidStructure.h`.

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

- BraidGroup/include/**LinkedBraidStructure.h**
- BraidGroup/include/**LinkedBraidStructure\_old.h**



## 9.17 BSets Class Reference

Implements construction of the initial commuting sets of subgroup generators.

```
#include <AEProtocol.h>
```

### Static Public Member Functions

- static **BSets generateRandom** (int *N*)  
*Generates two sets of commuting generators by splitting the original  $N$  generators in 2, generators are taken randomly.*
- static **BSets generateEqual** (int *N*)  
*Generates two sets of commuting generators by separating first  $(N-2)/2$  generators from  $(N-2)/2$  last ones.*

### Public Attributes

- vector< **Word** > **BL**
- vector< **Word** > **BR**

### Friends

- ostream & **operator**<< (ostream &out, const **BSets** &bs)  
*Prints  $B$  sets.*

#### 9.17.1 Detailed Description

Implements construction of the initial commuting sets of subgroup generators.

Definition at line 75 of file AEProtocol.h.

#### 9.17.2 Member Function Documentation

##### 9.17.2.1 static BSets BSets::generateEqual (int *N*) [static]

Generates two sets of commuting generators by separating first  $(N-2)/2$  generators from  $(N-2)/2$  last ones.

**9.17.2.2 static BSets BSets::generateRandom (int *N*) [static]**

Generates two sets of commuting generators by splitting the original *N* generators in 2, generators are taken randomly.

**9.17.3 Friends And Related Function Documentation****9.17.3.1 ostream& operator<< (ostream & *out*, const BSets & *bs*) [friend]**

Prints B sets.

Definition at line 87 of file AEProtocol.h.

**9.17.4 Member Data Documentation****9.17.4.1 vector<Word> BSets::BL**

Definition at line 78 of file AEProtocol.h.

**9.17.4.2 vector<Word> BSets::BR**

Definition at line 79 of file AEProtocol.h.

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

- CryptoAE/include/AEProtocol.h

## 9.18 BalancedTree< Obj >::BTNode Struct Reference

### Public Member Functions

- **BTNode** ()
- **BTNode** (Obj obj)
- **~BTNode** ()

### Public Attributes

- Obj **theObject**
- **BTNode** \* **subtree1**
- int **height1**
- int **weight1**
- **BTNode** \* **subtree2**
- int **height2**
- int **weight2**

### 9.18.1 Detailed Description

**template<class Obj> struct BalancedTree< Obj >::BTNode**

Definition at line 44 of file BalancedTree.h.

### 9.18.2 Constructor & Destructor Documentation

**9.18.2.1** **template<class Obj > BalancedTree< Obj >::BTNode::BTNode ()**  
[inline]

Definition at line 45 of file BalancedTree.h.

**9.18.2.2** **template<class Obj > BalancedTree< Obj >::BTNode::BTNode (Obj**  
**obj)** [inline]

Definition at line 46 of file BalancedTree.h.

**9.18.2.3** **template<class Obj > BalancedTree< Obj >::BTNode::~~BTNode ()**  
[inline]

Definition at line 47 of file BalancedTree.h.

References `BalancedTree< Obj >::BTNode::subtree1`, and `BalancedTree< Obj >::BTNode::subtree2`.

### 9.18.3 Member Data Documentation

#### 9.18.3.1 `template<class Obj> int BalancedTree< Obj >::BTNode::height1`

Definition at line 53 of file `BalancedTree.h`.

Referenced by `BalancedTree< Obj >::insert()`, `BalancedTree< Obj >::rotateLeft()`, and `BalancedTree< Obj >::rotateRight()`.

#### 9.18.3.2 `template<class Obj> int BalancedTree< Obj >::BTNode::height2`

Definition at line 54 of file `BalancedTree.h`.

Referenced by `BalancedTree< Obj >::insert()`, `BalancedTree< Obj >::rotateLeft()`, and `BalancedTree< Obj >::rotateRight()`.

#### 9.18.3.3 `template<class Obj> BTNode* BalancedTree< Obj >::BTNode::subtree1`

Definition at line 53 of file `BalancedTree.h`.

Referenced by `BalancedTree< Obj >::getList()`, `BalancedTree< Obj >::insert()`, `BalancedTree< Obj >::rotateLeft()`, `BalancedTree< Obj >::rotateRight()`, and `BalancedTree< Obj >::BTNode::~~BTNode()`.

#### 9.18.3.4 `template<class Obj> BTNode* BalancedTree< Obj >::BTNode::subtree2`

Definition at line 54 of file `BalancedTree.h`.

Referenced by `BalancedTree< Obj >::getList()`, `BalancedTree< Obj >::insert()`, `BalancedTree< Obj >::rotateLeft()`, `BalancedTree< Obj >::rotateRight()`, and `BalancedTree< Obj >::BTNode::~~BTNode()`.

#### 9.18.3.5 `template<class Obj> Obj BalancedTree< Obj >::BTNode::theObject`

Definition at line 52 of file `BalancedTree.h`.

Referenced by `BalancedTree< Obj >::getList()`.

**9.18.3.6 `template<class Obj > int BalancedTree< Obj >::BTNode::weight1`**

Definition at line 53 of file `BalancedTree.h`.

Referenced by `BalancedTree< Obj >::insert()`, `BalancedTree< Obj >::rotateLeft()`, `BalancedTree< Obj >::rotateRight()`, and `BalancedTree< Obj >::size()`.

**9.18.3.7 `template<class Obj > int BalancedTree< Obj >::BTNode::weight2`**

Definition at line 54 of file `BalancedTree.h`.

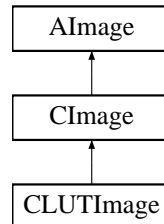
Referenced by `BalancedTree< Obj >::insert()`, `BalancedTree< Obj >::rotateLeft()`, `BalancedTree< Obj >::rotateRight()`, and `BalancedTree< Obj >::size()`.

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

- `general/include/BalancedTree.h`

## 9.19 CImage Class Reference

#include <AImage.h> Inheritance diagram for CImage::



### Public Member Functions

- **CImage** ()
- **CImage** (int w, int h)
- **CImage** (const string &in\_file\_name, **FILE\_TYPE** ft=PPM)
- **CImage** (const **CImage** &image)
- **CImage** (const **CImage** &i, int p1, int p2, int p3, int p4, bool p5)
- void **setRedPixel** (int i, int j, int v)
- void **setRedPixel** (int n, int v)
- void **setBluePixel** (int i, int j, int v)
- void **setBluePixel** (int n, int v)
- void **setGreenPixel** (int i, int j, int v)
- void **setGreenPixel** (int n, int v)
- void **setRedPixelClipped** (int n, int v)
- void **setRedPixelClipped** (int i, int j, int v)
- void **setBluePixelClipped** (int n, int v)
- void **setBluePixelClipped** (int i, int j, int v)
- void **setGreenPixelClipped** (int n, int v)
- void **setGreenPixelClipped** (int i, int j, int v)
- virtual unsigned char **getRedPixel** (int i, int j) const
- virtual unsigned char **getRedPixel** (int n) const
- virtual unsigned char **getBluePixel** (int i, int j) const
- virtual unsigned char **getBluePixel** (int n) const
- virtual unsigned char **getGreenPixel** (int i, int j) const
- virtual unsigned char **getGreenPixel** (int n) const
- const **GRIImage** & **getRedImage** () const
- const **GRIImage** & **getGreenImage** () const
- const **GRIImage** & **getBlueImage** () const
- void **saveTo** (const string &file\_name)
- **IMAGE\_TYPE** **getType** () const

## Protected Member Functions

- **CImage & operator=** (const **CImage** &)
- void **printOnPPM** (ostream &out) const
- void **readFromPPM** (istream &in)

## Protected Attributes

- **GRIImage** redImage
- **GRIImage** blueImage
- **GRIImage** greenImage

## Friends

- class **CLUTImage**

### 9.19.1 Detailed Description

Definition at line 237 of file AImage.h.

### 9.19.2 Constructor & Destructor Documentation

#### 9.19.2.1 CImage::CImage () [inline]

Definition at line 240 of file AImage.h.

#### 9.19.2.2 CImage::CImage (int *w*, int *h*) [inline]

Definition at line 243 of file AImage.h.

#### 9.19.2.3 CImage::CImage (const string & *in\_file\_name*, FILE\_TYPE *ft* = PPM)

#### 9.19.2.4 CImage::CImage (const CImage & *image*) [inline]

Definition at line 249 of file AImage.h.

#### 9.19.2.5 CImage::CImage (const CImage & *i*, int *p1*, int *p2*, int *p3*, int *p4*, bool *p5*) [inline]

Definition at line 256 of file AImage.h.

### 9.19.3 Member Function Documentation

#### 9.19.3.1 `const GRImage& CImage::getBlueImage () const [inline]`

Definition at line 299 of file AImage.h.

References blueImage.

#### 9.19.3.2 `virtual unsigned char CImage::getBluePixel (int n) const [inline, virtual]`

Reimplemented in **CLUTImage** (p. 160).

Definition at line 290 of file AImage.h.

References blueImage, and GRImage::getPixel().

#### 9.19.3.3 `virtual unsigned char CImage::getBluePixel (int i, int j) const [inline, virtual]`

Reimplemented in **CLUTImage** (p. 160).

Definition at line 289 of file AImage.h.

References blueImage, and GRImage::getPixel().

Referenced by CLUTImage::getBluePixel().

#### 9.19.3.4 `const GRImage& CImage::getGreenImage () const [inline]`

Definition at line 297 of file AImage.h.

References greenImage.

#### 9.19.3.5 `virtual unsigned char CImage::getGreenPixel (int n) const [inline, virtual]`

Reimplemented in **CLUTImage** (p. 160).

Definition at line 293 of file AImage.h.

References GRImage::getPixel(), and greenImage.

#### 9.19.3.6 `virtual unsigned char CImage::getGreenPixel (int i, int j) const [inline, virtual]`

Reimplemented in **CLUTImage** (p. 160).



Definition at line 292 of file AImage.h.

References GRImage::getPixel(), and greenImage.

Referenced by CLUTImage::getGreenPixel().

#### 9.19.3.7 const GRImage& CImage::getRedImage () const [inline]

Definition at line 295 of file AImage.h.

References redImage.

#### 9.19.3.8 virtual unsigned char CImage::getRedPixel (int *n*) const [inline, virtual]

Reimplemented in **CLUTImage** (p. 161).

Definition at line 287 of file AImage.h.

References GRImage::getPixel(), and redImage.

#### 9.19.3.9 virtual unsigned char CImage::getRedPixel (int *i*, int *j*) const [inline, virtual]

Reimplemented in **CLUTImage** (p. 161).

Definition at line 286 of file AImage.h.

References GRImage::getPixel(), and redImage.

Referenced by CLUTImage::getRedPixel().

#### 9.19.3.10 IMAGE\_TYPE CImage::getType () const [inline, virtual]

Implements **AImage** (p. 101).

Definition at line 305 of file AImage.h.

References COLOR.

**9.19.3.11** `CImage& CImage::operator= (const CImage &) [protected]`

**9.19.3.12** `void CImage::printOnPPM (ostream & out) const [protected]`

**9.19.3.13** `void CImage::readFromPPM (istream & in) [protected]`

**9.19.3.14** `void CImage::saveTo (const string & file_name) [virtual]`

Implements **AImage** (p. 102).

Referenced by WordDraw::saveTo().

**9.19.3.15** `void CImage::setBluePixel (int n, int v) [inline]`

Definition at line 270 of file AImage.h.

References blueImage, and GRImage::setPixel().

**9.19.3.16** `void CImage::setBluePixel (int i, int j, int v) [inline]`

Definition at line 269 of file AImage.h.

References blueImage, and GRImage::setPixel().

Referenced by WordDraw::drawGenerator(), WordDraw::drawHorizontalGrid(), WordDraw::drawVerticalGrid(), and WordDraw::WordDraw().

**9.19.3.17** `void CImage::setBluePixelClipped (int i, int j, int v) [inline]`

Definition at line 279 of file AImage.h.

References blueImage, and GRImage::setPixelClipped().

**9.19.3.18** `void CImage::setBluePixelClipped (int n, int v) [inline]`

Definition at line 278 of file AImage.h.

References blueImage, and GRImage::setPixelClipped().

**9.19.3.19** `void CImage::setGreenPixel (int n, int v) [inline]`

Definition at line 273 of file AImage.h.

References greenImage, and GRImage::setPixel().

**9.19.3.20 void CImage::setGreenPixel (int *i*, int *j*, int *v*) [inline]**

Definition at line 272 of file AImage.h.

References greenImage, and GRImage::setPixel().

Referenced by WordDraw::drawGenerator(), WordDraw::drawHorizontalGrid(), WordDraw::drawVerticalGrid(), and WordDraw::WordDraw().

**9.19.3.21 void CImage::setGreenPixelClipped (int *i*, int *j*, int *v*) [inline]**

Definition at line 282 of file AImage.h.

References greenImage, and GRImage::setPixelClipped().

**9.19.3.22 void CImage::setGreenPixelClipped (int *n*, int *v*) [inline]**

Definition at line 281 of file AImage.h.

References greenImage, and GRImage::setPixelClipped().

**9.19.3.23 void CImage::setRedPixel (int *n*, int *v*) [inline]**

Definition at line 267 of file AImage.h.

References redImage, and GRImage::setPixel().

**9.19.3.24 void CImage::setRedPixel (int *i*, int *j*, int *v*) [inline]**

Definition at line 266 of file AImage.h.

References redImage, and GRImage::setPixel().

Referenced by WordDraw::drawGenerator(), WordDraw::drawHorizontalGrid(), WordDraw::drawVerticalGrid(), and WordDraw::WordDraw().

**9.19.3.25 void CImage::setRedPixelClipped (int *i*, int *j*, int *v*) [inline]**

Definition at line 276 of file AImage.h.

References redImage, and GRImage::setPixelClipped().

**9.19.3.26 void CImage::setRedPixelClipped (int *n*, int *v*) [inline]**

Definition at line 275 of file AImage.h.

References redImage, and GRImage::setPixelCliped().

## 9.19.4 Friends And Related Function Documentation

### 9.19.4.1 friend class CLUTImage [friend]

Definition at line 310 of file AImage.h.

## 9.19.5 Member Data Documentation

### 9.19.5.1 GRImage CImage::blueImage [protected]

Definition at line 321 of file AImage.h.

Referenced by getBlueImage(), getBluePixel(), setBluePixel(), and setBluePixelCliped().

### 9.19.5.2 GRImage CImage::greenImage [protected]

Definition at line 322 of file AImage.h.

Referenced by getGreenImage(), getGreenPixel(), setGreenPixel(), and setGreenPixelCliped().

### 9.19.5.3 GRImage CImage::redImage [protected]

Definition at line 320 of file AImage.h.

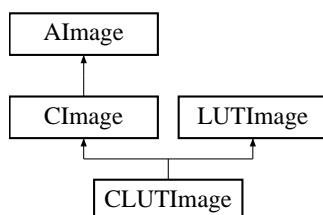
Referenced by getRedImage(), getRedPixel(), setRedPixel(), and setRedPixelCliped().

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

- Graphics/include/AImage.h

## 9.20 CLUTImage Class Reference

#include <AImage.h> Inheritance diagram for CLUTImage::



### Public Member Functions

- **CLUTImage** (const **CLUTImage** &li)
- **CLUTImage** ()
- **CLUTImage** (const **CImage** &i)
- **CLUTImage** (int w, int h)
- virtual unsigned char **getRedPixel** (int i, int j) const
- virtual unsigned char **getRedPixel** (int n) const
- virtual unsigned char **getBluePixel** (int i, int j) const
- virtual unsigned char **getBluePixel** (int n) const
- virtual unsigned char **getGreenPixel** (int i, int j) const
- virtual unsigned char **getGreenPixel** (int n) const

#### 9.20.1 Detailed Description

Definition at line 328 of file AImage.h.

#### 9.20.2 Constructor & Destructor Documentation

##### 9.20.2.1 CLUTImage::CLUTImage (const CLUTImage &li) [inline]

Definition at line 332 of file AImage.h.

##### 9.20.2.2 CLUTImage::CLUTImage () [inline]

Definition at line 334 of file AImage.h.

**9.20.2.3 CLUTImage::CLUTImage (const CImage & *i*) [inline]**

Definition at line 337 of file AImage.h.

**9.20.2.4 CLUTImage::CLUTImage (int *w*, int *h*) [inline]**

Definition at line 340 of file AImage.h.

**9.20.3 Member Function Documentation****9.20.3.1 virtual unsigned char CLUTImage::getBluePixel (int *n*) const [inline, virtual]**

Reimplemented from **CImage** (p. 154).

Definition at line 361 of file AImage.h.

References `LookUpTable::get()`, `CImage::getBluePixel()`, and `LUTImage::theLookUpTable`.

**9.20.3.2 virtual unsigned char CLUTImage::getBluePixel (int *i*, int *j*) const [inline, virtual]**

Reimplemented from **CImage** (p. 154).

Definition at line 358 of file AImage.h.

References `LookUpTable::get()`, `CImage::getBluePixel()`, and `LUTImage::theLookUpTable`.

**9.20.3.3 virtual unsigned char CLUTImage::getGreenPixel (int *n*) const [inline, virtual]**

Reimplemented from **CImage** (p. 154).

Definition at line 367 of file AImage.h.

References `LookUpTable::get()`, `CImage::getGreenPixel()`, and `LUTImage::theLookUpTable`.

**9.20.3.4 virtual unsigned char CLUTImage::getGreenPixel (int *i*, int *j*) const [inline, virtual]**

Reimplemented from **CImage** (p. 154).

Definition at line 364 of file AImage.h.

References `LookUpTable::get()`, `CImage::getGreenPixel()`, and `LUTImage::theLookUpTable`.

#### 9.20.3.5 `virtual unsigned char CLUTImage::getRedPixel (int n) const [inline, virtual]`

Reimplemented from **CImage** (p. 155).

Definition at line 354 of file AImage.h.

References `LookUpTable::get()`, `CImage::getRedPixel()`, and `LUTImage::theLookUpTable`.

#### 9.20.3.6 `virtual unsigned char CLUTImage::getRedPixel (int i, int j) const [inline, virtual]`

Reimplemented from **CImage** (p. 155).

Definition at line 351 of file AImage.h.

References `LookUpTable::get()`, `CImage::getRedPixel()`, and `LUTImage::theLookUpTable`.

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

- Graphics/include/AImage.h

## 9.21 PC::Col Struct Reference

```
#include <SignMatrix.h>
```

### Public Member Functions

- `Col()`

### Public Attributes

- unsigned int **colid**

### Static Public Attributes

- static const unsigned int **UNDEFINED** = ~0

#### 9.21.1 Detailed Description

Definition at line 25 of file SignMatrix.h.

#### 9.21.2 Constructor & Destructor Documentation

##### 9.21.2.1 PC::Col::Col() [inline]

Definition at line 25 of file SignMatrix.h.

#### 9.21.3 Member Data Documentation

##### 9.21.3.1 unsigned int PC::Col::colid

Definition at line 25 of file SignMatrix.h.

Referenced by `PC::SignMatrix< RowHdr, ColHdr, INTERNAL_TYPE >::addToCol()`, `PC::SignMatrix< RowHdr, ColHdr, INTERNAL_TYPE >::colHdr()`, `PC::SignMatrix< RowHdr, ColHdr, INTERNAL_TYPE >::columnsEqual()`, `PC::SignMatrix< RowHdr, ColHdr, INTERNAL_TYPE >::deleteCol()`, `PC::SignMatrix< RowHdr, ColHdr, INTERNAL_TYPE >::getCol()`, `PC::SignMatrix< RowHdr, ColHdr, INTERNAL_TYPE >::getColNumber()`, `PC::SignMatrix< RowHdr, ColHdr, INTERNAL_TYPE >::insertCol()`, `PC::SignMatrix< RowHdr, ColHdr, INTERNAL_TYPE >::insertColCopy()`, `PC::SignMatrix< RowHdr, ColHdr, INTERNAL_TYPE >::insertColSum()`,



PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::insertIntersectionCol(),  
PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::insertInvCol(),  
PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::insertUnitCol(),  
PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::insertZeroCol(),  
PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::moveCol(),  
PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::operator()(), and  
PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::setEntry().

### 9.21.3.2 `const unsigned int PC::Col::UNDEFINED = ~0` [static]

Definition at line 25 of file SignMatrix.h.

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

- HigmanGroup/include/**SignMatrix.h**

## 9.22 PC::PowerCircuitCompMatrix::ColHdr Struct Reference

### Public Attributes

- `std::list< Marking > markings`

### 9.22.1 Detailed Description

Definition at line 29 of file PowerCircuitCompMatrix.h.

### 9.22.2 Member Data Documentation

#### 9.22.2.1 `std::list<Marking>` `PC::PowerCircuitCompMatrix::ColHdr::markings`

Definition at line 31 of file PowerCircuitCompMatrix.h.

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

- `HigmanGroup/include/PowerCircuitCompMatrix.h`

## 9.23 CommDivider Struct Reference

```
#include <DCBraidReduction.h>
```

### Public Attributes

- int **length**
- int **begin**
- int **end**
- int **gen**

### Friends

- ostream & **operator**<< (ostream &out, const **CommDivider** &cd)

#### 9.23.1 Detailed Description

Definition at line 57 of file DCBraidReduction.h.

#### 9.23.2 Friends And Related Function Documentation

##### 9.23.2.1 ostream& operator<< (ostream & out, const CommDivider & cd) [friend]

Definition at line 63 of file DCBraidReduction.h.

#### 9.23.3 Member Data Documentation

##### 9.23.3.1 int CommDivider::begin

Definition at line 60 of file DCBraidReduction.h.

Referenced by MaxCommutePartition::findCommutParts().

##### 9.23.3.2 int CommDivider::end

Definition at line 61 of file DCBraidReduction.h.

Referenced by MaxCommutePartition::findCommutParts().

### 9.23.3.3 `int CommDivider::gen`

Definition at line 62 of file `DCBraidReduction.h`.

Referenced by `MaxCommutePartition::findCommutParts()`.

### 9.23.3.4 `int CommDivider::length`

Definition at line 59 of file `DCBraidReduction.h`.

Referenced by `MaxCommutePartition::findCommutParts()`, and `lCommDivider::operator()`.

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

- `Experiments/include/DCBraidReduction.h`

## 9.24 compMaps Struct Reference

Implements a comparison operator of two maps.

```
#include <WhiteheadAutoSet.h>
```

### Public Member Functions

- `bool operator() (Map m1, Map m2) const`

#### 9.24.1 Detailed Description

Implements a comparison operator of two maps.

Definition at line 15 of file WhiteheadAutoSet.h.

#### 9.24.2 Member Function Documentation

##### 9.24.2.1 `bool compMaps::operator() (Map m1, Map m2) const` `[inline]`

Definition at line 17 of file WhiteheadAutoSet.h.

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

- Maps/include/WhiteheadAutoSet.h

## 9.25 ConfigFile Class Reference

Implements a mechanism for passing parameters from a configuration file.

```
#include <ConfigFile.h>
```

### Public Member Functions

- **ConfigFile** ()  
*Default constructor.*
- **ConfigFile** (const string &f\_name)  
*Constructor. Collects parameters and their values from a configuration file.*
- void **setVariable** (const char \*p\_name, const **Value** &p\_value)  
*Set a value of a parameter.*
- const **Value** & **getValue** (const char \*p\_name)  
*Get a value of a parameter.*

### Private Member Functions

- void **readFrom** (istream &istr)
- void **printOn** (ostream &ostr) const
- void **rtrim** (char \*ch)
- void **ltrim** (char \*ch)
- void **trim** (char \*ch)

### Private Attributes

- **ParameterType** parameters

### Friends

- ostream & **operator**<< (ostream &ostr, const **ConfigFile** &C)  
*Output operator.*

### 9.25.1 Detailed Description

Implements a mechanism for passing parameters from a configuration file. This class generates a list of parameters and their values from a given configuration file. The format of the configuration file is follows:

ParameterName1 : ParameterValue1

ParameterName2 : ParameterValue2

. . .

ParameterNameN : ParameterValueN

Parameter names are identifiers containing letters and numbers. Parameter values can be an integer or float point numbers and strings.

It is possible to add comments. Everything after a symbol # and until the end of a line is considered to be a comment.

#### Bug

On some platforms scanning may hang if an empty string is contained in the input file.

Definition at line 111 of file ConfigFile.h.

### 9.25.2 Constructor & Destructor Documentation

#### 9.25.2.1 ConfigFile::ConfigFile ()

Default constructor.

#### 9.25.2.2 ConfigFile::ConfigFile (const string & *f\_name*)

Constructor. Collects parameters and their values from a configuration file. Will scan the file whose name is given as a parameter and create a list of parameters together with the corresponding values.

#### Parameters:

*f\_name* - name of the configuration file.

### 9.25.3 Member Function Documentation

#### 9.25.3.1 const Value& ConfigFile::getValue (const char \* *p\_name*)

Get a value of a parameter. Returns a value of the parameter with the name *p\_name*.

Note, since class **Value** (p. 617) is equipped with the operators converting the parameter value into standard types, it is not necessary to create instances of the class **Value** (p. 617). Typical use of this function:

```
int v = configFile.getValue("THENAME");
```

or with explicit conversion:

```
string v = string(configFile.getValue("THENAME"));
```

**9.25.3.2 void ConfigFile::ltrim (char \* *ch*) [private]**

**9.25.3.3 void ConfigFile::printOn (ostream & *ostr*) const [private]**

**9.25.3.4 void ConfigFile::readFrom (istream & *istr*) [private]**

**9.25.3.5 void ConfigFile::rtrim (char \* *ch*) [private]**

**9.25.3.6 void ConfigFile::setVariable (const char \* *p\_name*, const Value & *p\_value*)**

Set a value of a parameter. Sets a value of a parameter with the given name. Will create the parameter if not in the list.

#### Parameters:

*p\_name* - the name of the parameter.

*p\_value* - the value of the parameter.

**9.25.3.7 void ConfigFile::trim (char \* *ch*) [private]**

## 9.25.4 Friends And Related Function Documentation

**9.25.4.1 ostream& operator<< (ostream & *ostr*, const ConfigFile & *C*) [friend]**

Output operator. Prints a list of parameters together with their values.

Definition at line 179 of file ConfigFile.h.



## 9.25.5 Member Data Documentation

### 9.25.5.1 ParameterType ConfigFile::parameters [private]

Definition at line 208 of file ConfigFile.h.

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

- general/include/ConfigFile.h

## 9.26 ConstWordIterator Class Reference

```
#include <PowerWordIterator.h>
```

### Public Types

- typedef pair< int, int > **PII**
- typedef pair< int, int > **PII**

### Public Member Functions

- **ConstWordIterator** ()
- **ConstWordIterator** (const **Word** &w, bool begin=true)
- **ConstWordIterator** (const **WordIterator** &WI)
- **ConstWordIterator** & **operator=** (const **WordIterator** &WI)
- bool **operator!=** (const **ConstWordIterator** &WI) const
- bool **operator==** (const **ConstWordIterator** &WI) const
- const **ConstWordIterator** & **operator++** ()
- **ConstWordIterator** **operator++** (int doomy)
- const **ConstWordIterator** & **operator--** ()
- **ConstWordIterator** **operator--** (int doomy)
- **PII** **operator\*** () const
- **ConstWordIterator** ()
- **ConstWordIterator** (const **Word** &w, bool begin=true)
- **ConstWordIterator** (const **WordIterator** &WI)
- **ConstWordIterator** & **operator=** (const **WordIterator** &WI)
- bool **operator!=** (const **ConstWordIterator** &WI) const
- bool **operator==** (const **ConstWordIterator** &WI) const
- const **ConstWordIterator** & **operator++** ()
- **ConstWordIterator** **operator++** (int doomy)
- const **ConstWordIterator** & **operator--** ()
- **ConstWordIterator** **operator--** (int doomy)
- int **operator\*** () const

### Private Attributes

- const **Word** \* **theWord**
- list< **PII** >::const\_iterator **theIterator**
- int **theOffset**
- const list< int > \* **theList**
- list< int >::const\_iterator **theIterator**

## Friends

- class **WordRep**

### 9.26.1 Detailed Description

Definition at line 81 of file PowerWordIterator.h.

### 9.26.2 Member Typedef Documentation

#### 9.26.2.1 typedef pair< int , int > ConstWordIterator::PII

Definition at line 90 of file WordIterator.h.

#### 9.26.2.2 typedef pair< int , int > ConstWordIterator::PII

Definition at line 83 of file PowerWordIterator.h.

### 9.26.3 Constructor & Destructor Documentation

9.26.3.1 `ConstWordIterator::ConstWordIterator ()`

9.26.3.2 `ConstWordIterator::ConstWordIterator (const Word & w, bool begin = true)`

9.26.3.3 `ConstWordIterator::ConstWordIterator (const WordIterator & WI)`

9.26.3.4 `ConstWordIterator::ConstWordIterator ()`

9.26.3.5 `ConstWordIterator::ConstWordIterator (const Word & w, bool begin = true)`

9.26.3.6 `ConstWordIterator::ConstWordIterator (const WordIterator & WI)`

### 9.26.4 Member Function Documentation

9.26.4.1 `bool ConstWordIterator::operator!= (const ConstWordIterator & WI) const`

9.26.4.2 `bool ConstWordIterator::operator!= (const ConstWordIterator & WI) const`

9.26.4.3 `int ConstWordIterator::operator* () const`

9.26.4.4 `Pii ConstWordIterator::operator* () const`

9.26.4.5 `ConstWordIterator ConstWordIterator::operator++ (int doomy)`

9.26.4.6 `const ConstWordIterator& ConstWordIterator::operator++ ()`

9.26.4.7 `ConstWordIterator ConstWordIterator::operator++ (int doomy)`

9.26.4.8 `const ConstWordIterator& ConstWordIterator::operator++ ()`

9.26.4.9 `ConstWordIterator ConstWordIterator::operator-- (int doomy)`

9.26.4.10 `const ConstWordIterator& ConstWordIterator::operator-- ()`

9.26.4.11 `ConstWordIterator ConstWordIterator::operator-- (int doomy)`

9.26.4.12 `const ConstWordIterator& ConstWordIterator::operator-- ()`

9.26.4.13 `ConstWordIterator& ConstWordIterator::operator= (const WordIterator & WI)`

---

9.26.4.14 `ConstWordIterator& ConstWordIterator::operator= (const WordIterator & WI)`

9.26.4.15 `bool ConstWordIterator::operator== (const ConstWordIterator & WI) const`

9.26.4.16 `bool ConstWordIterator::operator== (const ConstWordIterator & WI) const`

## 9.26.6 Member Data Documentation

### 9.26.6.1 `list< int >::const_iterator ConstWordIterator::theIterator` `[private]`

Definition at line 136 of file WordIterator.h.

### 9.26.6.2 `list< PII >::const_iterator ConstWordIterator::theIterator` `[private]`

Definition at line 132 of file PowerWordIterator.h.

### 9.26.6.3 `const list< int >* ConstWordIterator::theList` `[private]`

Definition at line 135 of file WordIterator.h.

### 9.26.6.4 `int ConstWordIterator::theOffset` `[private]`

Definition at line 133 of file PowerWordIterator.h.

### 9.26.6.5 `const Word* ConstWordIterator::theWord` `[private]`

Definition at line 131 of file PowerWordIterator.h.

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

- `Elt/include/PowerWordIterator.h`
- `Elt/include/WordIterator.h`

## 9.27 CutVertices Class Reference

Implements an algorithm to find all articulation points of a graph//.

```
#include <WhiteheadGraph.h>
```

### Public Member Functions

- **CutVertices** (const int \*\*G, int n)  
*Constructor.*
- **~CutVertices** ()
- void **compute** ()  
*Finds articulation points.*
- void **computeBruteForce** ()  
*Finds articulation points (Brute force algorithm).*
- int **numberOfComponents** (bool ignore\_single\_vertices=false)  
*Computes the number of connected components of the graph.*
- vector< int > **getCutVertices** () const  
*Get the list of articulation points.*

### Private Member Functions

- void **init** ()
- void **DepthFirstSearch** ()  
*Execute depth first search.*
- void **RecursiveDepthFirstSearch** (int v)  
*Execute recursive depth first search.*
- void **RDFS\_Compute\_Low** (int v)  
*Compute the low subtree recursively.*
- void **ArticulationPoints** ()  
*Extract articulation points from the labels.*

## Private Attributes

- **int \*\* theGraph**  
*Adjacency matrix of the graph.*
- **int N**  
*Number of vertices.*
- **int time**  
*Current time label (internal use only).*
- **vector< int > visit**  
*List of visited points (internal use only).*
- **vector< int > pred**  
*List of predecessors (internal use only).*
- **vector< int > discover**  
*Labels of discovered vertices (internal use only).*
- **vector< int > Low**
- **vector< int > articulation\_point**  
*List of articulation points.*

### 9.27.1 Detailed Description

Implements an algorithm to find all articulation points of a graph//.

Definition at line 29 of file WhiteheadGraph.h.

### 9.27.2 Constructor & Destructor Documentation

#### 9.27.2.1 CutVertices::CutVertices (const int \*\* *G*, int *n*)

Constructor.

##### Parameters:

*G* - the adjacency matrix of a graph ( there is an edge from *i* to *j* in the graph iff  $G[i][j] \neq 0$  ).

*n* - the number of vertices.

### 9.27.2.2 `CutVertices::~~CutVertices ()`

## 9.27.3 Member Function Documentation

### 9.27.3.1 `void CutVertices::ArticulationPoints () [private]`

Extract articulation points from the labels.

### 9.27.3.2 `void CutVertices::compute ()`

Finds articulation points. Finds all articulation points of the graph. The result can be obtained through `getCutVertices()` (p. 178) function. This algorithm requires linear time in the size of the graph, or  $O(n^2)$ .

### 9.27.3.3 `void CutVertices::computeBruteForce ()`

Finds articulation points (Brute force algorithm). Finds all articulation points of the graph. The result can be obtained through `getCutVertices()` (p. 178) function. Complexity:  $O(n^2)$ .

### 9.27.3.4 `void CutVertices::DepthFirstSearch () [private]`

Execute depth first search.

### 9.27.3.5 `vector<int> CutVertices::getCutVertices () const [inline]`

Get the list of articulation points. Return the list of articulation points obtained using one of the algorithms. `compute()` (p. 178) or `computeBruteForce()` (p. 178) must be called first.

#### Returns:

The list of articulation vertices.

Definition at line 72 of file WhiteheadGraph.h.

References `articulation_point`.



**9.27.3.6 void CutVertices::init () [private]****9.27.3.7 int CutVertices::numberOfComponents (bool *ignore\_single\_vertices* = false)**

Computes the number of connected components of the graph. Computes the number of connected components of the graph.

**Parameters:**

*ignore\_single\_vertices* - if set to `true`, then components containing only one vertex are ignored. Default value is `false`.

**Returns:**

The number of connected components

**9.27.3.8 void CutVertices::RDFS\_Compute\_Low (int *v*) [private]**

Compute the low subtree recursively.

**9.27.3.9 void CutVertices::RecursiveDepthFirstSearch (int *v*) [private]**

Execute recursive depth first search.

**9.27.4 Member Data Documentation****9.27.4.1 vector<int> CutVertices::articulation\_point [private]**

List of articulation points.

Definition at line 108 of file WhiteheadGraph.h.

Referenced by `getCutVertices()`.

**9.27.4.2 vector<int> CutVertices::discover [private]**

Labels of discovered vertices (internal use only).

Definition at line 103 of file WhiteheadGraph.h.

**9.27.4.3 vector<int> CutVertices::Low [private]**

Definition at line 104 of file WhiteheadGraph.h.

**9.27.4.4 int CutVertices::N [private]**

Number of vertices.

Definition at line 94 of file WhiteheadGraph.h.

**9.27.4.5 vector<int> CutVertices::pred [private]**

List of predecessors (internal use only).

Definition at line 101 of file WhiteheadGraph.h.

**9.27.4.6 int\*\* CutVertices::theGraph [private]**

Adjacency matrix of the graph.

Definition at line 91 of file WhiteheadGraph.h.

**9.27.4.7 int CutVertices::time [private]**

Current time label (internal use only).

Definition at line 97 of file WhiteheadGraph.h.

**9.27.4.8 vector<int> CutVertices::visit [private]**

List of visited points (internal use only).

Definition at line 99 of file WhiteheadGraph.h.

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

- FreeGroup/include/**WhiteheadGraph.h**

## 9.28 DCBraidReduction Class Reference

```
#include <DCBraidReduction.h>
```

### Public Member Functions

- **DCBraidReduction** (int *n*, **Perturbation** \**pert*, **Partition** \**part*)
- **Word shorten** (const **Word** &*w*) const

### Private Attributes

- int *N*
- **Perturbation** \* *thePerturb*
- **Partition** \* *thePartition*

#### 9.28.1 Detailed Description

Definition at line 133 of file DCBraidReduction.h.

#### 9.28.2 Constructor & Destructor Documentation

##### 9.28.2.1 DCBraidReduction::DCBraidReduction (int *n*, **Perturbation** \* *pert*, **Partition** \* *part*) [inline]

Definition at line 136 of file DCBraidReduction.h.

#### 9.28.3 Member Function Documentation

##### 9.28.3.1 Word DCBraidReduction::shorten (const **Word** & *w*) const [inline]

Definition at line 142 of file DCBraidReduction.h.

References `Word::begin()`, `Word::end()`, `Partition::getPartition()`, `N`, `Word::push_back()`, `shortenBraid()`, and `thePartition`.

#### 9.28.4 Member Data Documentation

##### 9.28.4.1 int DCBraidReduction::N [private]

Definition at line 167 of file DCBraidReduction.h.

Referenced by shorten().

#### **9.28.4.2 Partition\* DCBraidReduction::thePartition [private]**

Definition at line 169 of file DCBraidReduction.h.

Referenced by shorten().

#### **9.28.4.3 Perturbation\* DCBraidReduction::thePerturb [private]**

Definition at line 168 of file DCBraidReduction.h.

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

- Experiments/include/**DCBraidReduction.h**

## 9.29 DDL Struct Reference

```
#include <DDL.h>
```

### Public Member Functions

- **DDL ()**
- **DDL (Word w, bool bFreelyReduce=true)**
- **~DDL ()**
- void **append** (int v)
- void **delend** ()
- **Word toWord** ()
- void **delcur** (DDLNode \*p)
- void **insertA** (DDLNode \*p, int v)
- void **insertA** (DDLNode \*p, Word w)

### Public Attributes

- **DDLNode \* first**
- **DDLNode \* last**
- **int len**

#### 9.29.1 Detailed Description

Definition at line 34 of file DDL.h.

#### 9.29.2 Constructor & Destructor Documentation

##### 9.29.2.1 DDL::DDL () [inline]

Definition at line 42 of file DDL.h.

**9.29.2.2** `DDL::DDL (Word w, bool bFreelyReduce = true)`

**9.29.2.3** `DDL::~~DDL ()`

### **9.29.3 Member Function Documentation**

**9.29.3.1** `void DDL::append (int v)`

**9.29.3.2** `void DDL::delcur (DDLNode * p)`

**9.29.3.3** `void DDL::delend ()`

**9.29.3.4** `void DDL::insertA (DDLNode * p, Word w)`

**9.29.3.5** `void DDL::insertA (DDLNode * p, int v)`

**9.29.3.6** `Word DDL::toWord ()`

### **9.29.4 Member Data Documentation**

**9.29.4.1** `DDLNode* DDL::first`

Definition at line 74 of file DDL.h.

**9.29.4.2** `DDLNode* DDL::last`

Definition at line 75 of file DDL.h.

**9.29.4.3** `int DDL::len`

Definition at line 76 of file DDL.h.

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

- Experiments/include/**DDL.h**

## 9.30 DDLNode Struct Reference

```
#include <DDL.h>
```

### Public Member Functions

- **DDLNode** (int v)

### Public Attributes

- int value
- **DDLNode** \* left
- **DDLNode** \* right

#### 9.30.1 Detailed Description

Definition at line 24 of file DDL.h.

#### 9.30.2 Constructor & Destructor Documentation

##### 9.30.2.1 DDLNode::DDLNode (int v) [inline]

Definition at line 26 of file DDL.h.

#### 9.30.3 Member Data Documentation

##### 9.30.3.1 DDLNode\* DDLNode::left

Definition at line 29 of file DDL.h.

##### 9.30.3.2 DDLNode\* DDLNode::right

Definition at line 30 of file DDL.h.

##### 9.30.3.3 int DDLNode::value

Definition at line 28 of file DDL.h.

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

- Experiments/include/**DDL.h**



## 9.31 DehornoyForm Class Reference

Dehornoy Form of a braid word (aka/ handle free form).

```
#include <DehornoyForm.h>
```

### Public Member Functions

- **DehornoyForm** (const int N, const **Word** &w)  
 • **Word getDehornoyForm** () const  
 • **DehornoyForm** (const int N, const **Word** &w)  
     *Create a Dehornoy form of a braid word.*
- **Word getDehornoyForm** () const  
     *(Accessor function) Get a braid word representing a form.*

### Private Member Functions

- **Word computeDehornoyForm** (const **Word** &w)
- **Word computeDehornoyForm** (const **Word** &w)  
     *Function which computes the form (using **LinkedBraidStructure** (p. 305)).*

### Private Attributes

- const int **theIndex**  
     *the rank of the braid group = number of strands = number of generators + 1*
- **Word theDehornoyForm**  
     *The Dehornoy form.*
- **LinkedBraidStructure theLinkedStructure**  
     *Structure in which the form is kept.*

#### 9.31.1 Detailed Description

Dehornoy Form of a braid word (aka/ handle free form). This class uses **BraidLinkedStructure** to compute the Dehornoy form a braid-word. The object of this class is just a container to keep the form.

Definition at line 14 of file **DehornoyForm.h**.

## 9.31.2 Constructor & Destructor Documentation

### 9.31.2.1 DehornoyForm::DehornoyForm (const int $N$ , const Word & $w$ )

### 9.31.2.2 DehornoyForm::DehornoyForm (const int $N$ , const Word & $w$ )

Create a Dehornoy form of a braid word. Note that the form will be computed right here. So, if you are not sure that the form will be used later do not create this object.

#### Parameters:

$N$  - rank of a braid group;

$w$  - braid word

## 9.31.3 Member Function Documentation

### 9.31.3.1 Word DehornoyForm::computeDehornoyForm (const Word & $w$ ) [private]

Function which computes the form (using **LinkedBraidStructure** (p. 305)).

### 9.31.3.2 Word DehornoyForm::computeDehornoyForm (const Word & $w$ ) [private]

### 9.31.3.3 Word DehornoyForm::getDehornoyForm () const [inline]

(Accessor function) Get a braid word representing a form.

Definition at line 56 of file DehornoyForm\_old.h.

References theDehornoyForm.

### 9.31.3.4 Word DehornoyForm::getDehornoyForm () const [inline]

Definition at line 36 of file DehornoyForm.h.

References theDehornoyForm.

## 9.31.4 Member Data Documentation

### 9.31.4.1 Word DehornoyForm::theDehornoyForm [private]

The Dehornoy form.

Definition at line 60 of file DehornoyForm.h.

Referenced by `getDehornoyForm()`.

#### 9.31.4.2 `const int DehornoyForm::theIndex` **[private]**

the rank of the braid group = number of strands = number of generators + 1 The rank of the corresponding braid group.

Definition at line 58 of file `DehornoyForm.h`.

#### 9.31.4.3 `LinkedBraidStructure DehornoyForm::theLinkedStructure` **[private]**

Structure in which the form is kept.

Definition at line 85 of file `DehornoyForm_old.h`.

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

- `BraidGroup/include/DehornoyForm.h`
- `BraidGroup/include/DehornoyForm_old.h`

## 9.32 `dump< T >` Class Template Reference

```
#include <dump.h>
```

### Public Types

- typedef class T::const\_iterator **const\_iterator**

### Public Member Functions

- **dump** (const T &s)
- void **printOn** (ostream &o) const

### Private Attributes

- const T & **theCont**

### Friends

- ostream & **operator<<** (ostream &o, const **dump** &d)

#### 9.32.1 Detailed Description

`template<class T> class dump< T >`

Definition at line 25 of file dump.h.

#### 9.32.2 Member Typedef Documentation

**9.32.2.1** `template<class T > typedef class T::const_iterator dump< T >::const_iterator`

Definition at line 28 of file dump.h.

#### 9.32.3 Constructor & Destructor Documentation

**9.32.3.1** `template<class T > dump< T >::dump (const T &s) [inline]`

Definition at line 29 of file dump.h.

## 9.32.4 Member Function Documentation

### 9.32.4.1 `template<class T> void dump< T >::printOn (ostream & o) const` `[inline]`

Definition at line 31 of file dump.h.

References dump< T >::theCont.

## 9.32.5 Friends And Related Function Documentation

### 9.32.5.1 `template<class T> ostream& operator<< (ostream & o, const` `dump< T > & d) [friend]`

Definition at line 38 of file dump.h.

## 9.32.6 Member Data Documentation

### 9.32.6.1 `template<class T> const T& dump< T >::theCont` `[private]`

Definition at line 44 of file dump.h.

Referenced by dump< T >::printOn().

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

- general/include/dump.h

## 9.33 Dump Class Reference

```
#include <MajorDump.h>
```

### Static Public Attributes

- static ostream \* **dump\_out**

#### 9.33.1 Detailed Description

Definition at line 8 of file MajorDump.h.

#### 9.33.2 Member Data Documentation

##### 9.33.2.1 ostream\* Dump::dump\_out [static]

Definition at line 10 of file MajorDump.h.

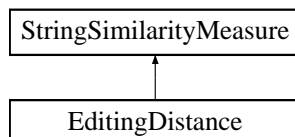
The documentation for this class was generated from the following file:

- CryptoAAG/include/**MajorDump.h**

## 9.34 EditingDistance Class Reference

Implements the Editing (**Levenstein** (p. 303)) distance between two words.

`#include <SimilarityMeasures.h>`Inheritance diagram for EditingDistance::



### Public Member Functions

- **EditingDistance** ()
- double **measure** (const **Word** &w1, const **Word** &w2) const  
*Returns the editing distance.*

#### 9.34.1 Detailed Description

Implements the Editing (**Levenstein** (p. 303)) distance between two words.

Definition at line 162 of file SimilarityMeasures.h.

#### 9.34.2 Constructor & Destructor Documentation

##### 9.34.2.1 EditingDistance::EditingDistance () [inline]

Definition at line 165 of file SimilarityMeasures.h.

#### 9.34.3 Member Function Documentation

##### 9.34.3.1 double EditingDistance::measure (const Word & w1, const Word & w2) const [virtual]

Returns the editing distance.

##### Parameters:

*w1* - the first word

$w_2$  - the second word

**Returns:**

the scaled Editing distance.

Implements **StringSimilarityMeasure** (p. 520).

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

- StringSimilarity/include/**SimilarityMeasures.h**



## 9.35 Equation Class Reference

```
#include <Equation.h>
```

### Public Member Functions

- **Equation** (int nGen, int nVar, const **Word** &eq)
  - bool **isGenerator** (int g) const  
*Determine if g is a letter in this equation.*
- bool **isVariable** (int g) const  
*Determine if g is a variable in this equation.*
- int **getTheNumberOfGenerators** () const  
*Get the number of the generators of the group.*
- int **getTheNumberOfVariables** () const  
*Get the number of variables in the formula.*
- const **Word** & **getTheEquation** () const  
*Get the word presentation of the equation.*
- bool **isQuadratic** () const  
*Determine if the equation is quadratic.*
- bool **trivialSolution** () const  
*Determine if the equation has trivial solution.*

### Static Public Member Functions

- static **Equation randomQuadraticEquation** (int nGen, int nVar, int len)  
*Generate random (strictly) quadratic equation of length  $len + 2 \cdot nVars$  with  $nGen$  generators,  $nVar$  variables.*

### Private Attributes

- int **theNumberOfGenerators**  
*the number of the generators of the group*

- **int theNumberOfVariables**  
*the number of variables in the formula (the actual number of variables in the equation can be lesser)*
- **Word theEquation**  
*the presentation of the equation*

## Friends

- **ostream & operator<<** (ostream &os, const **Equation** &eq)

### 9.35.1 Detailed Description

Definition at line 27 of file Equation.h.

### 9.35.2 Constructor & Destructor Documentation

#### 9.35.2.1 **Equation::Equation** (int *nGen*, int *nVar*, const Word & *eq*)

### 9.35.3 Member Function Documentation

#### 9.35.3.1 **const Word& Equation::getTheEquation** () const **[inline]**

Get the word presentation of the equation.

Definition at line 66 of file Equation.h.

References theEquation.

#### 9.35.3.2 **int Equation::getTheNumberOfGenerators** () const **[inline]**

Get the number of the generators of the group.

Definition at line 58 of file Equation.h.

References theNumberOfGenerators.

#### 9.35.3.3 **int Equation::getTheNumberOfVariables** () const **[inline]**

Get the number of variables in the formula.

Definition at line 62 of file Equation.h.

References theNumberOfVariables.

**9.35.3.4 bool Equation::isGenerator (int *g*) const**

Determine if *g* is a letter in this equation.

**9.35.3.5 bool Equation::isQuadratic () const**

Determine if the equation is quadratic.

**9.35.3.6 bool Equation::isVariable (int *g*) const**

Determine if *g* is a variable in this equation.

**9.35.3.7 static Equation Equation::randomQuadraticEquation (int *nGen*, int *nVar*, int *len*) [static]**

Generate random (strictly) quadratic equation of length  $len + 2 \cdot nVars$  with *nGen* generators, *nVar* variables. Routine "arranges" *len* generators and  $2 \cdot nVars$  variables (each variable twice) into a reduced equation. The distribution is not uniform among equations of this type (even though for large values of *len* I think it will be close to uniform). For uniform distribution one has to construct a **FSA** (p. 216) accepting all equations of this type, then assign weights to edges using dynamic programming, and finally choose words from that **FSA** (p. 216) according to the weights.

**9.35.3.8 bool Equation::trivialSolution () const**

Determine if the equation has trivial solution.

**9.35.4 Friends And Related Function Documentation****9.35.4.1 ostream& operator<< (ostream & *os*, const Equation & *eq*) [friend]****9.35.5 Member Data Documentation****9.35.5.1 Word Equation::theEquation [private]**

the presentation of the equation A word *theEquation* is a sequence of generators. Each generator *g* is interpreted the following way:

- if  $|g| \leq theNumberOfGenerators$  then *g* is the corresponding generator of the group

- if  $|g| > theNumberOfGenerators$  then  $g$  is a variable with index  $|g| - theNumberOfGenerators$  raised in the power  $\pm 1$  depending on the sign of  $g$ .

Definition at line 126 of file Equation.h.

Referenced by `getTheEquation()`.

#### 9.35.5.2 `int Equation::theNumberOfGenerators` **[private]**

the number of the generators of the group

Definition at line 116 of file Equation.h.

Referenced by `getTheNumberOfGenerators()`.

#### 9.35.5.3 `int Equation::theNumberOfVariables` **[private]**

the number of variables in the formula (the actual number of variables in the equation can be lesser)

Definition at line 118 of file Equation.h.

Referenced by `getTheNumberOfVariables()`.

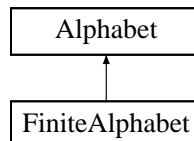
The documentation for this class was generated from the following file:

- Equation/include/**Equation.h**

## 9.36 FiniteAlphabet Class Reference

Implements a finite size alphabet.

`#include <Alphabet.h>`Inheritance diagram for FiniteAlphabet::



### Public Member Functions

- **FiniteAlphabet ()**

*Default Constructor.*

#### 9.36.1 Detailed Description

Implements a finite size alphabet. This is an implementation of a finite alphabet. Letter names are given as a parameter in the constructor

Definition at line 67 of file Alphabet.h.

#### 9.36.2 Constructor & Destructor Documentation

##### 9.36.2.1 FiniteAlphabet::FiniteAlphabet () [inline]

Default Constructor.

Definition at line 71 of file Alphabet.h.

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

- Alphabet/include/**Alphabet.h**

## 9.37 FoldDetails Struct Reference

```
#include <FSAREp.h>
```

### Public Member Functions

- **FoldDetails** (int *b*, int *l*, int *n*, const **FSASState** &*s1*, const **FSASState** &*s2*)

### Public Attributes

- int **base**
- int **label**
- int **state\_num**
- **FSASState** **state1**
- **FSASState** **state2**

#### 9.37.1 Detailed Description

Definition at line 88 of file FSAREp.h.

#### 9.37.2 Constructor & Destructor Documentation

**9.37.2.1** **FoldDetails::FoldDetails** (int *b*, int *l*, int *n*, const **FSASState** &*s1*, const **FSASState** &*s2*) [**inline**]

Definition at line 90 of file FSAREp.h.

#### 9.37.3 Member Data Documentation

**9.37.3.1** int **FoldDetails::base**

Definition at line 93 of file FSAREp.h.

**9.37.3.2** int **FoldDetails::label**

Definition at line 94 of file FSAREp.h.

**9.37.3.3** **FSASState** **FoldDetails::state1**

Definition at line 96 of file FSAREp.h.

#### 9.37.3.4 FSAStruct FoldDetails::state2

Definition at line 97 of file FSAStruct.h.

#### 9.37.3.5 int FoldDetails::state\_num

Definition at line 95 of file FSAStruct.h.

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

- Graph/include/FSAStruct.h

## 9.38 Graphs::FoldDetails< VertexType, EdgeType > Struct Template Reference

```
#include <GraphConceptAlgorithms.h>
```

### Public Member Functions

- **FoldDetails** (int *o*, const EdgeType &*e1*, const EdgeType &*e2*, const VertexType &*V1*, const VertexType &*V2*)

### Public Attributes

- int **theOrigin**
- EdgeType **theEdge1**
- EdgeType **theEdge2**
- VertexType **theVertex1**
- VertexType **theVertex2**

### 9.38.1 Detailed Description

```
template<class VertexType, class EdgeType> struct Graphs::FoldDetails< VertexType, EdgeType >
```

It is assumed that `theEdge1.theTarget < theEdge2.theTarget`.

Definition at line 377 of file `GraphConceptAlgorithms.h`.

### 9.38.2 Constructor & Destructor Documentation

**9.38.2.1** `template<class VertexType , class EdgeType > Graphs::FoldDetails< VertexType, EdgeType >::FoldDetails (int o, const EdgeType & e1, const EdgeType & e2, const VertexType & V1, const VertexType & V2) [inline]`

Definition at line 379 of file `GraphConceptAlgorithms.h`.



### 9.38.3 Member Data Documentation

**9.38.3.1** `template<class VertexType , class EdgeType > EdgeType  
Graphs::FoldDetails< VertexType, EdgeType >::theEdge1`

Definition at line 387 of file GraphConceptAlgorithms.h.

**9.38.3.2** `template<class VertexType , class EdgeType > EdgeType  
Graphs::FoldDetails< VertexType, EdgeType >::theEdge2`

Definition at line 388 of file GraphConceptAlgorithms.h.

**9.38.3.3** `template<class VertexType , class EdgeType > int  
Graphs::FoldDetails< VertexType, EdgeType >::theOrigin`

Definition at line 386 of file GraphConceptAlgorithms.h.

**9.38.3.4** `template<class VertexType , class EdgeType > VertexType  
Graphs::FoldDetails< VertexType, EdgeType >::theVertex1`

Definition at line 390 of file GraphConceptAlgorithms.h.

**9.38.3.5** `template<class VertexType , class EdgeType > VertexType  
Graphs::FoldDetails< VertexType, EdgeType >::theVertex2`

Definition at line 391 of file GraphConceptAlgorithms.h.

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

- Graph/include/GraphConceptAlgorithms.h

## 9.39 FPGGroup Class Reference

Class **FPGGroup** (p. 204) (finitely presented group).

```
#include <FPGGroup.h>
```

### Public Member Functions

- **FPGGroup** (int num=0)
- **FPGGroup** (int numOfGen, const vector< **Word** > &relators)
- **FPGGroup** (const **FiniteAlphabet** &a)
- **FPGGroup** (const **FiniteAlphabet** &a, const vector< **Word** > &relators)
- int **numberOfGenerators** () const  
*Get the number of generators.*
- const vector< string > &**getGeneratorsNames** () const  
*Get the names of the generators.*
- const **FiniteAlphabet** &**getAlphabet** () const  
*Get the alphabet.*
- const vector< **Word** > &**relators** () const
- **Word randomEqWord\_Baltimore** (const **Word** &w, int length, float conj\_param) const  
*Generate random equivalent word.*
- **Word randomIdentity\_Stack** (int length) const  
*Generate random trivial word.*
- **Word randomIdentity\_Classic** (int length, float conj\_param) const  
*Generate random trivial word.*
- **Word randomIdentity\_Baltimore** (int length, float conj\_param) const  
*Generate random trivial word (using randomEqWord\_Baltimore).*
- **FPGGroup triangulatePresentation** () const  
*Triangulate the set of relations.*

### Static Public Member Functions

- static vector< string > **initializeGenNames** (int num)

## Protected Attributes

- int **numOfGenerators**
- vector< Word > **theRelators**
- FiniteAlphabet **theAlphabet**
- bool **useDefaultAlphabet**

## Friends

- ostream & **operator**<< (ostream &os, const **FPGROUP** &group)
- istream & **operator**>> (istream &is, **FPGROUP** &group)

### 9.39.1 Detailed Description

Class **FPGROUP** (p. 204) (finitely presented group).

Definition at line 32 of file FPGROUP.h.

### 9.39.2 Constructor & Destructor Documentation

**9.39.2.1** FPGROUP::FPGROUP (int *num* = 0)

**9.39.2.2** FPGROUP::FPGROUP (int *numOfGen*, const vector< Word > & *relators*)

**9.39.2.3** FPGROUP::FPGROUP (const FiniteAlphabet & *a*)

**9.39.2.4** FPGROUP::FPGROUP (const FiniteAlphabet & *a*, const vector< Word > & *relators*)

### 9.39.3 Member Function Documentation

**9.39.3.1** const FiniteAlphabet& FPGROUP::getAlphabet () const [inline]

Get the alphabet.

Definition at line 63 of file FPGROUP.h.

References theAlphabet.

**9.39.3.2** const vector< string >& FPGROUP::getGeneratorsNames () const [inline]

Get the names of the generators.

Definition at line 61 of file FPGroup.h.

References theAlphabet.

**9.39.3.3 static vector< string > FPGroup::initializeGenNames (int *num*)  
[static]**

**9.39.3.4 int FPGroup::numberOfGenerators () const [inline]**

Get the number of generators.

Definition at line 59 of file FPGroup.h.

References numOfGenerators.

**9.39.3.5 Word FPGroup::randomEqWord\_Baltimore (const Word & *w*, int  
*length*, float *conj\_param*) const**

Generate random equivalent word. Function inserts into random positions in the given word  $w$  relators (and their cyclic permutations and inverses) of the group conjugated by randomly chosen words. The lengths of conjugators is chosen using geometric distribution with parameter = *conj\_param*. When the required length is reached the word is being reduced and output. So, the result often is shorter than the given parameter length.

**9.39.3.6 Word FPGroup::randomIdentity\_Baltimore (int *length*, float  
*conj\_param*) const**

Generate random trivial word (using randomEqWord\_Baltimore).

**9.39.3.7 Word FPGroup::randomIdentity\_Classic (int *length*, float  
*conj\_param*) const**

Generate random trivial word. Function starts with a trivial word  $w = \varepsilon$ . On each iteration it multiplies  $w$  on the right by a relator (and their cyclic permutations and inverses) of the group conjugated by randomly chosen words. Lengths of conjugators are chosen using geometric distribution with parameter = *conj\_param*. When the required length is reached the word is being reduced and output. So, the result often is shorter than the given parameter length.

**9.39.3.8 Word FPGroup::randomIdentity\_Stack (int *length*) const**

Generate random trivial word. Function starts with a pair of trivial words  $w_1 = \varepsilon, w_2 = \varepsilon$ . On each iteration randomly takes a relators  $r$ , takes a random cyclic permutation  $r'$

of it or its inverse, then randomly cuts in two pieces  $r' = r_1 \circ r_2$  (one can be trivial) and multiplies  $w_1$  on the right by  $r_1$  and  $w_2$  on the right by  $r_2^{-1}$ . When  $|w_1| + |w_2|$  reaches the length output freely reduced  $w_1 w_2^{-1}$ .

#### 9.39.3.9 `const vector< Word >& FPGROUP::relators () const [inline]`

Definition at line 65 of file FPGROUP.h.

References `theRelators`.

#### 9.39.3.10 `FPGROUP FPGROUP::triangulatePresentation () const`

Triangulate the set of relations.

#### Returns:

The result is a finite group presentation  $\langle Y; S \rangle$  with all relators of length at most 3. Notice that the names of the original generators change to  $x_1, \dots, x_n$  to avoid possible name collisions. New generator names are  $x_{n+1}, \dots$

### 9.39.4 Friends And Related Function Documentation

#### 9.39.4.1 `ostream& operator<< (ostream & os, const FPGROUP & group) [friend]`

#### 9.39.4.2 `istream& operator>> (istream & is, FPGROUP & group) [friend]`

### 9.39.5 Member Data Documentation

#### 9.39.5.1 `int FPGROUP::numOfGenerators [protected]`

Definition at line 184 of file FPGROUP.h.

Referenced by `numberOfGenerators()`.

#### 9.39.5.2 `FiniteAlphabet FPGROUP::theAlphabet [protected]`

Definition at line 189 of file FPGROUP.h.

Referenced by `getAlphabet()`, and `getGeneratorsNames()`.

#### 9.39.5.3 `vector< Word > FPGROUP::theRelators [protected]`

Definition at line 187 of file FPGROUP.h.

Referenced by `relators()`.

#### **9.39.5.4   `bool FPGroup::useDefaultAlphabet`   `[protected]`**

Definition at line 190 of file `FPGroup.h`.

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

- `Group/include/FPGroup.h`

## 9.40 FRDFIT Class Reference

```
#include <FRDFIT.h>
```

### Public Member Functions

- **FRDFIT** (int *n*)
- **Word compute** (const **Word** &, int &numOfHandles)

### Private Member Functions

- void **pp** (const **DDL** &*L*, **DDLNode** \**p1*, **DDLNode** \**p2*)

### Private Attributes

- int **theGroupGens**

#### 9.40.1 Detailed Description

Definition at line 26 of file FRDFIT.h.

#### 9.40.2 Constructor & Destructor Documentation

##### 9.40.2.1 FRDFIT::FRDFIT (int *n*) [inline]

Definition at line 36 of file FRDFIT.h.

#### 9.40.3 Member Function Documentation

##### 9.40.3.1 Word FRDFIT::compute (const Word &, int & *numOfHandles*)

##### 9.40.3.2 void FRDFIT::pp (const DDL & *L*, DDLNode \* *p1*, DDLNode \* *p2*) [private]

#### 9.40.4 Member Data Documentation

##### 9.40.4.1 int FRDFIT::theGroupGens [private]

Definition at line 68 of file FRDFIT.h.

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

- Experiments/include/**FRDFIT.h**



## 9.41 FreeGroup Class Reference

```
#include <FreeGroup.h>
```

### Public Member Functions

- **FreeGroup** (int rank)  
*(Constructor) Free group of the given rank.*
- **FreeGroup** (const **FiniteAlphabet** &a)
- bool **isPrimitive** (const **Word** &w) const  
*Determine if the word w is primitive or not.*
- bool **isAlmostPrimitive** (const **Word** &w) const  
*Determine if the word w is almost primitive or not.*
- bool **doesContain** (const **SubgroupFG** &sbgp, const **Word** &w) const  
*Determine whether a subgroup sbgp of a free group contains a word w.*
- const **FiniteAlphabet** & **getAlphabet** () const

### Private Attributes

- int **theRank**
- **FiniteAlphabet** **theAlphabet**
- bool **useDefaultAlphabet**

### Friends

- ostream & **operator**<< (ostream &out, const **FreeGroup** &g)  
*Output the presentation into a stream.*
- istream & **operator**>> (istream &in, **FreeGroup** &g)  
*Read an alphabet from a string.*

#### 9.41.1 Detailed Description

Definition at line 24 of file FreeGroup.h.

## 9.41.2 Constructor & Destructor Documentation

### 9.41.2.1 FreeGroup::FreeGroup (int *rank*)

(Constructor) Free group of the given rank.

### 9.41.2.2 FreeGroup::FreeGroup (const FiniteAlphabet & *a*)

## 9.41.3 Member Function Documentation

### 9.41.3.1 bool FreeGroup::doesContain (const SubgroupFG & *sbgp*, const Word & *w*) const

Determine whether a subgroup *sbgp* of a free group contains a word *w*.

### 9.41.3.2 const FiniteAlphabet& FreeGroup::getAlphabet () const [inline]

Definition at line 56 of file FreeGroup.h.

References `theAlphabet`.

### 9.41.3.3 bool FreeGroup::isAlmostPrimitive (const Word & *w*) const

Determine if the word *w* is almost primitive or not.

### 9.41.3.4 bool FreeGroup::isPrimitive (const Word & *w*) const

Determine if the word *w* is primitive or not.

## 9.41.4 Friends And Related Function Documentation

### 9.41.4.1 ostream& operator<< (ostream & *out*, const FreeGroup & *g*) [friend]

Output the presentation into a stream.

Definition at line 65 of file FreeGroup.h.

### 9.41.4.2 istream& operator>> (istream & *in*, FreeGroup & *g*) [friend]

Read an alphabet from a string.

Definition at line 75 of file FreeGroup.h.

## 9.41.5 Member Data Documentation

### 9.41.5.1 FiniteAlphabet FreeGroup::theAlphabet [private]

Definition at line 92 of file FreeGroup.h.

Referenced by getAlphabet().

### 9.41.5.2 int FreeGroup::theRank [private]

Definition at line 91 of file FreeGroup.h.

### 9.41.5.3 bool FreeGroup::useDefaultAlphabet [private]

Definition at line 93 of file FreeGroup.h.

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

- FreeGroup/include/**FreeGroup.h**

## 9.42 FreeMetabelianGroupAlgorithms Class Reference

Static class **FreeMetabelianGroupAlgorithms** (p. 214) encapsulates algorithms for FreeMetabelianGroup groups.

```
#include <FreeMetabelianGroupAlgorithms.h>
```

### Public Member Functions

- **FreeMetabelianGroupAlgorithms** ()

*Default constructor is not instantiated to protect from creating the objects of this class.*

### Static Public Member Functions

- static bool **trivial** (int N, const **Word** &w)

*Decide if a word represents the identity of a free metabelian group.*

- static pair< bool, **Word** > **conjugate** (int N, **Word** w1, **Word** w2)

*Decide if words represent conjugate elements in a free metabelian group.*

- static **Word** **getWordFromEdgeMap** (int N, const map< vector< int >, int > &EM)

*Compute a word (not a shortest) defining the same edge map as EM.*

### 9.42.1 Detailed Description

Static class **FreeMetabelianGroupAlgorithms** (p. 214) encapsulates algorithms for FreeMetabelianGroup groups. The class **FreeMetabelianGroupAlgorithms** (p. 214) is static, i.e., all member functions are static and there is no constructor defined.

Definition at line 36 of file FreeMetabelianGroupAlgorithms.h.

### 9.42.2 Constructor & Destructor Documentation

#### 9.42.2.1 FreeMetabelianGroupAlgorithms::FreeMetabelianGroupAlgorithms ()

Default constructor is not instantiated to protect from creating the objects of this class.

### 9.42.3 Member Function Documentation

**9.42.3.1** `static pair< bool , Word > FreeMetabelianGroupAlgorithms::conjugate (int  $N$ , Word  $w1$ , Word  $w2$ )`  
**[static]**

Decide if words represent conjugate elements in a free metabelian group.  $N$  is the rank of the corresponding metabelian group,  $w1, w2$  are the given word. The function returns a pair  $(A, B)$  where  $A$  is true if and only if words are conjugate and  $B$  is a conjugator, i.e.,  $B^{-1}w1B = w2$  holds.

**9.42.3.2** `static Word FreeMetabelianGroupAlgorithms::getWordFromEdgeMap (int  $N$ , const map< vector< int >, int > &  $EM$ )` **[static]**

Compute a word (not a shortest) defining the same edge map as  $EM$ . It is assumed here that  $EM$  defines an element of  $M'_r$ .

**9.42.3.3** `static bool FreeMetabelianGroupAlgorithms::trivial (int  $N$ , const Word &  $w$ )` **[static]**

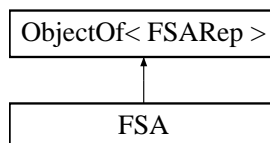
Decide if a word represents the identity of a free metabelian group.  $N$  is the rank of the corresponding metabelian group,  $w$  is the given word.

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

- FreeMetabelianGroup/include/FreeMetabelianGroupAlgorithms.h

## 9.43 FSA Class Reference

#include <FSA.h> Inheritance diagram for FSA::



### Public Types

- typedef **FSAState** **state\_type**
- typedef **FSAState::edge\_type** **edge\_type**

### Public Member Functions

- **FSA** ()
- **FSA operator\*** (const **FSA** &F) const
- bool **operator==** (const **FSA** &F) const
- void **fold** (const set< int > \*candidates=NULL, list< **FoldDetails** > \*details=NULL)
- void **pinch** (int state1, int state2)
- void **unfold** (const list< **FoldDetails** > &details)
- void **liftup** (const list< **FoldDetails** > &details, list< **FSAEdge** > &path, int init\_state)
- bool **isDeterministic** () const
- **FSA deterministic** () const
- int **newState** ()
- void **eraseState** (int state)
- void **newEdge** (int state1, int state2, int label)
- void **eraseEdge** (int state1, int state2, int label)
- template<class ConstIntIterator >  
void **addLoop** (int vert, ConstIntIterator F, ConstIntIterator L)
- template<class ConstIntIterator >  
void **addRay** (int vert, ConstIntIterator F, ConstIntIterator L)
- void **addFSA** (int vert1, int vert2, const **FSA** &fsa)
- const map< int, **FSAState** > & **getStates** () const
- map< int, **FSAState** > & **getStates** ()
- void **makeInitial** (int s)
- void **makeTerminal** (int s)

- void **makeNonInitial** (int s)
- void **makeNonTerminal** (int s)
- const set< int > & **getInitStates** () const
- const set< int > & **getTermStates** () const

## Private Member Functions

- **FSA** (const **FSARep** &rep)

### 9.43.1 Detailed Description

Definition at line 21 of file FSA.h.

### 9.43.2 Member Typedef Documentation

#### 9.43.2.1 typedef FSAState::edge\_type FSA::edge\_type

Definition at line 26 of file FSA.h.

#### 9.43.2.2 typedef FSAState FSA::state\_type

Definition at line 25 of file FSA.h.

### 9.43.3 Constructor & Destructor Documentation

#### 9.43.3.1 FSA::FSA () [inline]

Definition at line 36 of file FSA.h.

#### 9.43.3.2 FSA::FSA (const FSARep & rep) [inline, private]

Definition at line 42 of file FSA.h.

### 9.43.4 Member Function Documentation

#### 9.43.4.1 void FSA::addFSA (int *vert1*, int *vert2*, const FSA & *fsa*) [inline]

Definition at line 91 of file FSA.h.

References `FSARep::addFSA()`, `ObjectOf< FSARep >::change()`, and `ObjectOf< Rep >::look()`.

**9.43.4.2** `template<class ConstIntIterator > void FSA::addLoop (int vert,  
ConstIntIterator F, ConstIntIterator L) [inline]`

Definition at line 87 of file FSA.h.

References FSARep::addLoop(), and ObjectOf< FSARep >::change().

**9.43.4.3** `template<class ConstIntIterator > void FSA::addRay (int vert,  
ConstIntIterator F, ConstIntIterator L) [inline]`

Definition at line 89 of file FSA.h.

References FSARep::addRay(), and ObjectOf< FSARep >::change().

**9.43.4.4** `FSA FSA::deterministic () const`

**9.43.4.5** `void FSA::eraseEdge (int state1, int state2, int label) [inline]`

Definition at line 85 of file FSA.h.

References ObjectOf< FSARep >::change(), and eraseEdge().

Referenced by eraseEdge().

**9.43.4.6** `void FSA::eraseState (int state) [inline]`

Definition at line 83 of file FSA.h.

References ObjectOf< FSARep >::change(), and eraseState().

Referenced by eraseState().

**9.43.4.7** `void FSA::fold (const set< int > * candidates = NULL, list<  
FoldDetails > * details = NULL) [inline]`

Definition at line 64 of file FSA.h.

References ObjectOf< FSARep >::change().

**9.43.4.8** `const set< int >& FSA::getInitStates () const [inline]`

Definition at line 108 of file FSA.h.

References getInitStates(), and ObjectOf< FSARep >::look().

Referenced by getInitStates().



**9.43.4.9 map< int , FSASState >& FSA::getStates () [inline]**

Definition at line 94 of file FSA.h.

References ObjectOf< FSASRep >::change(), and FSASRep::getStates().

**9.43.4.10 const map< int , FSASState >& FSA::getStates () const [inline]**

Definition at line 93 of file FSA.h.

References FSASRep::getStates(), and ObjectOf< FSASRep >::look().

**9.43.4.11 const set< int >& FSA::getTermStates () const [inline]**

Definition at line 109 of file FSA.h.

References getTermStates(), and ObjectOf< FSASRep >::look().

Referenced by getTermStates().

**9.43.4.12 bool FSA::isDeterministic () const****9.43.4.13 void FSA::liftup (const list< FoldDetails > & details, list< FSAEdge > & path, int init\_state) [inline]**

Definition at line 70 of file FSA.h.

References ObjectOf< FSASRep >::change(), and liftup().

Referenced by liftup().

**9.43.4.14 void FSA::makeInitial (int s) [inline]**

Definition at line 104 of file FSA.h.

References ObjectOf< FSASRep >::change(), and makeInitial().

Referenced by makeInitial().

**9.43.4.15 void FSA::makeNonInitial (int s) [inline]**

Definition at line 106 of file FSA.h.

References ObjectOf< FSASRep >::change(), and makeNonInitial().

Referenced by makeNonInitial().

**9.43.4.16 void FSA::makeNonTerminal (int *s*) [inline]**

Definition at line 107 of file FSA.h.

References `ObjectOf< FSARep >::change()`, and `makeNonTerminal()`.

Referenced by `makeNonTerminal()`.

**9.43.4.17 void FSA::makeTerminal (int *s*) [inline]**

Definition at line 105 of file FSA.h.

References `ObjectOf< FSARep >::change()`, and `makeTerminal()`.

Referenced by `makeTerminal()`.

**9.43.4.18 void FSA::newEdge (int *state1*, int *state2*, int *label*) [inline]**

Definition at line 84 of file FSA.h.

References `ObjectOf< FSARep >::change()`, and `newEdge()`.

Referenced by `newEdge()`.

**9.43.4.19 int FSA::newState () [inline]**

Definition at line 82 of file FSA.h.

References `ObjectOf< FSARep >::change()`, and `newState()`.

Referenced by `newState()`.

**9.43.4.20 FSA FSA::operator\* (const FSA & *F*) const****9.43.4.21 bool FSA::operator== (const FSA & *F*) const****9.43.4.22 void FSA::pinch (int *state1*, int *state2*) [inline]**

Definition at line 67 of file FSA.h.

References `ObjectOf< FSARep >::change()`, and `pinch()`.

Referenced by `pinch()`.

**9.43.4.23 void FSA::unfold (const list< FoldDetails > & *details*) [inline]**

Definition at line 69 of file FSA.h.

References `ObjectOf< FSAREp >::change()`, and `unfold()`.

Referenced by `unfold()`.

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

- `Graph/include/FSA.h`

## 9.44 FSAEdge Struct Reference

```
#include <FSAREp.h>
```

### Public Member Functions

- **FSAEdge** ()
- **FSAEdge** (int *t*, int *l*)
- bool **operator==** (const **FSAEdge** &*e*) const
- bool **operator<** (const **FSAEdge** &*e*) const

### Public Attributes

- int **target**
- int **label**

#### 9.44.1 Detailed Description

Definition at line 29 of file FSAREp.h.

#### 9.44.2 Constructor & Destructor Documentation

##### 9.44.2.1 FSAEdge::FSAEdge () [inline]

Definition at line 32 of file FSAREp.h.

##### 9.44.2.2 FSAEdge::FSAEdge (int *t*, int *l*) [inline]

Definition at line 35 of file FSAREp.h.

#### 9.44.3 Member Function Documentation

##### 9.44.3.1 bool FSAEdge::operator< (const FSAEdge & *e*) const [inline]

Definition at line 41 of file FSAREp.h.

References `label`, and `target`.

**9.44.3.2 bool FSAEdge::operator==(const FSAEdge & e) const [inline]**

Definition at line 37 of file FSAREp.h.

References label, and target.

**9.44.4 Member Data Documentation****9.44.4.1 int FSAEdge::label**

Definition at line 53 of file FSAREp.h.

Referenced by operator<(), and operator==().

**9.44.4.2 int FSAEdge::target**

Definition at line 52 of file FSAREp.h.

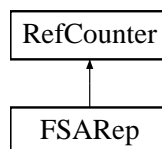
Referenced by operator<(), and operator==().

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

- Graph/include/**FSAREp.h**

## 9.45 FSAREp Class Reference

#include <FSAREp.h> Inheritance diagram for FSAREp::



### Public Member Functions

- **FSAREp \* clone ()** const
- void **fold** (const set< int > \*candidates=NULL, list< **FoldDetails** > \*details=NULL)  
*Fold a FSA (p. 216). Outputs details of the fold, that will allow to unfold latter. Argument candidates is a list of vertices to check for fold (all others won't be checked). If candidates=NULL then all vertices will be checked.*
- void **pinch** (int state1, int state2)  
*Pinch two vertices.*
- void **unfold** (const list< **FoldDetails** > &details)  
*Unfold 2 vertices using details of the previous fold.*
- void **liftup** (const list< **FoldDetails** > &details, list< **FSAEdge** > &path, int init\_state)  
*Function allows to lift a path in the folded FSA (p. 216) up to unfolded one.*
- int **newState** ()  
*Add a new state to a FSA (p. 216).*
- void **eraseState** (int state)  
*Erase a state from a FSA (p. 216).*
- void **newEdge** (int state1, int state2, int label)  
*Add a new edge to a FSA (p. 216).*
- void **eraseEdge** (int state1, int state2, int label)  
*Erase an edge from a FSA (p. 216).*

- `template<class ConstIntIterator >`  
`void addLoop (int vert, ConstIntIterator F, ConstIntIterator L)`  
*Add a loop to a FSA (p. 216) at a vertex vert labelled with [F,L).*
- `template<class ConstIntIterator >`  
`void addRay (int vert, ConstIntIterator F, ConstIntIterator L)`  
*Add a ray to a FSA (p. 216) at a vertex vert labelled with [F,L).*
- `void addFSA (int vert1, int vert2, const FSAREP &fsa)`  
*not implemented yet*
- `const map< int, FSASState > &getStates () const`
- `map< int, FSASState > &getStates ()`
- `void makeInitial (int s)`
- `void makeTerminal (int s)`
- `void makeNonInitial (int s)`
- `void makeNonTerminal (int s)`
- `const set< int > &getInitStates () const`
- `const set< int > &getTermStates () const`

## Private Types

- `typedef FSASState::edge_type edge_type`

## Private Member Functions

- `FSAREP ()`

## Private Attributes

- `map< int, FSASState > theStates`
- `int maxState`
- `set< int > initState`
- `set< int > termStates`

## Friends

- `class FSA`

### 9.45.1 Detailed Description

Definition at line 106 of file FSARep.h.

### 9.45.2 Member Typedef Documentation

#### 9.45.2.1 `typedef FSASState::edge_type FSARep::edge_type` `[private]`

Definition at line 110 of file FSARep.h.

### 9.45.3 Constructor & Destructor Documentation

#### 9.45.3.1 `FSARep::FSARep()` `[private]`

Referenced by `clone()`.

### 9.45.4 Member Function Documentation

#### 9.45.4.1 `void FSARep::addFSA (int vert1, int vert2, const FSARep & fsa)` `[inline]`

not implemented yet

Definition at line 197 of file FSARep.h.

Referenced by `FSA::addFSA()`.

#### 9.45.4.2 `template<class ConstIntIterator > void FSARep::addLoop (int vert, ConstIntIterator F, ConstIntIterator L)` `[inline]`

Add a loop to a **FSA** (p. 216) at a vertex `vert` labelled with `[F,L]`.

Definition at line 169 of file FSARep.h.

References `newEdge()`, and `newState()`.

Referenced by `FSA::addLoop()`.

#### 9.45.4.3 `template<class ConstIntIterator > void FSARep::addRay (int vert, ConstIntIterator F, ConstIntIterator L)` `[inline]`

Add a ray to a **FSA** (p. 216) at a vertex `vert` labelled with `[F,L]`.

Definition at line 186 of file FSARep.h.



References newEdge(), and newState().

Referenced by FSA::addRay().

#### 9.45.4.4 FSAREP\* FSAREP::clone () const [inline]

Definition at line 131 of file FSAREP.h.

References FSAREP().

#### 9.45.4.5 void FSAREP::eraseEdge (int state1, int state2, int label)

Erase an edge from a FSA (p. 216).

#### 9.45.4.6 void FSAREP::eraseState (int state)

Erase a state from a FSA (p. 216).

#### 9.45.4.7 void FSAREP::fold (const set< int > \* candidates = NULL, list< FoldDetails > \* details = NULL)

Fold a FSA (p. 216). Outputs details of the fold, that will allow to unfold latter. Argument candidates is a list of vertices to check for fold (all others won't be checked). If candidates=NULL then all vertices will be checked.

#### 9.45.4.8 const set< int >& FSAREP::getInitStates () const [inline]

Definition at line 212 of file FSAREP.h.

References initStates.

#### 9.45.4.9 map< int , FSAREP::FSAREPState >& FSAREP::getStates () [inline]

Definition at line 200 of file FSAREP.h.

References theStates.

#### 9.45.4.10 const map< int , FSAREP::FSAREPState >& FSAREP::getStates () const [inline]

Definition at line 199 of file FSAREP.h.

References theStates.

Referenced by FSA::getStates().

**9.45.4.11** `const set< int >& FSARep::getTermStates () const [inline]`

Definition at line 213 of file FSAREp.h.

References termStates.

**9.45.4.12** `void FSARep::liftup (const list< FoldDetails > & details, list< FSAEdge > & path, int init_state)`

Function allows to lift a path in the folded FSA (p.216) up to unfolded one.

**9.45.4.13** `void FSARep::makeInitial (int s) [inline]`

Definition at line 208 of file FSAREp.h.

References initStates.

**9.45.4.14** `void FSARep::makeNonInitial (int s) [inline]`

Definition at line 210 of file FSAREp.h.

References initStates.

**9.45.4.15** `void FSARep::makeNonTerminal (int s) [inline]`

Definition at line 211 of file FSAREp.h.

References termStates.

**9.45.4.16** `void FSARep::makeTerminal (int s) [inline]`

Definition at line 209 of file FSAREp.h.

References termStates.

**9.45.4.17** `void FSARep::newEdge (int state1, int state2, int label)`

Add a new edge to a FSA (p.216).

Referenced by addLoop(), and addRay().

**9.45.4.18 int FSAREP::newState ()**

Add a new state to a FSA (p. 216).

Referenced by addLoop(), and addRay().

**9.45.4.19 void FSAREP::pinch (int *state1*, int *state2*)**

Pinch two vertices.

**9.45.4.20 void FSAREP::unfold (const list< FoldDetails > & *details*)**

Unfold 2 vertices using details of the previous fold.

**9.45.5 Friends And Related Function Documentation****9.45.5.1 friend class FSA [friend]**

Definition at line 108 of file FSAREP.h.

**9.45.6 Member Data Documentation****9.45.6.1 set< int > FSAREP::initStates [private]**

Definition at line 230 of file FSAREP.h.

Referenced by getInitStates(), makeInitial(), and makeNonInitial().

**9.45.6.2 int FSAREP::maxState [private]**

Definition at line 228 of file FSAREP.h.

**9.45.6.3 set< int > FSAREP::termStates [private]**

Definition at line 232 of file FSAREP.h.

Referenced by getTermStates(), makeNonTerminal(), and makeTerminal().

**9.45.6.4 map< int , FSASState > FSAREP::theStates [private]**

Definition at line 225 of file FSAREP.h.

Referenced by `getStates()`.

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

- `Graph/include/FSARep.h`

## 9.46 FSAStruct Reference

```
#include <FSAStruct.h>
```

### Public Types

- typedef **FSAEdge** **edge\_type**

### Public Member Functions

- **FSAStruct** (int s)
- **FSAStruct** ()
- bool **operator==** (const **FSAStruct** &s) const
- bool **operator<** (const **FSAStruct** &s) const

### Public Attributes

- set< **edge\_type** > **in**
- set< **edge\_type** > **out**
- int **theState**

#### 9.46.1 Detailed Description

Definition at line 62 of file FSAStruct.h.

#### 9.46.2 Member Typedef Documentation

##### 9.46.2.1 typedef FSAEdge FSAStruct::edge\_type

Definition at line 64 of file FSAStruct.h.

#### 9.46.3 Constructor & Destructor Documentation

##### 9.46.3.1 FSAStruct::FSAStruct (int s) [inline]

Definition at line 66 of file FSAStruct.h.

##### 9.46.3.2 FSAStruct::FSAStruct () [inline]

Definition at line 67 of file FSAStruct.h.

## 9.46.4 Member Function Documentation

### 9.46.4.1 `bool FSASState::operator< (const FSASState & s) const` `[inline]`

Definition at line 73 of file FSASRep.h.

References `theState`.

### 9.46.4.2 `bool FSASState::operator==(const FSASState & s) const` `[inline]`

Definition at line 70 of file FSASRep.h.

References `theState`.

## 9.46.5 Member Data Documentation

### 9.46.5.1 `set< edge_type > FSASState::in`

Definition at line 77 of file FSASRep.h.

### 9.46.5.2 `set< edge_type > FSASState::out`

Definition at line 78 of file FSASRep.h.

### 9.46.5.3 `int FSASState::theState`

Definition at line 79 of file FSASRep.h.

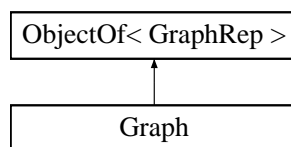
Referenced by `operator<()`, and `operator==(())`.

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

- `Graph/include/FSASRep.h`

## 9.47 Graph Class Reference

`#include <Graph.h>` Inheritance diagram for `Graph::`



### Public Types

- `typedef GraphRep::state_type state_type`
- `typedef GraphRep::edge_type edge_type`

### Public Member Functions

- `Graph ()`
- `const map< int, state_type > & getStates () const`
- `int newState ()`
- `void newEdge (int v1, int v2)`
- `void clear ()`

#### 9.47.1 Detailed Description

Definition at line 25 of file `Graph.h`.

#### 9.47.2 Member Typedef Documentation

##### 9.47.2.1 `typedef GraphRep::edge_type Graph::edge_type`

Definition at line 30 of file `Graph.h`.

##### 9.47.2.2 `typedef GraphRep::state_type Graph::state_type`

Definition at line 29 of file `Graph.h`.

### 9.47.3 Constructor & Destructor Documentation

#### 9.47.3.1 `Graph::Graph () [inline]`

Definition at line 39 of file Graph.h.

### 9.47.4 Member Function Documentation

#### 9.47.4.1 `void Graph::clear () [inline]`

Definition at line 51 of file Graph.h.

References `ObjectOf< GraphRep >::change()`, and `clear()`.

Referenced by `WhiteheadGraph::assign()`, and `clear()`.

#### 9.47.4.2 `const map< int, state_type >& Graph::getStates () const [inline]`

Definition at line 48 of file Graph.h.

References `GraphRep::getStates()`, and `ObjectOf< GraphRep >::look()`.

Referenced by `getDistances_out()`, `getGeodesicTree_in()`, and `getGeodesicTree_out()`.

#### 9.47.4.3 `void Graph::newEdge (int v1, int v2) [inline]`

Definition at line 50 of file Graph.h.

References `ObjectOf< GraphRep >::change()`, and `newEdge()`.

Referenced by `WhiteheadGraph::assign()`, and `newEdge()`.

#### 9.47.4.4 `int Graph::newState () [inline]`

Definition at line 49 of file Graph.h.

References `ObjectOf< GraphRep >::change()`, and `newState()`.

Referenced by `newState()`.

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

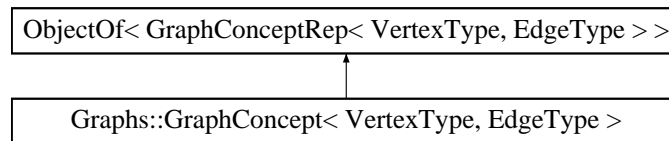
- `Graph/include/Graph.h`



## 9.48 Graphs::GraphConcept< VertexType, EdgeType > Class Template Reference

The main class for graph types.

#include <GraphConcept.h>Inheritance diagram for  
Graphs::GraphConcept< VertexType, EdgeType >::



### Public Types

- typedef VertexType **vertex\_type**  
*The type of the graph vertex.*
- typedef EdgeType **edge\_type**  
*The type of the graph edge.*

### Public Member Functions

- **GraphConcept ()**  
*Create empty graph.*
- const map< int, **vertex\_type** > & **getVertices ()** const  
*Get the vertex set.*
- int **newVertex ()**  
*Create new vertex (default). Function returns the number of the new vertex.*
- int **newVertex** (const **vertex\_type** &V)  
*Add new vertex into a graph. Function returns the number of the new vertex.*
- void **eraseVertex** (int v)  
*Erase a vertex from the graph.*
- void **replaceVertex** (int v, const **vertex\_type** &V)

*Function locally changes graph by replacing the vertex with number  $v$  by the vertex  $V$ .*

- void **newEdge** (int v1, const EdgeType &E)

*Create new edge with the origin v1 and target [and, maybe other parameters] provided in E.*

- void **clear** ()

*Make the graph trivial.*

- void **pinch** (int v1, int v2)

*Pinch two vertices v1 and v2.*

## Private Types

- typedef **GraphConceptRep**< VertexType, EdgeType > **presentation\_type**

### 9.48.1 Detailed Description

```
template<class VertexType, class EdgeType> class Graphs::GraphConcept<
VertexType, EdgeType >
```

The main class for graph types.

Definition at line 274 of file GraphConcept.h.

### 9.48.2 Member Typedef Documentation

**9.48.2.1**    template<class VertexType, class EdgeType> typedef EdgeType  
Graphs::GraphConcept< VertexType, EdgeType >::edge\_type

The type of the graph edge.

Definition at line 282 of file GraphConcept.h.

**9.48.2.2**    template<class VertexType, class EdgeType> typedef  
GraphConceptRep< VertexType , EdgeType >  
Graphs::GraphConcept< VertexType, EdgeType >::presentation\_type  
[private]

Definition at line 286 of file GraphConcept.h.

**9.48.2.3    template<class VertexType, class EdgeType> typedef VertexType  
             Graphs::GraphConcept< VertexType, EdgeType >::vertex\_type**

The type of the graph vertex.

Definition at line 280 of file GraphConcept.h.

**9.48.3    Constructor & Destructor Documentation**

**9.48.3.1    template<class VertexType, class EdgeType>  
             Graphs::GraphConcept< VertexType, EdgeType >::GraphConcept ()  
             [inline]**

Create empty graph.

Definition at line 296 of file GraphConcept.h.

**9.48.4    Member Function Documentation**

**9.48.4.1    template<class VertexType, class EdgeType> void  
             Graphs::GraphConcept< VertexType, EdgeType >::clear ()  
             [inline]**

Make the graph trivial.

Definition at line 331 of file GraphConcept.h.

Referenced by Graphs::GraphConcept< GraphVertex< IntLabeledEdge >, IntLabeledEdge >::clear().

**9.48.4.2    template<class VertexType, class EdgeType> void  
             Graphs::GraphConcept< VertexType, EdgeType >::eraseVertex (int  
             v) [inline]**

Erase a vertex from the graph.

Definition at line 319 of file GraphConcept.h.

Referenced by Graphs::GraphConcept< GraphVertex< IntLabeledEdge >, IntLabeledEdge >::eraseVertex().

**9.48.4.3    template<class VertexType, class EdgeType> const map< int,  
             vertex\_type >& Graphs::GraphConcept< VertexType, EdgeType  
             >::getVertices () const [inline]**

Get the vertex set.

Definition at line 307 of file GraphConcept.h.

**9.48.4.4** `template<class VertexType, class EdgeType> void  
Graphs::GraphConcept< VertexType, EdgeType >::newEdge (int vI,  
const EdgeType & E) [inline]`

Create new edge with the origin vI and target [and, maybe other parameters] provided in E.

Definition at line 327 of file GraphConcept.h.

Referenced by addLoop(), addRay(), and Graphs::GraphConcept< GraphVertex< IntLabeledEdge >, IntLabeledEdge >::newEdge().

**9.48.4.5** `template<class VertexType, class EdgeType> int  
Graphs::GraphConcept< VertexType, EdgeType >::newVertex (const  
vertex_type & V) [inline]`

Add new vertex into a graph. Function returns the number of the new vertex.

Definition at line 315 of file GraphConcept.h.

Referenced by Graphs::GraphConcept< GraphVertex< IntLabeledEdge >, IntLabeledEdge >::newVertex().

**9.48.4.6** `template<class VertexType, class EdgeType> int  
Graphs::GraphConcept< VertexType, EdgeType >::newVertex ()  
[inline]`

Create new vertex (default). Function returns the number of the new vertex.

Definition at line 311 of file GraphConcept.h.

Referenced by addLoop(), addRay(), and Graphs::GraphConcept< GraphVertex< IntLabeledEdge >, IntLabeledEdge >::newVertex().

**9.48.4.7** `template<class VertexType, class EdgeType> void  
Graphs::GraphConcept< VertexType, EdgeType >::pinch (int vI, int  
v2) [inline]`

Pinch two vertices vI and v2.

Definition at line 335 of file GraphConcept.h.

Referenced by Graphs::GraphConcept< GraphVertex< IntLabeledEdge >, IntLabeledEdge >::pinch().

**9.48.4.8** `template<class VertexType, class EdgeType> void  
Graphs::GraphConcept< VertexType, EdgeType >::replaceVertex (int  
v, const vertex_type & V) [inline]`

Function locally changes graph by replacing the vertex with number v by the vertex V.

Definition at line 323 of file GraphConcept.h.

Referenced by Graphs::GraphConcept< GraphVertex< IntLabeledEdge >, IntLabeledEdge >::replaceVertex().

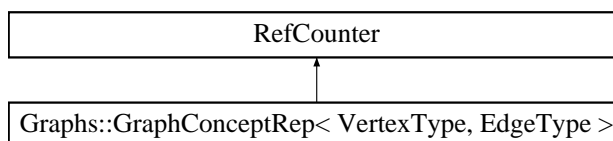
The documentation for this class was generated from the following file:

- Graph/include/**GraphConcept.h**

## 9.49 Graphs::GraphConceptRep< VertexType, EdgeType > Class Template Reference

Representation class for graph types.

```
#include <GraphConcept.h> Inheritance diagram for
Graphs::GraphConceptRep< VertexType, EdgeType >::
```



### Public Types

- typedef VertexType **vertex\_type**
- typedef EdgeType **edge\_type**

### Public Member Functions

- **GraphConceptRep** \* **clone** () const

### Private Member Functions

- **GraphConceptRep** ()
- const map< int, **vertex\_type** > & **getVertices** () const  
*Get a set of all vertices.*
- int **newVertex** ()  
*Add a new vertex to a graph disconnected from all others.*
- int **newVertex** (const **vertex\_type** &V)  
*Add a new vertex to a graph. All edges in "in" and "out" sets will be incorporated to the graph.*
- void **replaceVertex** (int v, const **vertex\_type** &V)  
*Replace a vertex v with V. If v does not exist then V simply becomes a vertex with the number v.*
- void **eraseVertex** (int v)

---

*Erase a vertex from a graph.*

- void **newEdge** (int v, const **edge\_type** &E)  
*Add a new edge to a graph.*
- void **eraseVertex** (int v, const **edge\_type** &E)  
*Erase an edge from a graph.*
- void **clear** ()  
*Clear a graph.*
- void **pinch** (int v1, int v2)
- int **\_newVertex** (int v, const **vertex\_type** &V)  
*This function assumes that the vertex v does not exist and V has a correct structure to be incorporated to a graph.*

## Private Attributes

- map< int, **vertex\_type** > **theVertices**
- int **maxVertex**  
*The maximal unused number of a vertex (used in newVertex).*

## Friends

- class **GraphConcept**< **vertex\_type**, **edge\_type** >

### 9.49.1 Detailed Description

**template<class VertexType, class EdgeType> class Graphs::GraphConceptRep< VertexType, EdgeType >**

Representation class for graph types.

Definition at line 47 of file GraphConcept.h.

### 9.49.2 Member Typedef Documentation

**9.49.2.1    template<class VertexType , class EdgeType > typedef EdgeType  
             Graphs::GraphConceptRep< VertexType, EdgeType >::edge\_type**

Definition at line 57 of file GraphConcept.h.

**9.49.2.2** `template<class VertexType , class EdgeType > typedef VertexType  
Graphs::GraphConceptRep< VertexType, EdgeType >::vertex_type`

Definition at line 56 of file GraphConcept.h.

### 9.49.3 Constructor & Destructor Documentation

**9.49.3.1** `template<class VertexType , class EdgeType >  
Graphs::GraphConceptRep< VertexType, EdgeType  
>::GraphConceptRep () [inline, private]`

Definition at line 68 of file GraphConcept.h.

Referenced by `Graphs::GraphConceptRep< VertexType, EdgeType >::clone()`.

### 9.49.4 Member Function Documentation

**9.49.4.1** `template<class VertexType , class EdgeType > int  
Graphs::GraphConceptRep< VertexType, EdgeType >::_newVertex  
(int v, const vertex_type & V) [inline, private]`

This function assumes that the vertex `v` does not exist and `V` has a correct structure to be incorporated to a graph.

Definition at line 213 of file GraphConcept.h.

References `Graphs::GraphConceptRep< VertexType, EdgeType >::theVertices`.

Referenced by `Graphs::GraphConceptRep< VertexType, EdgeType >::newVertex()`, and `Graphs::GraphConceptRep< VertexType, EdgeType >::replaceVertex()`.

**9.49.4.2** `template<class VertexType , class EdgeType > void  
Graphs::GraphConceptRep< VertexType, EdgeType >::clear ()  
[inline, private]`

Clear a graph.

Definition at line 152 of file GraphConcept.h.

References `Graphs::GraphConceptRep< VertexType, EdgeType >::maxVertex`, and `Graphs::GraphConceptRep< VertexType, EdgeType >::theVertices`.



**9.49.4.3** `template<class VertexType , class EdgeType > GraphConceptRep*  
Graphs::GraphConceptRep< VertexType, EdgeType >::clone () const  
[inline]`

Definition at line 78 of file GraphConcept.h.

References `Graphs::GraphConceptRep< VertexType, EdgeType >::GraphConceptRep()`.

**9.49.4.4** `template<class VertexType , class EdgeType > void  
Graphs::GraphConceptRep< VertexType, EdgeType >::eraseVertex  
(int v, const edge_type & E) [inline, private]`

Erase an edge from a graph.

Definition at line 142 of file GraphConcept.h.

References `Graphs::GraphConceptRep< VertexType, EdgeType >::theVertices`.

**9.49.4.5** `template<class VertexType , class EdgeType > void  
Graphs::GraphConceptRep< VertexType, EdgeType >::eraseVertex  
(int v) [inline, private]`

Erase a vertex from a graph.

Definition at line 107 of file GraphConcept.h.

References `Graphs::GraphConceptRep< VertexType, EdgeType >::theVertices`.

Referenced by `Graphs::GraphConceptRep< VertexType, EdgeType >::replaceVertex()`.

**9.49.4.6** `template<class VertexType , class EdgeType > const map< int,  
vertex_type >& Graphs::GraphConceptRep< VertexType, EdgeType  
>::getVertices () const [inline, private]`

Get a set of all vertices.

Definition at line 88 of file GraphConcept.h.

References `Graphs::GraphConceptRep< VertexType, EdgeType >::theVertices`.

**9.49.4.7** `template<class VertexType , class EdgeType > void  
Graphs::GraphConceptRep< VertexType, EdgeType >::newEdge (int  
v, const edge_type & E) [inline, private]`

Add a new edge to a graph.

Definition at line 133 of file GraphConcept.h.

References Graphs::GraphConceptRep< VertexType, EdgeType >::theVertices.

Referenced by Graphs::GraphConceptRep< VertexType, EdgeType >::pinch().

**9.49.4.8** `template<class VertexType , class EdgeType > int  
Graphs::GraphConceptRep< VertexType, EdgeType >::newVertex  
(const vertex_type & V) [inline, private]`

Add a new vertex to a graph. All edges in "in" and "out" sets will be incorporated to the graph.

Definition at line 96 of file GraphConcept.h.

References Graphs::GraphConceptRep< VertexType, EdgeType >::\_newVertex(), and Graphs::GraphConceptRep< VertexType, EdgeType >::maxVertex.

**9.49.4.9** `template<class VertexType , class EdgeType > int  
Graphs::GraphConceptRep< VertexType, EdgeType >::newVertex ()  
[inline, private]`

Add a new vertex to a graph disconnected from all others.

Definition at line 91 of file GraphConcept.h.

References Graphs::GraphConceptRep< VertexType, EdgeType >::maxVertex, and Graphs::GraphConceptRep< VertexType, EdgeType >::theVertices.

**9.49.4.10** `template<class VertexType , class EdgeType > void  
Graphs::GraphConceptRep< VertexType, EdgeType >::pinch (int  
v1, int v2) [inline, private]`

Definition at line 157 of file GraphConcept.h.

References Graphs::GraphConceptRep< VertexType, EdgeType >::newEdge(), and Graphs::GraphConceptRep< VertexType, EdgeType >::theVertices.

**9.49.4.11** `template<class VertexType , class EdgeType > void  
Graphs::GraphConceptRep< VertexType, EdgeType  
>::replaceVertex (int v, const vertex_type & V) [inline,  
private]`

Replace a vertex v with V. If v does not exist then V simply becomes a vertex with the number v.

Definition at line 101 of file GraphConcept.h.

## 9.49 Graphs::GraphConceptRep< VertexType, EdgeType > Class Template Reference 245

---

References Graphs::GraphConceptRep< VertexType, EdgeType >::\_newVertex(), and Graphs::GraphConceptRep< VertexType, EdgeType >::eraseVertex().

### 9.49.5 Friends And Related Function Documentation

#### 9.49.5.1 `template<class VertexType , class EdgeType > friend class GraphConcept< vertex_type, edge_type > [friend]`

Definition at line 59 of file GraphConcept.h.

### 9.49.6 Member Data Documentation

#### 9.49.6.1 `template<class VertexType , class EdgeType > int Graphs::GraphConceptRep< VertexType, EdgeType >::maxVertex [private]`

The maximal unused number of a vertex (used in newVertex).

Definition at line 256 of file GraphConcept.h.

Referenced by Graphs::GraphConceptRep< VertexType, EdgeType >::clear(), and Graphs::GraphConceptRep< VertexType, EdgeType >::newVertex().

#### 9.49.6.2 `template<class VertexType , class EdgeType > map< int, vertex_type > Graphs::GraphConceptRep< VertexType, EdgeType >::theVertices [private]`

Definition at line 253 of file GraphConcept.h.

Referenced by Graphs::GraphConceptRep< VertexType, EdgeType >::\_newVertex(), Graphs::GraphConceptRep< VertexType, EdgeType >::clear(), Graphs::GraphConceptRep< VertexType, EdgeType >::eraseVertex(), Graphs::GraphConceptRep< VertexType, EdgeType >::getVertices(), Graphs::GraphConceptRep< VertexType, EdgeType >::newEdge(), Graphs::GraphConceptRep< VertexType, EdgeType >::newVertex(), and Graphs::GraphConceptRep< VertexType, EdgeType >::pinch().

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

- Graph/include/GraphConcept.h

## 9.50 GraphDrawingAttributes Class Reference

```
#include <GraphDrawingAttributes.h>
```

### Public Types

- enum **NODESHAPE** {  
    **box**, **polygon**, **ellipse**, **circle**,  
    **point**, **egg**, **triangle**, **plaintext**,  
    **diamond**, **trapezium**, **parallelogram**, **house**,  
    **pentagon**, **hexagon**, **septagon**, **octagon**,  
    **doublecircle**, **doubleoctagon**, **tripleoctagon**, **invtriangle**,  
    **invtrapezium**, **invhouse**, **Mdiamond**, **Msquare**,  
    **Mcircle**, **rect**, **rectangle**, **none** }  
• typedef **triple**< int, int, int > **COLOR**

### Public Member Functions

- **GraphDrawingAttributes** ()
- void **setNodeColor** (int v, const **COLOR** &c)  
    *Set the color for a node.*
- **COLOR** **getNodeColor** (int v) const  
    *Get the color of a node.*
- void **setNodeShape** (int v, const **NODESHAPE** &s)  
    *Set the shape for a node.*
- **NODESHAPE** **getNodeShape** (int v) const  
    *Get the shape of a node.*
- void **setDefaultNodeShape** (const **NODESHAPE** &s)

### Static Public Member Functions

- static const map< int, string > & **getNodeShapeNames** ()

### Static Private Member Functions

- static map< int, string > **initializeNodeShapeNames** ()

## Private Attributes

- **COLOR** theDefaultNodeColor
- map< int, **COLOR** > **nodeColor**
- **NODESHAPE** theDefaultNodeShape
- map< int, **NODESHAPE** > **nodeShape**

## Static Private Attributes

- static map< int, string > **nodeShapeNames**

### 9.50.1 Detailed Description

Definition at line 22 of file GraphDrawingAttributes.h.

### 9.50.2 Member Typedef Documentation

#### 9.50.2.1 typedef triple< int , int , int > GraphDrawingAttributes::COLOR

Definition at line 32 of file GraphDrawingAttributes.h.

### 9.50.3 Member Enumeration Documentation

#### 9.50.3.1 enum GraphDrawingAttributes::NODESHAPE

Enumerator:

*box*  
*polygon*  
*ellipse*  
*circle*  
*point*  
*egg*  
*triangle*  
*plaintext*  
*diamond*  
*trapezium*  
*parallelogram*  
*house*

*pentagon*

*hexagon*

*septagon*

*octagon*

*doublecircle*

*doubleoctagon*

*tripleoctagon*

*invtriangle*

*invtrapezium*

*invhouse*

*Mdiamond*

*Msquare*

*Mcircle*

*rect*

*rectangle*

*none*

Definition at line 33 of file GraphDrawingAttributes.h.

## 9.50.4 Constructor & Destructor Documentation

### 9.50.4.1 GraphDrawingAttributes::GraphDrawingAttributes () [inline]

Definition at line 46 of file GraphDrawingAttributes.h.

## 9.50.5 Member Function Documentation

### 9.50.5.1 COLOR GraphDrawingAttributes::getNodeColor (int *v*) const [inline]

Get the color of a node.

Definition at line 61 of file GraphDrawingAttributes.h.

References `nodeColor`, and `theDefaultNodeColor`.

**9.50.5.2 NODESHAPE GraphDrawingAttributes::getNodeShape (int *v*) const [inline]**

Get the shape of a node.

Definition at line 70 of file GraphDrawingAttributes.h.

References nodeShape, and theDefaultNodeShape.

**9.50.5.3 static const map< int , string >& GraphDrawingAttributes::getNodeShapeNames () [inline, static]**

Definition at line 78 of file GraphDrawingAttributes.h.

References nodeShapeNames.

**9.50.5.4 static map< int , string > GraphDrawingAttributes::initializeNodeShapeNames () [static, private]****9.50.5.5 void GraphDrawingAttributes::setDefaultNodeShape (const NODESHAPE & *s*) [inline]**

Definition at line 76 of file GraphDrawingAttributes.h.

References theDefaultNodeShape.

**9.50.5.6 void GraphDrawingAttributes::setNodeColor (int *v*, const COLOR & *c*) [inline]**

Set the color for a node.

Definition at line 59 of file GraphDrawingAttributes.h.

References nodeColor.

**9.50.5.7 void GraphDrawingAttributes::setNodeShape (int *v*, const NODESHAPE & *s*) [inline]**

Set the shape for a node.

Definition at line 68 of file GraphDrawingAttributes.h.

References nodeShape.

## 9.50.6 Member Data Documentation

### 9.50.6.1 `map< int , COLOR > GraphDrawingAttributes::nodeColor` `[private]`

Definition at line 100 of file `GraphDrawingAttributes.h`.

Referenced by `getNodeColor()`, and `setNodeColor()`.

### 9.50.6.2 `map< int , NODESHAPE > GraphDrawingAttributes::nodeShape` `[private]`

Definition at line 106 of file `GraphDrawingAttributes.h`.

Referenced by `getNodeShape()`, and `setNodeShape()`.

### 9.50.6.3 `map< int , string > GraphDrawingAttributes::nodeShapeNames` `[static, private]`

Definition at line 102 of file `GraphDrawingAttributes.h`.

Referenced by `getNodeShapeNames()`.

### 9.50.6.4 `COLOR GraphDrawingAttributes::theDefaultNodeColor` `[private]`

Definition at line 98 of file `GraphDrawingAttributes.h`.

Referenced by `getNodeColor()`.

### 9.50.6.5 `NODESHAPE GraphDrawingAttributes::theDefaultNodeShape` `[private]`

Definition at line 104 of file `GraphDrawingAttributes.h`.

Referenced by `getNodeShape()`, and `setDefaultNodeShape()`.

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

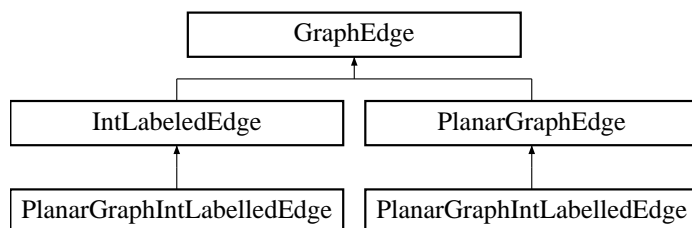
- `Graph/include/GraphDrawingAttributes.h`



## 9.51 GraphEdge Struct Reference

Defines non-labelled edge of the directed graph.

`#include <GraphRep.h>` Inheritance diagram for `GraphEdge`::



### Public Member Functions

- **GraphEdge** ()
- **GraphEdge** (int t)
- bool **operator<** (const **GraphEdge** &e) const
- bool **operator==** (const **GraphEdge** &e) const
- **GraphEdge** (int t)

*Constructor for an edge. Argument t is a number of the target.*

- **GraphEdge** ()

*Dummy constructor, not to be used, required for STL::map. Also, can be used to denote "failure" or "dead-end" edge.*

- **GraphEdge** **inverse** (int origin)

*Invert an edge.*

- bool **operator<** (const **GraphEdge** &e) const

*Check if one edge is less than the other.*

- bool **operator==** (const **GraphEdge** &e) const

*Check if two edges equal.*

- bool **operator!=** (const **GraphEdge** &e) const

*Check if two edges are not equal.*

## Public Attributes

- int **target**
- int **theTarget**

*The target of the edge (for out edges), or origin of the vertex (for in edges).*

### 9.51.1 Detailed Description

Defines non-labelled edge of the directed graph. The order on edges must be defined so that the edges with the least labels go first for folding to be correct.

Definition at line 26 of file GraphRep.h.

### 9.51.2 Constructor & Destructor Documentation

#### 9.51.2.1 GraphEdge::GraphEdge () [inline]

Definition at line 29 of file GraphRep.h.

Referenced by inverse().

#### 9.51.2.2 GraphEdge::GraphEdge (int *t*) [inline]

Definition at line 32 of file GraphRep.h.

#### 9.51.2.3 GraphEdge::GraphEdge (int *t*) [inline]

Constructor for an edge. Argument *t* is a number of the target.

Definition at line 41 of file GraphType.h.

#### 9.51.2.4 GraphEdge::GraphEdge () [inline]

Dummy constructor, not to be used, required for STL::map. Also, can be used to denote "failure" or "dead-end" edge.

Definition at line 44 of file GraphType.h.

### 9.51.3 Member Function Documentation

#### 9.51.3.1 GraphEdge GraphEdge::inverse (int *origin*) [inline]

Invert an edge.

Reimplemented in **IntLabeledEdge** (p. 278), **PlanarGraphEdge** (p. 389), and **PlanarGraphIntLabelledEdge** (p. 393).

Definition at line 49 of file GraphType.h.

References GraphEdge().

#### 9.51.3.2 bool GraphEdge::operator!= (const GraphEdge & *e*) const [inline]

Check if two edges are not equal.

Reimplemented in **IntLabeledEdge** (p. 278), **PlanarGraphEdge** (p. 390), and **PlanarGraphIntLabelledEdge** (p. 393).

Definition at line 61 of file GraphType.h.

References theTarget.

#### 9.51.3.3 bool GraphEdge::operator< (const GraphEdge & *e*) const [inline]

Check if one edge is less than the other.

Reimplemented in **IntLabeledEdge** (p. 279), **PlanarGraphEdge** (p. 390), and **PlanarGraphIntLabelledEdge** (p. 394).

Definition at line 53 of file GraphType.h.

References theTarget.

#### 9.51.3.4 bool GraphEdge::operator< (const GraphEdge & *e*) const [inline]

Reimplemented in **IntLabeledEdge** (p. 279), **PlanarGraphEdge** (p. 390), and **PlanarGraphIntLabelledEdge** (p. 394).

Definition at line 34 of file GraphRep.h.

References target.

### 9.51.3.5 **bool GraphEdge::operator==(const GraphEdge & e) const** [inline]

Check if two edges equal.

Reimplemented in **IntLabeledEdge** (p. 279), **PlanarGraphEdge** (p. 390), and **PlanarGraphIntLabelledEdge** (p. 394).

Definition at line 57 of file GraphType.h.

References theTarget.

### 9.51.3.6 **bool GraphEdge::operator==(const GraphEdge & e) const** [inline]

Reimplemented in **IntLabeledEdge** (p. 279), **PlanarGraphEdge** (p. 390), and **PlanarGraphIntLabelledEdge** (p. 394).

Definition at line 35 of file GraphRep.h.

References target.

## 9.51.4 Member Data Documentation

### 9.51.4.1 **int GraphEdge::target**

Definition at line 37 of file GraphRep.h.

Referenced by operator<(), and operator==(()).

### 9.51.4.2 **int GraphEdge::theTarget**

The target of the edge (for out edges), or origin of the vertex (for in edges).

Definition at line 65 of file GraphType.h.

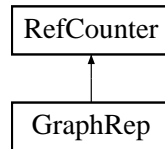
Referenced by PlanarGraphIntLabelledEdge::operator!=(()), PlanarGraphEdge::operator!=(()), operator!=(()), PlanarGraphIntLabelledEdge::operator<(), PlanarGraphEdge::operator<(), IntLabeledEdge::operator<(), operator<(), PlanarGraphIntLabelledEdge::operator==(()), PlanarGraphEdge::operator==(()), and operator==(()).

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

- Graph/include/**GraphRep.h**
- Graph/include/**GraphType.h**

## 9.52 GraphRep Class Reference

`#include <GraphRep.h>`Inheritance diagram for GraphRep::



### Public Types

- typedef **GraphEdge** **edge\_type**
- typedef **GraphState** **state\_type**

### Public Member Functions

- **GraphRep** \* **clone** () const

### Private Member Functions

- **GraphRep** ()
- int **newState** ()
- void **newEdge** (int v1, int v2)
- void **clear** ()
- const map< int, **state\_type** > & **getStates** () const

### Private Attributes

- map< int, **state\_type** > **theStates**
- int **maxState**

### Friends

- class **Graph**  
*Directed Graph* (p. 233).

### 9.52.1 Detailed Description

Definition at line 72 of file GraphRep.h.

### 9.52.2 Member Typedef Documentation

#### 9.52.2.1 `typedef GraphEdge GraphRep::edge_type`

Definition at line 76 of file GraphRep.h.

#### 9.52.2.2 `typedef GraphState GraphRep::state_type`

Definition at line 77 of file GraphRep.h.

### 9.52.3 Constructor & Destructor Documentation

#### 9.52.3.1 `GraphRep::GraphRep () [inline, private]`

Definition at line 89 of file GraphRep.h.

Referenced by clone().

### 9.52.4 Member Function Documentation

#### 9.52.4.1 `void GraphRep::clear () [private]`

#### 9.52.4.2 `GraphRep* GraphRep::clone () const [inline]`

Definition at line 107 of file GraphRep.h.

References GraphRep().

#### 9.52.4.3 `const map< int, state_type >& GraphRep::getStates () const [inline, private]`

Definition at line 115 of file GraphRep.h.

References theStates.

Referenced by Graph::getStates().

**9.52.4.4** void GraphRep::newEdge (int *v1*, int *v2*) [private]

**9.52.4.5** int GraphRep::newState () [private]

## **9.52.5 Friends And Related Function Documentation**

**9.52.5.1** friend class Graph [friend]

Directed **Graph** (p. 233).

Definition at line 79 of file GraphRep.h.

## **9.52.6 Member Data Documentation**

**9.52.6.1** int GraphRep::maxState [private]

Definition at line 134 of file GraphRep.h.

**9.52.6.2** map< int, state\_type > GraphRep::theStates [private]

Definition at line 131 of file GraphRep.h.

Referenced by getStates().

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

- Graph/include/**GraphRep.h**

## 9.53 GraphState Struct Reference

```
#include <GraphRep.h>
```

### Public Types

- typedef **GraphEdge** **edge\_type**

### Public Member Functions

- **GraphState** (int *s*)
- **GraphState** ()
- bool **operator==** (const **GraphState** &*s*) const
- bool **operator<** (const **GraphState** &*s*) const

### Public Attributes

- set< **edge\_type** > **in**
- set< **edge\_type** > **out**
- int **theState**

#### 9.53.1 Detailed Description

Definition at line 46 of file GraphRep.h.

#### 9.53.2 Member Typedef Documentation

##### 9.53.2.1 typedef GraphEdge GraphState::edge\_type

Definition at line 48 of file GraphRep.h.

#### 9.53.3 Constructor & Destructor Documentation

##### 9.53.3.1 GraphState::GraphState (int *s*) [inline]

Definition at line 50 of file GraphRep.h.

##### 9.53.3.2 GraphState::GraphState () [inline]

Definition at line 51 of file GraphRep.h.



## 9.53.4 Member Function Documentation

### 9.53.4.1 `bool GraphState::operator< (const GraphState & s) const` `[inline]`

Definition at line 57 of file GraphRep.h.

References `theState`.

### 9.53.4.2 `bool GraphState::operator== (const GraphState & s) const` `[inline]`

Definition at line 54 of file GraphRep.h.

References `theState`.

## 9.53.5 Member Data Documentation

### 9.53.5.1 `set< edge_type > GraphState::in`

Definition at line 61 of file GraphRep.h.

### 9.53.5.2 `set< edge_type > GraphState::out`

Definition at line 62 of file GraphRep.h.

### 9.53.5.3 `int GraphState::theState`

Definition at line 63 of file GraphRep.h.

Referenced by `operator<()`, and `operator==()`.

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

- Graph/include/**GraphRep.h**

## 9.54 GraphVertex< EdgeType > Struct Template Reference

**Graph** (p. 233) vertex.

```
#include <GraphType.h>
```

### Public Types

- typedef EdgeType **edge\_type**  
*Type of graph edges.*

### Public Member Functions

- **GraphVertex** ()  
*Default vertex.*

### Public Attributes

- set< **edge\_type** > **in**  
*Incoming edges.*
- set< **edge\_type** > **out**  
*Leaving edges.*

#### 9.54.1 Detailed Description

```
template<class EdgeType> struct GraphVertex< EdgeType >
```

**Graph** (p. 233) vertex.

Definition at line 78 of file GraphType.h.

#### 9.54.2 Member Typedef Documentation

**9.54.2.1** template<class EdgeType > typedef EdgeType GraphVertex< EdgeType >::edge\_type

Type of graph edges.

Definition at line 81 of file GraphType.h.

### 9.54.3 Constructor & Destructor Documentation

#### 9.54.3.1 `template<class EdgeType > GraphVertex< EdgeType >::GraphVertex () [inline]`

Default vertex.

Definition at line 85 of file GraphType.h.

### 9.54.4 Member Data Documentation

#### 9.54.4.1 `template<class EdgeType > set< edge_type > GraphVertex< EdgeType >::in`

Incoming edges.

Definition at line 89 of file GraphType.h.

#### 9.54.4.2 `template<class EdgeType > set< edge_type > GraphVertex< EdgeType >::out`

Leaving edges.

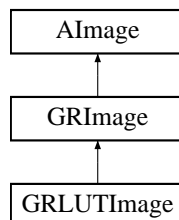
Definition at line 93 of file GraphType.h.

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

- Graph/include/**GraphType.h**

## 9.55 GRImage Class Reference

#include <AImage.h> Inheritance diagram for GRImage::



### Public Member Functions

- **GRImage** ()
- **GRImage** (int w, int h)
- **GRImage** (const string &in\_file\_name)
- **GRImage** (const **GRImage** &image)
- **GRImage** (const **GRImage** &, int, int, int, int, bool)
- **~GRImage** ()
- void **setPixel** (int i, int j, int v)
- virtual int **getPixel** (int i, int j) const
- void **setPixel** (int n, int v)
- void **setPixelCliped** (int n, int v)
- void **setPixelCliped** (int i, int j, int v)
- virtual int **getPixel** (int n) const
- void **saveTo** (const string &file\_name)
- **IMAGE\_TYPE** **getType** () const
- **GRImage** & **operator=** (const **GRImage** &)

### Protected Member Functions

- virtual void **printOnPGM** (ostream &out) const
- virtual void **printOnBW** (ostream &) const
- void **readFromBW** (istream &in)
- void **readFromPGM** (istream &in)

### Protected Attributes

- unsigned char \* **image\_string**

## Friends

- class **GRLUTImage**

### 9.55.1 Detailed Description

Definition at line 135 of file AImage.h.

### 9.55.2 Constructor & Destructor Documentation

#### 9.55.2.1 GRImage::GRImage () [inline]

Definition at line 139 of file AImage.h.

#### 9.55.2.2 GRImage::GRImage (int *w*, int *h*) [inline]

Definition at line 142 of file AImage.h.

References `image_string`, and `AImage::size`.

#### 9.55.2.3 GRImage::GRImage (const string & *in\_file\_name*)

#### 9.55.2.4 GRImage::GRImage (const GRImage & *image*)

#### 9.55.2.5 GRImage::GRImage (const GRImage &, int, int, int, int, bool)

#### 9.55.2.6 GRImage::~~GRImage ()

### 9.55.3 Member Function Documentation

#### 9.55.3.1 virtual int GRImage::getPixel (int *n*) const [virtual]

Reimplemented in **GRLUTImage** (p. 267).

#### 9.55.3.2 virtual int GRImage::getPixel (int *i*, int *j*) const [virtual]

Reimplemented in **GRLUTImage** (p. 267).

Referenced by `CImage::getBluePixel()`, `CImage::getGreenPixel()`, and `CImage::getRedPixel()`.

### 9.55.3.3 IMAGE\_TYPE GRImage::getType () const [inline, virtual]

Implements **AImage** (p. 101).

Definition at line 176 of file AImage.h.

References GRAY.

### 9.55.3.4 GRImage& GRImage::operator= (const GRImage &)

### 9.55.3.5 virtual void GRImage::printOnBW (ostream &) const [protected, virtual]

### 9.55.3.6 virtual void GRImage::printOnPGM (ostream & out) const [protected, virtual]

### 9.55.3.7 void GRImage::readFromBW (istream & in) [protected]

### 9.55.3.8 void GRImage::readFromPGM (istream & in) [protected]

### 9.55.3.9 void GRImage::saveTo (const string & file\_name) [virtual]

Implements **AImage** (p. 102).

### 9.55.3.10 void GRImage::setPixel (int *n*, int *v*)

### 9.55.3.11 void GRImage::setPixel (int *i*, int *j*, int *v*)

Referenced by CImage::setBluePixel(), CImage::setGreenPixel(), and CImage::setRedPixel().

### 9.55.3.12 void GRImage::setPixelClipped (int *i*, int *j*, int *v*)

### 9.55.3.13 void GRImage::setPixelClipped (int *n*, int *v*)

Referenced by CImage::setBluePixelClipped(), CImage::setGreenPixelClipped(), and CImage::setRedPixelClipped().

## 9.55.4 Friends And Related Function Documentation

### 9.55.4.1 friend class GRLUTImage [friend]

Definition at line 182 of file AImage.h.

## 9.55.5 Member Data Documentation

### 9.55.5.1 unsigned char\* GRIImage::image\_string [protected]

Definition at line 188 of file AImage.h.

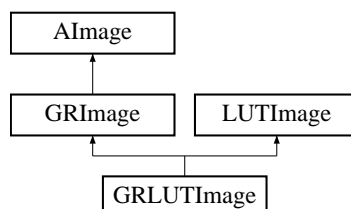
Referenced by GRIImage().

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

- Graphics/include/AImage.h

## 9.56 GRLUTImage Class Reference

#include <AImage.h> Inheritance diagram for GRLUTImage::



### Public Member Functions

- **GRLUTImage** (const **GRLUTImage** &li)
- **GRLUTImage** ()
- **GRLUTImage** (const **GRIImage** &i)
- **GRLUTImage** (int w, int h)
- int **getPixel** (int i, int j) const
- int **getPixel** (int n) const

### Private Member Functions

- void **printOn** (ostream &) const

### Friends

- ostream & **operator<<** (ostream &o, const **GRLUTImage** &i)

### 9.56.1 Detailed Description

Definition at line 198 of file AImage.h.

### 9.56.2 Constructor & Destructor Documentation

#### 9.56.2.1 GRLUTImage::GRLUTImage (const GRLUTImage &li) [inline]

Definition at line 203 of file AImage.h.



**9.56.2.2 GRLUTImage::GRLUTImage () [inline]**

Definition at line 206 of file AImage.h.

**9.56.2.3 GRLUTImage::GRLUTImage (const GRImage & i) [inline]**

Definition at line 209 of file AImage.h.

**9.56.2.4 GRLUTImage::GRLUTImage (int w, int h) [inline]**

Definition at line 212 of file AImage.h.

**9.56.3 Member Function Documentation****9.56.3.1 int GRLUTImage::getPixel (int n) const [virtual]**

Reimplemented from **GRImage** (p. 263).

**9.56.3.2 int GRLUTImage::getPixel (int i, int j) const [virtual]**

Reimplemented from **GRImage** (p. 263).

**9.56.3.3 void GRLUTImage::printOn (ostream &) const [private]****9.56.4 Friends And Related Function Documentation****9.56.4.1 ostream& operator<< (ostream & o, const GRLUTImage & i) [friend]**

Definition at line 220 of file AImage.h.

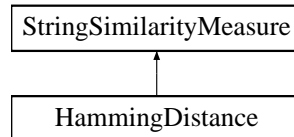
The documentation for this class was generated from the following file:

- Graphics/include/AImage.h

## 9.57 HammingDistance Class Reference

Implements the Hamming distance between two words.

#include <SimilarityMeasures.h> Inheritance diagram for HammingDistance::



### Public Member Functions

- **HammingDistance ()**
- double **measure** (const **Word** &w1, const **Word** &w2) const  
*Returns the hamming distance scaled by the length of the longest word.*

#### 9.57.1 Detailed Description

Implements the Hamming distance between two words.

Definition at line 108 of file SimilarityMeasures.h.

#### 9.57.2 Constructor & Destructor Documentation

##### 9.57.2.1 HammingDistance::HammingDistance () [inline]

Definition at line 111 of file SimilarityMeasures.h.

#### 9.57.3 Member Function Documentation

##### 9.57.3.1 double HammingDistance::measure (const Word &w1, const Word &w2) const [virtual]

Returns the hamming distance scaled by the length of the longest word.

**Parameters:**

*w1* - the first word

$w_2$  - the second word

**Returns:**

the scaled Hamming distance.

Implements **StringSimilarityMeasure** (p. 520).

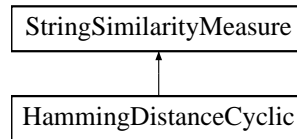
The documentation for this class was generated from the following file:

- StringSimilarity/include/**SimilarityMeasures.h**

## 9.58 HammingDistanceCyclic Class Reference

Implements the Hamming distance between two cyclic words.

#include <SimilarityMeasures.h> Inheritance diagram for HammingDistanceCyclic::



### Public Member Functions

- **HammingDistanceCyclic ()**
- double **measure** (const **Word** &w1, const **Word** &w2) const

*Returns the hamming distance computed for all cyclic permutations of given words and scaled by the length of the longest word.*

### 9.58.1 Detailed Description

Implements the Hamming distance between two cyclic words.

Definition at line 123 of file SimilarityMeasures.h.

### 9.58.2 Constructor & Destructor Documentation

#### 9.58.2.1 HammingDistanceCyclic::HammingDistanceCyclic () [inline]

Definition at line 126 of file SimilarityMeasures.h.

### 9.58.3 Member Function Documentation

#### 9.58.3.1 double HammingDistanceCyclic::measure (const Word &w1, const Word &w2) const [virtual]

Returns the hamming distance computed for all cyclic permutations of given words and scaled by the length of the longest word.

**Parameters:**

- w1* - the first word
- w2* - the second word

**Returns:**

the scaled Hamming distance between cyclic words.

Implements **StringSimilarityMeasure** (p. 520).

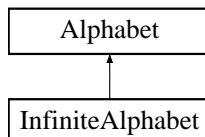
The documentation for this class was generated from the following file:

- StringSimilarity/include/**SimilarityMeasures.h**

## 9.59 InfiniteAlphabet Class Reference

Implements an infinite size alphabet.

#include <Alphabet.h> Inheritance diagram for InfiniteAlphabet::



### Public Member Functions

- **InfiniteAlphabet** (string pref=string("x"))  
*Constructor.*
- int **getNum** (const string &letter) const  
*Implements the conversion from a letter name into its corresponding index.*
- string **getLetter** (int index) const  
*Returns the name of the letter with a given index.*

### Static Public Attributes

- static **InfiniteAlphabet defaultAlphabet**  
*Static instance of the default alphabet  $x_1, x_2, \dots$*

### Private Attributes

- string **thePrefix**

### Friends

- ostream & **operator<<** (ostream &out, const **InfiniteAlphabet** &a)  
*Output the alphabet into a stream.*

### 9.59.1 Detailed Description

Implements an infinite size alphabet.

Definition at line 153 of file Alphabet.h.

### 9.59.2 Constructor & Destructor Documentation

#### 9.59.2.1 InfiniteAlphabet::InfiniteAlphabet (string *pref* = string("x")) [inline]

Constructor.

**Parameters:**

*pref* - prefix to be used for letter names. By default the letter 'x' is used. In general letter names defined by appending the corresponding index to the prefix, i.e. <prefix><index>.

Definition at line 162 of file Alphabet.h.

### 9.59.3 Member Function Documentation

#### 9.59.3.1 string InfiniteAlphabet::getLetter (int *index*) const [virtual]

Returns the name of the letter with a given index.

**Parameters:**

*index* - the index of a letter from the alphabet

**Returns:**

the name of the letter with index *index*

Implements **Alphabet** (p. 104).

#### 9.59.3.2 int InfiniteAlphabet::getNum (const string & *letter*) const [virtual]

Implements the conversion from a letter name into its corresponding index.

**Parameters:**

*letter* - the name of a letter

**Returns:**

letter index in the alphabet. If cannot match the letter, 0 is returned.

Implements **Alphabet** (p. 104).

**9.59.4 Friends And Related Function Documentation****9.59.4.1 ostream& operator<< (ostream & *out*, const InfiniteAlphabet & *a*)  
[friend]**

Output the alphabet into a stream.

Definition at line 183 of file Alphabet.h.

**9.59.5 Member Data Documentation****9.59.5.1 InfiniteAlphabet InfiniteAlphabet::defaultAlphabet [static]**

Static instance of the default alphabet *x\_1*, *x\_2*, ...

Definition at line 180 of file Alphabet.h.

Referenced by Word::printOn().

**9.59.5.2 string InfiniteAlphabet::thePrefix [private]**

Definition at line 193 of file Alphabet.h.

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

- Alphabet/include/**Alphabet.h**



## 9.60 PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::intColHdr Struct Reference

### Public Attributes

- unsigned int **colno**
- unsigned int **col**
- ColHdr **ch**
- int **nextUnused**
- bool **used**

#### 9.60.1 Detailed Description

template<class RowHdr, class ColHdr, typename INTERNAL\_TYPE> struct  
PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::intColHdr

Definition at line 40 of file SignMatrix.h.

#### 9.60.2 Member Data Documentation

9.60.2.1 template<class RowHdr, class ColHdr, typename  
INTERNAL\_TYPE> ColHdr PC::SignMatrix< RowHdr, ColHdr,  
INTERNAL\_TYPE >::intColHdr::ch

Definition at line 40 of file SignMatrix.h.

9.60.2.2 template<class RowHdr, class ColHdr, typename  
INTERNAL\_TYPE> unsigned int PC::SignMatrix< RowHdr,  
ColHdr, INTERNAL\_TYPE >::intColHdr::col

Definition at line 40 of file SignMatrix.h.

9.60.2.3 template<class RowHdr, class ColHdr, typename  
INTERNAL\_TYPE> unsigned int PC::SignMatrix< RowHdr,  
ColHdr, INTERNAL\_TYPE >::intColHdr::colno

Definition at line 40 of file SignMatrix.h.

**9.60.2.4** `template<class RowHdr, class ColHdr, typename  
INTERNAL_TYPE> int PC::SignMatrix< RowHdr, ColHdr,  
INTERNAL_TYPE >::intColHdr::nextUnused`

Definition at line 40 of file SignMatrix.h.

**9.60.2.5** `template<class RowHdr, class ColHdr, typename  
INTERNAL_TYPE> bool PC::SignMatrix< RowHdr, ColHdr,  
INTERNAL_TYPE >::intColHdr::used`

Definition at line 40 of file SignMatrix.h.

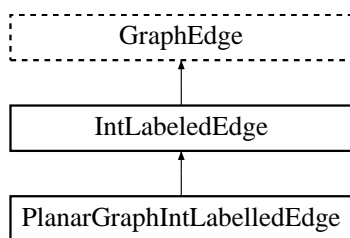
The documentation for this struct was generated from the following file:

- HigmanGroup/include/**SignMatrix.h**

## 9.61 IntLabeledEdge Struct Reference

Defines labelled (by integer) edge of the directed graph.

`#include <GraphType.h>` Inheritance diagram for IntLabeledEdge::



### Public Member Functions

- **IntLabeledEdge** (int **target**, int label)  
*Constructor for an edge. Argument t is a number of the target.*
- **IntLabeledEdge** ()  
*Dummy constructor, not to be used, required for STL::map. Also, can be used to denote "failure" or "dead-end" edge.*
- **IntLabeledEdge** **inverse** (int origin)  
*Invert an edge.*
- bool **operator<** (const **IntLabeledEdge** &e) const  
*Check if one edge is less than the other.*
- bool **operator==** (const **IntLabeledEdge** &e) const  
*Check if two edges equal.*
- bool **operator!=** (const **IntLabeledEdge** &e) const  
*Check if two edges are not equal.*

### Public Attributes

- int **theLabel**  
*The label of the edge.*

### 9.61.1 Detailed Description

Defines labelled (by integer) edge of the directed graph.

Definition at line 103 of file GraphType.h.

### 9.61.2 Constructor & Destructor Documentation

#### 9.61.2.1 `IntLabeledEdge::IntLabeledEdge (int target, int label) [inline]`

Constructor for an edge. Argument *t* is a number of the target.

Definition at line 106 of file GraphType.h.

#### 9.61.2.2 `IntLabeledEdge::IntLabeledEdge () [inline]`

Dummy constructor, not to be used, required for STL::map. Also, can be used to denote "failure" or "dead-end" edge.

Definition at line 109 of file GraphType.h.

Referenced by `inverse()`.

### 9.61.3 Member Function Documentation

#### 9.61.3.1 `IntLabeledEdge IntLabeledEdge::inverse (int origin) [inline]`

Invert an edge.

Reimplemented from **GraphEdge** (p. 253).

Reimplemented in **PlanarGraphIntLabelledEdge** (p. 393).

Definition at line 113 of file GraphType.h.

References `IntLabeledEdge()`, and `theLabel`.

#### 9.61.3.2 `bool IntLabeledEdge::operator!= (const IntLabeledEdge & e) const [inline]`

Check if two edges are not equal.

Reimplemented from **GraphEdge** (p. 253).

Reimplemented in **PlanarGraphIntLabelledEdge** (p. 393).

Definition at line 128 of file GraphType.h.

References `theLabel`.

### 9.61.3.3 `bool IntLabeledEdge::operator< (const IntLabeledEdge & e) const` `[inline]`

Check if one edge is less than the other.

Reimplemented from **GraphEdge** (p. 253).

Reimplemented in **PlanarGraphIntLabelledEdge** (p. 394).

Definition at line 116 of file GraphType.h.

References `theLabel`, and `GraphEdge::theTarget`.

### 9.61.3.4 `bool IntLabeledEdge::operator== (const IntLabeledEdge & e) const` `[inline]`

Check if two edges equal.

Reimplemented from **GraphEdge** (p. 254).

Reimplemented in **PlanarGraphIntLabelledEdge** (p. 394).

Definition at line 124 of file GraphType.h.

References `theLabel`.

## 9.61.4 Member Data Documentation

### 9.61.4.1 `int IntLabeledEdge::theLabel`

The label of the edge.

Definition at line 132 of file GraphType.h.

Referenced by `PlanarGraphIntLabelledEdge::inverse()`, `inverse()`, `PlanarGraphIntLabelledEdge::operator!=()`, `operator!=()`, `PlanarGraphIntLabelledEdge::operator<()`, `operator<()`, `PlanarGraphIntLabelledEdge::operator==()`, and `operator==()`.

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

- `Graph/include/GraphType.h`

## 9.62 PC::PowerCircuitGraph::IntMarking Struct Reference

### Public Attributes

- bool **deleted**
- bool **reduced**
- int **nextDeleted**
- int **refCount**
- std::list< std::pair< Node, Sign > > **nodes**

### 9.62.1 Detailed Description

Definition at line 32 of file PowerCircuitGraph.h.

### 9.62.2 Member Data Documentation

#### 9.62.2.1 bool PC::PowerCircuitGraph::IntMarking::deleted

Definition at line 34 of file PowerCircuitGraph.h.

#### 9.62.2.2 int PC::PowerCircuitGraph::IntMarking::nextDeleted

Definition at line 36 of file PowerCircuitGraph.h.

#### 9.62.2.3 std::list< std::pair<Node,Sign> > PC::PowerCircuitGraph::IntMarking::nodes

Definition at line 38 of file PowerCircuitGraph.h.

#### 9.62.2.4 bool PC::PowerCircuitGraph::IntMarking::reduced

Definition at line 35 of file PowerCircuitGraph.h.

#### 9.62.2.5 int PC::PowerCircuitGraph::IntMarking::refCount

Definition at line 37 of file PowerCircuitGraph.h.

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

- HigmanGroup/include/PowerCircuitGraph.h

## 9.63 PC::PowerCircuitCompMatrix::IntMarking Struct Reference

### Public Attributes

- Col col
- PowerCircuitCompMatrix \* pc
- MarkingType type
- Row node
- bool deleted
- int nextDeleted

#### 9.63.1 Detailed Description

Definition at line 55 of file PowerCircuitCompMatrix.h.

#### 9.63.2 Member Data Documentation

##### 9.63.2.1 Col PC::PowerCircuitCompMatrix::IntMarking::col

Definition at line 57 of file PowerCircuitCompMatrix.h.

##### 9.63.2.2 bool PC::PowerCircuitCompMatrix::IntMarking::deleted

Definition at line 61 of file PowerCircuitCompMatrix.h.

##### 9.63.2.3 int PC::PowerCircuitCompMatrix::IntMarking::nextDeleted

Definition at line 62 of file PowerCircuitCompMatrix.h.

##### 9.63.2.4 Row PC::PowerCircuitCompMatrix::IntMarking::node

Definition at line 60 of file PowerCircuitCompMatrix.h.

##### 9.63.2.5 PowerCircuitCompMatrix\* PC::PowerCircuitCompMatrix::IntMarking::pc

Definition at line 58 of file PowerCircuitCompMatrix.h.

### 9.63.2.6 MarkingType PC::PowerCircuitCompMatrix::IntMarking::type

Definition at line 59 of file PowerCircuitCompMatrix.h.

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

- HigmanGroup/include/**PowerCircuitCompMatrix.h**



## 9.64 PC::PowerCircuitCompMatrix::IntNode Struct Reference

### Public Attributes

- Row `row`
- `PowerCircuitCompMatrix * pc`
- bool `deleted`
- int `nextDeleted`

#### 9.64.1 Detailed Description

Definition at line 47 of file `PowerCircuitCompMatrix.h`.

#### 9.64.2 Member Data Documentation

##### 9.64.2.1 bool PC::PowerCircuitCompMatrix::IntNode::deleted

Definition at line 51 of file `PowerCircuitCompMatrix.h`.

##### 9.64.2.2 int PC::PowerCircuitCompMatrix::IntNode::nextDeleted

Definition at line 52 of file `PowerCircuitCompMatrix.h`.

##### 9.64.2.3 PowerCircuitCompMatrix\* PC::PowerCircuitCompMatrix::IntNode::pc

Definition at line 50 of file `PowerCircuitCompMatrix.h`.

##### 9.64.2.4 Row PC::PowerCircuitCompMatrix::IntNode::row

Definition at line 49 of file `PowerCircuitCompMatrix.h`.

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

- `HigmanGroup/include/PowerCircuitCompMatrix.h`

## 9.65 PC::PowerCircuitGraph::IntNode Struct Reference

### Public Attributes

- bool **BV**
- unsigned int **indexInOrder**
- bool **deleted**
- int **nextDeleted**
- std::list< std::pair< **Node**, **Sign** > > **successors**

### 9.65.1 Detailed Description

Definition at line 23 of file PowerCircuitGraph.h.

### 9.65.2 Member Data Documentation

#### 9.65.2.1 bool PC::PowerCircuitGraph::IntNode::BV

Definition at line 25 of file PowerCircuitGraph.h.

#### 9.65.2.2 bool PC::PowerCircuitGraph::IntNode::deleted

Definition at line 27 of file PowerCircuitGraph.h.

#### 9.65.2.3 unsigned int PC::PowerCircuitGraph::IntNode::indexInOrder

Definition at line 26 of file PowerCircuitGraph.h.

#### 9.65.2.4 int PC::PowerCircuitGraph::IntNode::nextDeleted

Definition at line 28 of file PowerCircuitGraph.h.

#### 9.65.2.5 std::list< std::pair<Node,Sign> > PC::PowerCircuitGraph::IntNode::successors

Definition at line 29 of file PowerCircuitGraph.h.

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

- HigmanGroup/include/**PowerCircuitGraph.h**

## 9.66 PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::intRowHdr Struct Reference

### Public Attributes

- unsigned int **rowno**
- unsigned int **row**
- RowHdr **rh**
- int **nextUnused**
- bool **used**

#### 9.66.1 Detailed Description

template<class RowHdr, class ColHdr, typename INTERNAL\_TYPE> struct  
PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::intRowHdr

Definition at line 39 of file SignMatrix.h.

#### 9.66.2 Member Data Documentation

**9.66.2.1** template<class RowHdr, class ColHdr, typename  
INTERNAL\_TYPE> int PC::SignMatrix< RowHdr, ColHdr,  
INTERNAL\_TYPE >::intRowHdr::nextUnused

Definition at line 39 of file SignMatrix.h.

**9.66.2.2** template<class RowHdr, class ColHdr, typename  
INTERNAL\_TYPE> RowHdr PC::SignMatrix< RowHdr, ColHdr,  
INTERNAL\_TYPE >::intRowHdr::rh

Definition at line 39 of file SignMatrix.h.

**9.66.2.3** template<class RowHdr, class ColHdr, typename  
INTERNAL\_TYPE> unsigned int PC::SignMatrix< RowHdr,  
ColHdr, INTERNAL\_TYPE >::intRowHdr::row

Definition at line 39 of file SignMatrix.h.

**9.66.2.4** `template<class RowHdr, class ColHdr, typename  
INTERNAL_TYPE> unsigned int PC::SignMatrix< RowHdr,  
ColHdr, INTERNAL_TYPE >::intRowHdr::rowno`

Definition at line 39 of file SignMatrix.h.

**9.66.2.5** `template<class RowHdr, class ColHdr, typename  
INTERNAL_TYPE> bool PC::SignMatrix< RowHdr, ColHdr,  
INTERNAL_TYPE >::intRowHdr::used`

Definition at line 39 of file SignMatrix.h.

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

- HigmanGroup/include/**SignMatrix.h**

## 9.67 KLProtocolInstance Class Reference

```
#include <KLKeyGeneration.h>
```

### Public Member Functions

- **KLProtocolInstance** (int braid\_rank, **Word** base\_word, **Word** keyA, **Word** KeyB)  
*Create an instance of the protocol.*
- int **getBraidRank** () const  
*(accessor function) get the rank of the braid group*
- **Word** **getPrivateKeyA** () const  
*(accessor function) get the private key of the first party*
- **Word** **getPrivateKeyB** () const  
*(accessor function) get the private key of the second party*
- **Word** **getPublicKeyA** () const  
*(accessor function) get the public key of the first party*
- **Word** **getPublicKeyB** () const  
*(accessor function) get the public key of the second party*

### Static Public Member Functions

- static **KLProtocolInstance** **random** (int braid\_rank, int baseLenth, int keyLength)  
*Generate a random instance of the protocol.*

### Private Attributes

- int **theRank**  
*the rank of the braid group*
- **Word** **theBase**  
*the base element (denoted by  $w$ )*

- **Word thePrivateKeyA**

*the private key of the first party (denoted by  $A$ )*

- **Word thePublicKeyA**

*the public key of the first party  $P_B = D(A^{-1}wA)$*

- **Word thePrivateKeyB**

*the private key of the second party (denoted by  $B$ )*

- **Word thePublicKeyB**

*the public key of the second party  $P_B = D(B^{-1}wB)$*

### 9.67.1 Detailed Description

Definition at line 23 of file KLKeyGeneration.h.

### 9.67.2 Constructor & Destructor Documentation

#### 9.67.2.1 KLProtocolInstance::KLProtocolInstance (int *braid\_rank*, Word *base\_word*, Word *keyA*, Word *keyB*)

Create an instance of the protocol.

### 9.67.3 Member Function Documentation

#### 9.67.3.1 int KLProtocolInstance::getBraidRank () const [inline]

(accessor function) get the rank of the braid group

Definition at line 55 of file KLKeyGeneration.h.

References theRank.

#### 9.67.3.2 Word KLProtocolInstance::getPrivateKeyA () const [inline]

(accessor function) get the private key of the first party

Definition at line 57 of file KLKeyGeneration.h.

References thePrivateKeyA.

**9.67.3.3 Word KLProtocolInstance::getPrivateKeyB () const [inline]**

(accessor function) get the private key of the second party

Definition at line 59 of file KLKeyGeneration.h.

References thePrivateKeyB.

**9.67.3.4 Word KLProtocolInstance::getPublicKeyA () const [inline]**

(accessor function) get the public key of the first party

Definition at line 61 of file KLKeyGeneration.h.

References thePublicKeyA.

**9.67.3.5 Word KLProtocolInstance::getPublicKeyB () const [inline]**

(accessor function) get the public key of the second party

Definition at line 63 of file KLKeyGeneration.h.

References thePublicKeyB.

**9.67.3.6 static KLProtocolInstance KLProtocolInstance::random (int *braid\_rank*, int *baseLenth*, int *keyLength*) [static]**

Generate a random instance of the protocol.

**Parameters:**

*braid\_rank* - rank of the braid group;

*baseLenth* - the length of the base word

*keyLength* - the length of the key

**9.67.4 Member Data Documentation****9.67.4.1 Word KLProtocolInstance::theBase [private]**

the base element (denoted by  $w$ )

Definition at line 77 of file KLKeyGeneration.h.

**9.67.4.2 Word KLProtocolInstance::thePrivateKeyA [private]**

the private key of the first party (denoted by  $A$ )

Definition at line 80 of file KLKeyGeneration.h.

Referenced by getPrivateKeyA().

#### 9.67.4.3 Word KLProtocolInstance::thePrivateKeyB [private]

the private key of the second party (denoted by  $B$ )

Definition at line 86 of file KLKeyGeneration.h.

Referenced by getPrivateKeyB().

#### 9.67.4.4 Word KLProtocolInstance::thePublicKeyA [private]

the public key of the first party  $P_B = D(A^{-1}wA)$

Definition at line 83 of file KLKeyGeneration.h.

Referenced by getPublicKeyA().

#### 9.67.4.5 Word KLProtocolInstance::thePublicKeyB [private]

the public key of the second party  $P_B = D(B^{-1}wB)$

Definition at line 89 of file KLKeyGeneration.h.

Referenced by getPublicKeyB().

#### 9.67.4.6 int KLProtocolInstance::theRank [private]

the rank of the braid group

Definition at line 74 of file KLKeyGeneration.h.

Referenced by getBraidRank().

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

- CryptoKL/include/KLKeyGeneration.h



## 9.68 ICommDivider Struct Reference

```
#include <DCBraidReduction.h>
```

### Public Member Functions

- **bool operator()** (const **CommDivider** &c1, const **CommDivider** &c2)

#### 9.68.1 Detailed Description

Definition at line 70 of file DCBraidReduction.h.

#### 9.68.2 Member Function Documentation

**9.68.2.1** **bool ICommDivider::operator()** (const **CommDivider** &*c1*, const **CommDivider** &*c2*) [**inline**]

Definition at line 72 of file DCBraidReduction.h.

References **CommDivider::length**.

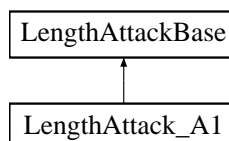
The documentation for this struct was generated from the following file:

- Experiments/include/**DCBraidReduction.h**

## 9.69 LengthAttack\_A1 Class Reference

Implements Length-based attack on AAG protocol.

#include <LengthAttack.h> Inheritance diagram for LengthAttack\_A1::



### Public Member Functions

- **LengthAttack\_A1** ()
- **int type** ()  
*Returns the algorithm type.*
- **findKey\_LengthBasedResult findKey\_LengthBased** (int N, const vector< **Word** > &A1, const vector< **Word** > &A2, const vector< **Word** > &B, int sec=9999999, ostream &out=cout)  
*Attack on an instances of the AAG protocol.*

### Private Member Functions

- **int sbgpGeneratorsWeight** (const vector< **Word** > &A)
- **void addNewElt** (const vector< **Word** > &A, set< **ELT** > &checkedElements, set< **ELT** > &uncheckedElements)
- **void tryElt** (int N, const **ELT** &cur, const vector< **Word** > &B, set< **ELT** > &checkedElements, set< **ELT** > &uncheckedElements)
- **bool check\_ifVectorsEqual** (int N, const vector< **Word** > &A1, const vector< **Word** > &A2)

### 9.69.1 Detailed Description

Implements Length-based attack on AAG protocol. This is a very basic version of the length-based attack which implements the best descend LBA where on each step we choose conjugator which gives the maximal decrease among all currently available tuples. See A.Myasnikov. A.Ushakov, "On the length-based attack" for more details.

Definition at line 66 of file LengthAttack.h.

## 9.69.2 Constructor & Destructor Documentation

### 9.69.2.1 LengthAttack\_A1::LengthAttack\_A1 () [inline]

Definition at line 69 of file LengthAttack.h.

## 9.69.3 Member Function Documentation

**9.69.3.1** void LengthAttack\_A1::addNewElt (const vector< Word > & A, set< ELT > & checkedElements, set< ELT > & uncheckedElements) [private]

**9.69.3.2** bool LengthAttack\_A1::check\_ifVectorsEqual (int N, const vector< Word > & A1, const vector< Word > & A2) [private]

**9.69.3.3** findKey\_LengthBasedResult LengthAttack\_A1::findKey\_LengthBased (int N, const vector< Word > & A1, const vector< Word > & A2, const vector< Word > & B, int sec = 9999999, ostream & out = cout) [virtual]

Attack on an instances of the AAG protocol. Executes a length-based attack on an instance of the AAG protocol

#### Parameters:

*N* - rank of the braid group (number of strands)

*A1* - Alices subgroup generators

*A2* - Alices subgroup generators conjugated by Bob's private key

*B* - Bob's subgroup (Bob's private key belongs to it)

*sec* - amount of time (in seconds) given to procedure to finish

#### Returns:

- Returns **findKey\_LengthBasedResult::SUCCESSFULL** (p. 702) if the attack succeeds, **findKey\_LengthBasedResult::FAILED** (p. 702) if the attack fails and **findKey\_LengthBasedResult::TIME\_EXPIRED** (p. 702) if the time limit is exceeded

Implements **LengthAttackBase** (p. 301).

**9.69.3.4** `int LengthAttack_A1::sbgpGeneratorsWeight (const vector< Word > & A) [private]`

**9.69.3.5** `void LengthAttack_A1::tryElt (int N, const ELT & cur, const vector< Word > & B, set< ELT > & checkedElements, set< ELT > & uncheckedElements) [private]`

**9.69.3.6** `int LengthAttack_A1::type () [inline, virtual]`

Returns the algorithm type.

Implements **LengthAttackBase** (p. 302).

Definition at line 70 of file LengthAttack.h.

References AL1.

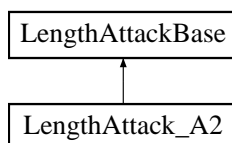
The documentation for this class was generated from the following file:

- CryptoAAG/include/**LengthAttack.h**

## 9.70 LengthAttack\_A2 Class Reference

Implements Length-based attack on AAG protocol.

#include <LengthAttack.h> Inheritance diagram for LengthAttack\_A2::



### Public Member Functions

- **LengthAttack\_A2** ()
- **int type** ()  
*Returns the algorithm type.*
- **findKey\_LengthBasedResult findKey\_LengthBased** (int N, const vector< **Word** > &A1, const vector< **Word** > &A2, const vector< **Word** > &B, int sec=9999999, ostream &out=cout)  
*Attack on an instances of the AAG protocol.*

### Private Member Functions

- **int sbgpGeneratorsWeight** (const vector< **Word** > &A)
- **void addNewElt** (const vector< **Word** > &A, set< **ELT** > &checkedElements, set< **ELT** > &uncheckedElements)
- **void tryElt** (int N, const **ELT** &cur, const vector< **Word** > &B, set< **ELT** > &checkedElements, set< **ELT** > &uncheckedElements)
- **void tryElt** (int N, const **ELT** &cur, const vector< **Word** > &B, set< **ELT** > &checkedElements, set< **ELT** > &uncheckedElements, ostream &out)
- **bool check\_ifVectorsEqual** (int N, const vector< **Word** > &A1, const vector< **Word** > &A2)

#### 9.70.1 Detailed Description

Implements Length-based attack on AAG protocol. This is an implementation of the generalised length-based attack. This is an LBA with backtracking in which the set of elements in Alice's public set is extended by all conjugators. See A.Myasnikov. A.Ushakov, "On the length-based attack" for more details.

Definition at line 103 of file LengthAttack.h.

## 9.70.2 Constructor & Destructor Documentation

### 9.70.2.1 LengthAttack\_A2::LengthAttack\_A2 () [inline]

Definition at line 106 of file LengthAttack.h.

## 9.70.3 Member Function Documentation

**9.70.3.1** void LengthAttack\_A2::addNewElt (const vector< Word > & A, set< ELT > & checkedElements, set< ELT > & uncheckedElements) [private]

**9.70.3.2** bool LengthAttack\_A2::check\_ifVectorsEqual (int N, const vector< Word > & A1, const vector< Word > & A2) [private]

**9.70.3.3** findKey\_LengthBasedResult LengthAttack\_A2::findKey\_LengthBased (int N, const vector< Word > & A1, const vector< Word > & A2, const vector< Word > & B, int sec = 9999999, ostream & out = cout) [virtual]

Attack on an instances of the AAG protocol. Executes a length-based attack on an instance of the AAG protocol

#### Parameters:

- $N$  - rank of the braid group (number of strands)
- $A1$  - Alices subgroup generators
- $A2$  - Alices subgroup generators conjugated by Bob's private key
- $B$  - Bob's subgroup (Bob's private key belongs to it)
- $sec$  - amount of time (in seconds) given to procedure to finish

#### Returns:

- Returns **findKey\_LengthBasedResult::SUCCESSFULL** (p. 702) if the attack succeeds, **findKey\_LengthBasedResult::FAILED** (p. 702) if the attack fails and **findKey\_LengthBasedResult::TIME\_EXPIRED** (p. 702) if the time limit is exceeded

Implements **LengthAttackBase** (p. 301).

- 9.70.3.4** `int LengthAttack_A2::sbgpGeneratorsWeight (const vector< Word > & A) [private]`
- 9.70.3.5** `void LengthAttack_A2::tryElt (int N, const ELT & cur, const vector< Word > & B, set< ELT > & checkedElements, set< ELT > & uncheckedElements, ostream & out) [private]`
- 9.70.3.6** `void LengthAttack_A2::tryElt (int N, const ELT & cur, const vector< Word > & B, set< ELT > & checkedElements, set< ELT > & uncheckedElements) [private]`
- 9.70.3.7** `int LengthAttack_A2::type () [inline, virtual]`

Returns the algorithm type.

Implements **LengthAttackBase** (p. 302).

Definition at line 107 of file LengthAttack.h.

References AL2.

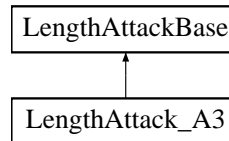
The documentation for this class was generated from the following file:

- CryptoAAG/include/**LengthAttack.h**

## 9.71 LengthAttack\_A3 Class Reference

Implements Length-based attack on AAG protocol.

#include <LengthAttack.h> Inheritance diagram for LengthAttack\_A3::



### Public Member Functions

- **LengthAttack\_A3** ()
- **int type** ()  
*Returns the algorithm type.*
- **findKey\_LengthBasedResult findKey\_LengthBased** (int N, const vector< **Word** > &A1, const vector< **Word** > &A2, const vector< **Word** > &B, int sec=9999999, ostream &out=cout)

*Attack on an instances of the AAG protocol.*

### Private Member Functions

- void **addProducts** (const vector< **Word** > &elem\_set, vector< **Word** > &ext\_set, vector< **Word** > &ext\_set\_sg\_gens, const **Word** &sel\_gen, int sel\_gen\_sg)
- void **addAllProducts** (const vector< **Word** > &elem\_set, vector< **Word** > &ext\_set, vector< **Word** > &ext\_set\_sg\_gens)
- int **sbgpGeneratorsWeight** (const vector< **Word** > &A)
- void **addNewElt** (const vector< **Word** > &A, set< **ELT** > &checkedElements, set< **ELT** > &uncheckedElements)
- void **tryElt** (int N, const **ELT** &cur, const vector< **Word** > &B, set< **ELT** > &checkedElements, set< **ELT** > &uncheckedElements)
- void **tryElt** (int N, const **ELT** &cur, const vector< **Word** > &B, const vector< **Word** > &B\_sg\_gens, set< **ELT** > &checkedElements, set< **ELT** > &uncheckedElements, bool is\_B\_extended, ostream &out)
- bool **check\_ifVectorsEqual** (int N, const vector< **Word** > &A1, const vector< **Word** > &A2)



### 9.71.1 Detailed Description

Implements Length-based attack on AAG protocol. This is an implementation of the generalised length-based attack. This is an LBA with backtracking in which the set of elements in Alice's public set on each iteration is extended by conjugators and two-products of the "best" generator. See A.Myasnikov. A.Ushakov, "On the length-based attack" for more details.

Definition at line 143 of file LengthAttack.h.

### 9.71.2 Constructor & Destructor Documentation

#### 9.71.2.1 LengthAttack\_A3::LengthAttack\_A3 () [inline]

Definition at line 146 of file LengthAttack.h.

### 9.71.3 Member Function Documentation

**9.71.3.1** void LengthAttack\_A3::addAllProducts (const vector< Word > & *elem\_set*, vector< Word > & *ext\_set*, vector< Word > & *ext\_set\_sg\_gens*) [private]

**9.71.3.2** void LengthAttack\_A3::addNewElt (const vector< Word > & *A*, set< ELT > & *checkedElements*, set< ELT > & *uncheckedElements*) [private]

**9.71.3.3** void LengthAttack\_A3::addProducts (const vector< Word > & *elem\_set*, vector< Word > & *ext\_set*, vector< Word > & *ext\_set\_sg\_gens*, const Word & *sel\_gen*, int *sel\_gen\_sg*) [private]

**9.71.3.4** bool LengthAttack\_A3::check\_ifVectorsEqual (int *N*, const vector< Word > & *A1*, const vector< Word > & *A2*) [private]

**9.71.3.5** findKey\_LengthBasedResult LengthAttack\_A3::findKey\_LengthBased (int *N*, const vector< Word > & *A1*, const vector< Word > & *A2*, const vector< Word > & *B*, int *sec* = 9999999, ostream & *out* = cout) [virtual]

Attack on an instances of the AAG protocol. Executes a length-based attack on an instance of the AAG protocol

#### Parameters:

*N* - rank of the braid group (number of strands)

*A1* - Alices subgroup generators

*A2* - Alices subgroup generators conjugated by Bob's private key

*B* - Bob's subgroup (Bob's private key belongs to it)

*sec* - amount of time (in seconds) given to procedure to finish

**Returns:**

- Returns **findKey\_LengthBasedResult::SUCCESSFULL** (p. 702) if the attack succeeds, **findKey\_LengthBasedResult::FAILED** (p. 702) if the attack fails and **findKey\_LengthBasedResult::TIME\_EXPIRED** (p. 702) if the time limit is exceeded

Implements **LengthAttackBase** (p. 301).

**9.71.3.6** `int LengthAttack_A3::sbgpGeneratorsWeight (const vector< Word > &A) [private]`

**9.71.3.7** `void LengthAttack_A3::tryElt (int N, const ELT &cur, const vector< Word > &B, const vector< Word > &B_sg_gens, set< ELT > &checkedElements, set< ELT > &uncheckedElements, bool is_B_extended, ostream &out) [private]`

**9.71.3.8** `void LengthAttack_A3::tryElt (int N, const ELT &cur, const vector< Word > &B, set< ELT > &checkedElements, set< ELT > &uncheckedElements) [private]`

**9.71.3.9** `int LengthAttack_A3::type () [inline, virtual]`

Returns the algorithm type.

Implements **LengthAttackBase** (p. 302).

Definition at line 147 of file LengthAttack.h.

References AL3.

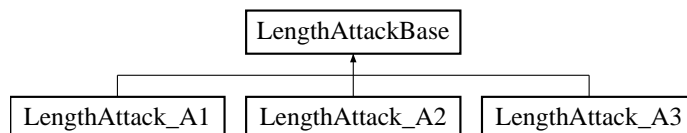
The documentation for this class was generated from the following file:

- CryptoAAG/include/LengthAttack.h

## 9.72 LengthAttackBase Class Reference

Basis interface for classes implementing the Length-Based attack.

#include <LengthAttack.h> Inheritance diagram for LengthAttackBase::



### Public Member Functions

- virtual int **type** ()=0  
*Returns the algorithm type.*
- virtual **findKey\_LengthBasedResult** **findKey\_LengthBased** (int *N*, const vector< **Word** > &*A1*, const vector< **Word** > &*A2*, const vector< **Word** > &*B*, int *sec*=9999999, ostream &*out*=cout)=0  
*Attack on an instances of the AAG protocol.*

### 9.72.1 Detailed Description

Basis interface for classes implementing the Length-Based attack.

Definition at line 32 of file LengthAttack.h.

### 9.72.2 Member Function Documentation

**9.72.2.1** virtual **findKey\_LengthBasedResult** **LengthAttackBase::findKey\_LengthBased** (int *N*, const vector< **Word** > &*A1*, const vector< **Word** > &*A2*, const vector< **Word** > &*B*, int *sec* = 9999999, ostream &*out* = cout) [**pure virtual**]

Attack on an instances of the AAG protocol. Executes a length-based attack on an instance of the AAG protocol

#### Parameters:

- N* - rank of the braid group (number of strands)
- A1* - Alices subgroup generators

**A2** - Alices subgroup generators conjugated by Bob's private key

**B** - Bob's subgroup (Bob's private key belongs to it)

**sec** - amount of time (in seconds) given to procedure to finish

**Returns:**

- Returns **findKey\_LengthBasedResult::SUCCESSFULL** (p. 702) if the attack succeeds, **findKey\_LengthBasedResult::FAILED** (p. 702) if the attack fails and **findKey\_LengthBasedResult::TIME\_EXPIRED** (p. 702) if the time limit is exceeded

Implemented in **LengthAttack\_A1** (p. 293), **LengthAttack\_A2** (p. 296), and **LengthAttack\_A3** (p. 299).

#### 9.72.2.2 **virtual int LengthAttackBase::type () [pure virtual]**

Returns the algorithm type.

Implemented in **LengthAttack\_A1** (p. 294), **LengthAttack\_A2** (p. 297), and **LengthAttack\_A3** (p. 300).

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

- CryptoAAG/include/LengthAttack.h

## 9.73 Levenstein Class Reference

A class for computing the **Levenstein** (p. 303) distance between two words.

```
#include <Levenstein.h>
```

### Public Member Functions

- **Levenstein** ()
- int **compute** (const **Word** &w1, const **Word** &w2)  
*Computes the **Levenstein** (p. 303) distance between two words.*

### Private Member Functions

- char \* **wordToString** (const **Word** &)

#### 9.73.1 Detailed Description

A class for computing the **Levenstein** (p. 303) distance between two words. Converts words to strings (char\*) and then calls a known string algorithm.

Definition at line 30 of file Levenstein.h.

#### 9.73.2 Constructor & Destructor Documentation

##### 9.73.2.1 **Levenstein::Levenstein** () **[inline]**

Definition at line 40 of file Levenstein.h.

#### 9.73.3 Member Function Documentation

##### 9.73.3.1 int **Levenstein::compute** (const **Word** & w1, const **Word** & w2)

Computes the **Levenstein** (p. 303) distance between two words.

##### Parameters:

- w1** - first word.
- w2** - second word

**Returns:**

non-scaled distance, i.e. the actual minimal number of operations required to rewrite one word into another.

**9.73.3.2 char\* Levenstein::wordToString (const Word &) [private]**

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

- StringSimilarity/include/**Levenstein.h**

## 9.74 LinkedBraidStructure Class Reference

```
#include <LinkedBraidStructure.h>
```

### Public Member Functions

- **LinkedBraidStructure** (int N)
- **LinkedBraidStructure** (int N, const **Word** &w)
- **LinkedBraidStructure** (const **LinkedBraidStructure** &LBS)
- **LinkedBraidStructure** & **operator=** (const **LinkedBraidStructure** &LBS)
- bool **operator<** (const **LinkedBraidStructure** &lbs) const
- int **size** () const
- void **clear** ()
- **LinkedBraidStructureTransform** **push\_back** (int g)
- **LinkedBraidStructureTransform** **push\_front** (int g)
- void **removeLeftHandles** (list< **LinkedBraidStructureTransform** > \*result=NULL)
- void **removeRightHandles** (list< **LinkedBraidStructureTransform** > \*result=NULL)
- **Word** **translateIntoWord** () const
- void **undo** (const list< **LinkedBraidStructureTransform** > &lst\_seq)
- void **undo** (const **LinkedBraidStructureTransform** &lst)
- **LinkedBraidStructure** (int N)
- ~**LinkedBraidStructure** ()
- void **push\_back** (int i)
- template<class ConstIntIterator >  
void **push\_back** (ConstIntIterator B, ConstIntIterator E)
- void **clear** ()
- void **push\_front** (int i)
- void **transformToDehornoyForm** ()
- list< int > **getWord** () const

### Private Member Functions

- **LinkedBraidStructureTransform** **make\_EraseTransform** (**BraidNode** \*bn, int pos) const
- **LinkedBraidStructureTransform** **make\_AddTransform** (**BraidNode** \*bn, int pos) const
- **LinkedBraidStructureTransform** **make\_ChangeType** (**BraidNode** \*bn, int pos) const
- int **checkIfStartsLeftHandle** (int pos, **BraidNode** \*bn)
- int **checkIfStartsRightHandle** (int pos, **BraidNode** \*bn)

- void **removeLeftHandle** (**triple**< int, int, **BraidNode** \* > node, set< **triple**< int, int, **BraidNode** \* > > &to\_check, list< **LinkedBraidStructureTransform** > \*lst)
- void **removeRightHandle** (**triple**< int, int, **BraidNode** \* > node, set< **triple**< int, int, **BraidNode** \* > > &to\_check, list< **LinkedBraidStructureTransform** > \*lst)
- **LinkedBraidStructureTransform** **removeNode** (**BraidNode** \*bn, int pos)
- **BraidNode** \* **insertBackRight** (**BraidNode** \*bn, int pos, bool type)
- **BraidNode** \* **insertBackLeft** (**BraidNode** \*bn, int pos, bool type)
- **BraidNode** \* **insert** (const **LinkedBraidStructureTransform** &lbst)
- void **processTree** (int al, **BraidNode** \*node, **Word** &result) const
- void **clearLinks** () const
- void **processNode** (list< int > &result, int index, const **BraidNode** \*bn) const

*Function used in **getWord**( ) (p. 307).*

## Private Attributes

- int **theIndex**

*Number of generators!!!*

- vector< **BraidNode** \* > **frontNodes**
- vector< **BraidNode** \* > **backNodes**
- map< int, **BraidNode** > **theNodes**
- int **maxNodeNumber**
- int **theRank**
- bool **theMark**

### 9.74.1 Detailed Description

Definition at line 104 of file **LinkedBraidStructure.h**.



## 9.74.2 Constructor & Destructor Documentation

9.74.2.1 `LinkedBraidStructure::LinkedBraidStructure (int N)`

9.74.2.2 `LinkedBraidStructure::LinkedBraidStructure (int N, const Word & w)`

9.74.2.3 `LinkedBraidStructure::LinkedBraidStructure (const LinkedBraidStructure & LBS)`

9.74.2.4 `LinkedBraidStructure::LinkedBraidStructure (int N)`

9.74.2.5 `LinkedBraidStructure::~~LinkedBraidStructure ()`

## 9.74.3 Member Function Documentation

9.74.3.1 `int LinkedBraidStructure::checkIfStartsLeftHandle (int pos, BraidNode * bn) [private]`

9.74.3.2 `int LinkedBraidStructure::checkIfStartsRightHandle (int pos, BraidNode * bn) [private]`

9.74.3.3 `void LinkedBraidStructure::clear ()`

9.74.3.4 `void LinkedBraidStructure::clear ()`

9.74.3.5 `void LinkedBraidStructure::clearLinks () const [private]`

9.74.3.6 `list< int > LinkedBraidStructure::getWord () const`

9.74.3.7 `BraidNode* LinkedBraidStructure::insert (const LinkedBraidStructureTransform & lbt) [private]`

9.74.3.8 `BraidNode* LinkedBraidStructure::insertBackLeft (BraidNode * bn, int pos, bool type) [private]`

9.74.3.9 `BraidNode* LinkedBraidStructure::insertBackRight (BraidNode * bn, int pos, bool type) [private]`

9.74.3.10 `LinkedBraidStructureTransform LinkedBraidStructure::make_-AddTransform (BraidNode * bn, int pos) const [private]`

9.74.3.11 `LinkedBraidStructureTransform LinkedBraidStructure::make_-ChangeType (BraidNode * bn, int pos) const [private]`

Generated on Mon Sep 26 18:43:45 2011 for CRyptography And Groups (CRAG) by Doxygen

9.74.3.12 `LinkedBraidStructureTransform LinkedBraidStructure::make_-EraseTransform (BraidNode * bn, int pos) const [private]`

9.74.3.13 `bool LinkedBraidStructure::operator< (const LinkedBraidStructure & lbs) const`

9.74.3.14 `LinkedBraidStructure & LinkedBraidStructure::operator= (const`

**9.74.3.16** `void LinkedBraidStructure::processTree (int al, BraidNode * node, Word & result) const` [**private**]

**9.74.3.17** `template<class ConstIntIterator > void LinkedBraidStructure::push_back (ConstIntIterator B, ConstIntIterator E)` [**inline**]

Definition at line 140 of file LinkedBraidStructure\_old.h.

References `push_back()`.

**9.74.3.18** `void LinkedBraidStructure::push_back (int i)`

**9.74.3.19** `LinkedBraidStructureTransform LinkedBraidStructure::push_back (int g)`

Referenced by `push_back()`.

**9.74.3.20** void LinkedBraidStructure::push\_front (int *i*)

**9.74.3.21** LinkedBraidStructureTransform LinkedBraidStructure::push\_front (int *g*)

**9.74.3.22** void LinkedBraidStructure::removeLeftHandle (triple< int, int, BraidNode \* > *node*, set< triple< int, int, BraidNode \* > > & *to\_check*, list< LinkedBraidStructureTransform > \* *lst*)  
[private]

**9.74.3.23** void LinkedBraidStructure::removeLeftHandles (list< LinkedBraidStructureTransform > \* *result* = NULL)

**9.74.3.24** LinkedBraidStructureTransform LinkedBraidStructure::removeNode (BraidNode \* *bn*, int *pos*)  
[private]

**9.74.3.25** void LinkedBraidStructure::removeRightHandle (triple< int, int, BraidNode \* > *node*, set< triple< int, int, BraidNode \* > > & *to\_check*, list< LinkedBraidStructureTransform > \* *lst*)  
[private]

**9.74.3.26** void LinkedBraidStructure::removeRightHandles (list< LinkedBraidStructureTransform > \* *result* = NULL)

**9.74.3.27** int LinkedBraidStructure::size () const [inline]

Definition at line 134 of file LinkedBraidStructure.h.

References theNodes.

**9.74.3.28** void `LinkedBraidStructure::transformToDehornoyForm ()`

**9.74.3.29** Word `LinkedBraidStructure::translateIntoWord () const`

**9.74.3.30** void `LinkedBraidStructure::undo (const  
LinkedBraidStructureTransform & lbst)`

**9.74.3.31** void `LinkedBraidStructure::undo (const list<  
LinkedBraidStructureTransform > & lbst_seq)`

## **9.74.4 Member Data Documentation**

**9.74.4.1** vector< `BraidNode *` > `LinkedBraidStructure::backNodes`  
[private]

Definition at line 185 of file `LinkedBraidStructure.h`.

**9.74.4.2** vector< `BraidNode *` > `LinkedBraidStructure::frontNodes`  
[private]

Definition at line 184 of file `LinkedBraidStructure.h`.

**9.74.4.3** int `LinkedBraidStructure::maxNodeNumber` [private]

Definition at line 188 of file `LinkedBraidStructure.h`.

**9.74.4.4** int `LinkedBraidStructure::theIndex` [private]

Number of generators!!!

Definition at line 183 of file `LinkedBraidStructure.h`.

**9.74.4.5** bool `LinkedBraidStructure::theMark` [mutable, private]

Definition at line 175 of file `LinkedBraidStructure_old.h`.

**9.74.4.6** map< int , `BraidNode` > `LinkedBraidStructure::theNodes`  
[private]

Definition at line 186 of file `LinkedBraidStructure.h`.

Referenced by `size()`.

**9.74.4.7 int LinkedBraidStructure::theRank [private]**

Definition at line 168 of file LinkedBraidStructure\_old.h.

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

- BraidGroup/include/LinkedBraidStructure.h
- BraidGroup/include/LinkedBraidStructure\_old.h

## 9.75 LinkedBraidStructureTransform Struct Reference

```
#include <LinkedBraidStructure.h>
```

### Public Types

- enum **TRANSFORM** { **ERASED**, **ADDED**, **CHANGE\_TYPE** }

### Public Member Functions

- **LinkedBraidStructureTransform** (int n, int p, **TRANSFORM** tr, bool t=false, int l=0, int a=0, int r=0, int bl=0, int b=0, int br=0)

### Public Attributes

- **TRANSFORM** **theTransform**
- int **left**
- int **ahead**
- int **right**
- int **back\_left**
- int **back**
- int **back\_right**
- bool **type**
- int **theNumber**
- int **thePosition**

### 9.75.1 Detailed Description

Definition at line 75 of file LinkedBraidStructure.h.

### 9.75.2 Member Enumeration Documentation

#### 9.75.2.1 enum LinkedBraidStructureTransform::TRANSFORM

Enumerator:

*ERASED*  
*ADDED*  
*CHANGE\_TYPE*

Definition at line 77 of file LinkedBraidStructure.h.

### 9.75.3 Constructor & Destructor Documentation

**9.75.3.1** `LinkedBraidStructureTransform::LinkedBraidStructureTransform`  
(int *n*, int *p*, TRANSFORM *tr*, bool *t* = `false`, int *l* = 0, int *a* = 0,  
int *r* = 0, int *bl* = 0, int *b* = 0, int *br* = 0) [`inline`]

Definition at line 79 of file `LinkedBraidStructure.h`.

### 9.75.4 Member Data Documentation

**9.75.4.1** `int LinkedBraidStructureTransform::ahead`

Definition at line 86 of file `LinkedBraidStructure.h`.

**9.75.4.2** `int LinkedBraidStructureTransform::back`

Definition at line 90 of file `LinkedBraidStructure.h`.

**9.75.4.3** `int LinkedBraidStructureTransform::back_left`

Definition at line 89 of file `LinkedBraidStructure.h`.

**9.75.4.4** `int LinkedBraidStructureTransform::back_right`

Definition at line 91 of file `LinkedBraidStructure.h`.

**9.75.4.5** `int LinkedBraidStructureTransform::left`

Definition at line 85 of file `LinkedBraidStructure.h`.

**9.75.4.6** `int LinkedBraidStructureTransform::right`

Definition at line 87 of file `LinkedBraidStructure.h`.

**9.75.4.7** `int LinkedBraidStructureTransform::theNumber`

Definition at line 94 of file `LinkedBraidStructure.h`.

**9.75.4.8 int LinkedBraidStructureTransform::thePosition**

Definition at line 95 of file LinkedBraidStructure.h.

**9.75.4.9 TRANSFORM LinkedBraidStructureTransform::theTransform**

Definition at line 83 of file LinkedBraidStructure.h.

**9.75.4.10 bool LinkedBraidStructureTransform::type**

Definition at line 93 of file LinkedBraidStructure.h.

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

- BraidGroup/include/**LinkedBraidStructure.h**



## 9.76 LookUpTable Class Reference

```
#include <AImage.h>
```

### Public Member Functions

- **LookUpTable** ()
- **LookUpTable** (const **LookUpTable** &)
- **LookUpTable** & **operator=** (const **LookUpTable** &)
- void **set** (int, int)
- int **get** (int) const

### Private Attributes

- unsigned char **table** [256]

#### 9.76.1 Detailed Description

Definition at line 31 of file AImage.h.

#### 9.76.2 Constructor & Destructor Documentation

##### 9.76.2.1 LookUpTable::LookUpTable ()

##### 9.76.2.2 LookUpTable::LookUpTable (const LookUpTable &)

#### 9.76.3 Member Function Documentation

##### 9.76.3.1 int LookUpTable::get (int) const

Referenced by CLUTImage::getBluePixel(), CLUTImage::getGreenPixel(), LUTImage::getInLT(), and CLUTImage::getRedPixel().

##### 9.76.3.2 LookUpTable& LookUpTable::operator= (const LookUpTable &)

##### 9.76.3.3 void LookUpTable::set (int, int)

Referenced by LUTImage::setInLT().

## 9.76.4 Member Data Documentation

### 9.76.4.1 unsigned char LookUpTable::table[256] [private]

Definition at line 46 of file AImage.h.

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

- Graphics/include/AImage.h

## 9.77 ltstr Struct Reference

Implements a comparison operator on two strings.

```
#include <ConfigFile.h>
```

### Public Member Functions

- `bool operator() (const string &s1, const string &s2) const`

#### 9.77.1 Detailed Description

Implements a comparison operator on two strings.

Definition at line 24 of file ConfigFile.h.

#### 9.77.2 Member Function Documentation

##### 9.77.2.1 `bool ltstr::operator() (const string & s1, const string & s2) const` `[inline]`

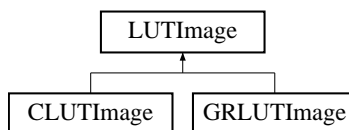
Definition at line 26 of file ConfigFile.h.

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

- `general/include/ConfigFile.h`

## 9.78 LUTImage Class Reference

#include <AImage.h> Inheritance diagram for LUTImage::



### Public Member Functions

- **LUTImage** ()
- **LUTImage** (const **AImage** &i)
- **LUTImage** (int w, int h)
- **LUTImage** (const **LUTImage** &li)
- void **setLookUpTable** (const **LookUpTable** &t)
- void **setInLT** (int i, int v)
- int **getInLT** (int i) const

### Protected Attributes

- **LookUpTable** theLookUpTable

#### 9.78.1 Detailed Description

Definition at line 101 of file AImage.h.

#### 9.78.2 Constructor & Destructor Documentation

##### 9.78.2.1 LUTImage::LUTImage () [inline]

Definition at line 105 of file AImage.h.

##### 9.78.2.2 LUTImage::LUTImage (const AImage & i) [inline]

Definition at line 108 of file AImage.h.

**9.78.2.3 LUTImage::LUTImage (int *w*, int *h*) [inline]**

Definition at line 111 of file AImage.h.

**9.78.2.4 LUTImage::LUTImage (const LUTImage & *li*) [inline]**

Definition at line 113 of file AImage.h.

**9.78.3 Member Function Documentation****9.78.3.1 int LUTImage::getInLT (int *i*) const [inline]**

Definition at line 123 of file AImage.h.

References LookUpTable::get(), and theLookUpTable.

**9.78.3.2 void LUTImage::setInLT (int *i*, int *v*) [inline]**

Definition at line 119 of file AImage.h.

References LookUpTable::set(), and theLookUpTable.

**9.78.3.3 void LUTImage::setLookUpTable (const LookUpTable & *t*) [inline]**

Definition at line 115 of file AImage.h.

References theLookUpTable.

**9.78.4 Member Data Documentation****9.78.4.1 LookUpTable LUTImage::theLookUpTable [protected]**

Definition at line 129 of file AImage.h.

Referenced by CLUTImage::getBluePixel(), CLUTImage::getGreenPixel(), getInLT(), CLUTImage::getRedPixel(), setInLT(), and setLookUpTable().

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

- Graphics/include/AImage.h

## 9.79 Map Class Reference

Defines a representation of a map between two sets of words.

```
#include <Map.h>
```

### Public Member Functions

- **Map** (int domain\_gens, int range\_gens)  
*Default constructor.*
- **Map** (int domain\_gens, int range\_gens, const vector< **Word** > &generatingImages)  
*Constructor. Creates a map defined by the images of an alphabet.*
- **Map** (const **FiniteAlphabet** &domain\_alph, const **FiniteAlphabet** &range\_alph)  
*Constructor. Creates a trivial map.*
- **Map** (const **FiniteAlphabet** &domain\_alph, const **FiniteAlphabet** &range\_alph, const vector< **Word** > &generatingImages)  
*Constructor. Creates a map defined by the images of an alphabet.*
- bool **operator==** (const **Map** &m) const  
*Comparison operator.*
- int **domainSize** () const  
*Returns the size of the domain's alphabet.*
- const **FiniteAlphabet** & **domainAlphabet** () const  
*Returns domain's alphabet.*
- int **rangeSize** () const  
*Returns the size of the range's alphabet.*
- const **FiniteAlphabet** & **rangeAlphabet** () const  
*Returns range's alphabet.*
- const vector< **Word** > & **generatingImages** () const  
*Returns the vector of images.*
- **Word** **generatingImages** (int i) const  
*Return the image of the ith letter.*

- void **setGeneratingImages** (const vector< **Word** > gi)  
*Sets or modifies generating images.*
- void **setGeneratingImages** (int i, const **Word** &e)  
*Assigns to the i-th (0-based) generating image.*
- **Word imageOf** (const **Word** &w) const  
*Compute the image of a word.*
- **Map operator|** (const **Map** &secondMap)  
*Map (p. 320) composition operator.*
- void **printOn** (ostream &ostr) const  
*Stream output.*
- void **readFrom** (istream &in)  
*Stream input.*

## Private Member Functions

- char **readChar** (istream &in) const

## Private Attributes

- int **theDomain**
- int **theRange**
- vector< **Word** > **theGeneratingImages**
- **FiniteAlphabet theDomainAlphabet**
- **FiniteAlphabet theRangeAlphabet**

## Friends

- **Map composition** (const **Map** &firstMap, const **Map** &secondMap)  
*Map-theoretic composition.*
- ostream & **operator<<** (ostream &out, const **Map** &m)  
*Stream output operator.*
- istream & **operator>>** (istream &in, **Map** &m)  
*Stream input operator.*

### 9.79.1 Detailed Description

Defines a representation of a map between two sets of words. Both the domain and the range of a map are defined by finite alphabets.

Definition at line 22 of file Map.h.

### 9.79.2 Constructor & Destructor Documentation

#### 9.79.2.1 Map::Map (int *domain\_gens*, int *range\_gens*) [inline]

Default constructor. Constructor. Creates a trivial map.

If no images of letters of the domain are specified the map, by default, becomes the trivial map, i.e. the map mapping everything to the identity.

##### Parameters:

*domain\_gens* - the size of the domain's alphabet

*range\_gens* - the size of the range's alphabet

Definition at line 45 of file Map.h.

#### 9.79.2.2 Map::Map (int *domain\_gens*, int *range\_gens*, const vector< Word > & *generatingImages*) [inline]

Constructor. Creates a map defined by the images of an alphabet. Constructs a map, given a domain and range group, and images for the generators of the domain

##### Parameters:

*domain\_gens* - the size of the domain's alphabet

*range\_gens* - the size of the range's alphabet

*generatingImages* - vector of images of letters of the domain.

Definition at line 60 of file Map.h.

References `msgs::error()`.

#### 9.79.2.3 Map::Map (const FiniteAlphabet & *domain\_alph*, const FiniteAlphabet & *range\_alph*) [inline]

Constructor. Creates a trivial map. If no images of letters of the domain are specified the map, by default, becomes the trivial map, i.e. the map mapping everything to the identity.



**Parameters:**

*domain\_alph* - alphabet of the domain

*range\_alph* - alphabet of the range

Definition at line 80 of file Map.h.

**9.79.2.4** `Map::Map (const FiniteAlphabet & domain_alph, const FiniteAlphabet & range_alph, const vector< Word > & generatingImages) [inline]`

Constructor. Creates a map defined by the images of an alphabet. Constructs a map, given a domain and range group, and images for the generators of the domain

**Parameters:**

*domain\_alph* - alphabet of the domain

*range\_alph* - alphabet of the range

*generatingImages* - vector of images of letters of the domain.

Definition at line 95 of file Map.h.

References `msgs::error()`, and `theDomain`.

## 9.79.3 Member Function Documentation

**9.79.3.1** `const FiniteAlphabet& Map::domainAlphabet () const [inline]`

Returns domain's alphabet.

Definition at line 136 of file Map.h.

References `theDomainAlphabet`.

**9.79.3.2** `int Map::domainSize () const [inline]`

Returns the size of the domain's alphabet.

Definition at line 133 of file Map.h.

References `theDomain`.

**9.79.3.3** `Word Map::generatingImages (int i) const [inline]`

Return the image of the *i*th letter.

Definition at line 149 of file Map.h.

References `msgs::error()`, `theDomain`, and `theGeneratingImages`.

#### 9.79.3.4 `const vector<Word>& Map::generatingImages () const [inline]`

Returns the vector of images.

Definition at line 146 of file Map.h.

References `theGeneratingImages`.

Referenced by `__gnu_cxx::map_hash::operator()()`.

#### 9.79.3.5 `Word Map::imageOf (const Word & w) const`

Compute the image of a word. Takes a formal word in the generators of domain and evaluates its ‘image’ in range. The image is computed as follows. Let input word  $w = x_1x_2 \dots x_n$ , where  $x_i \in \text{Domain}$  and  $image(x) \in \text{Range}$  be the image of the letter  $x$ , then image of  $w$  -  $image(w) = image(x_1)image(x_2) \dots image(x_n)$

##### Parameters:

$w$  - a word in the domain

##### Returns:

an image of  $w$ .

#### 9.79.3.6 `bool Map::operator==(const Map & m) const [inline]`

Comparison operator. Two maps are equal if the sizes of domains and ranges coincide respectively and images are defined by the same words.

Definition at line 121 of file Map.h.

References `theDomain`, `theGeneratingImages`, and `theRange`.

#### 9.79.3.7 `Map Map::operator| (const Map & secondMap) [inline]`

**Map** (p. 320) composition operator. See **composition( ... )** (p. 326) for details.

Definition at line 218 of file Map.h.

References `composition`.

**9.79.3.8 void Map::printOn (ostream & *ostr*) const**

Stream output.

**9.79.3.9 const FiniteAlphabet& Map::rangeAlphabet () const [inline]**

Returns range's alphabet.

Definition at line 142 of file Map.h.

References theRangeAlphabet.

**9.79.3.10 int Map::rangeSize () const [inline]**

Returns the size of the range's alphabet.

Definition at line 139 of file Map.h.

References theRange.

**9.79.3.11 char Map::readChar (istream & *in*) const [private]****9.79.3.12 void Map::readFrom (istream & *in*)**

Stream input.

**9.79.3.13 void Map::setGeneratingImages (int *i*, const Word & *e*) [inline]**

Assigns to the *i*-th (0-based) generating image.

**Parameters:**

*i* - the index of a letter

*e* - the new image

Definition at line 170 of file Map.h.

References msgs::error(), theDomain, and theGeneratingImages.

**9.79.3.14 void Map::setGeneratingImages (const vector< Word > *gi*) [inline]**

Sets or modifies generating images.

Definition at line 157 of file Map.h.

References msgs::error(), theDomain, and theGeneratingImages.

## 9.79.4 Friends And Related Function Documentation

### 9.79.4.1 Map composition (const Map & *firstMap*, const Map & *secondMap*) [friend]

Map-theoretic composition. Returns the composition of two maps

**Parameters:**

*firstMap* - the first map

*secondMap* - the second map

**Returns:**

return the map constructed as a composition `secondMap(firstMap)`

Referenced by operator|().

### 9.79.4.2 ostream& operator<< (ostream & *out*, const Map & *m*) [friend]

Stream output operator.

Definition at line 233 of file Map.h.

### 9.79.4.3 istream& operator>> (istream & *in*, Map & *m*) [friend]

Stream input operator.

Definition at line 239 of file Map.h.

## 9.79.5 Member Data Documentation

### 9.79.5.1 int Map::theDomain [private]

Definition at line 254 of file Map.h.

Referenced by domainSize(), generatingImages(), Map(), operator==(), and setGeneratingImages().

### 9.79.5.2 FiniteAlphabet Map::theDomainAlphabet [private]

Definition at line 258 of file Map.h.

Referenced by domainAlphabet().

**9.79.5.3   vector<Word> Map::theGeneratingImages   [private]**

Definition at line 256 of file Map.h.

Referenced by generatingImages(), operator==(), and setGeneratingImages().

**9.79.5.4   int Map::theRange   [private]**

Definition at line 255 of file Map.h.

Referenced by operator==(), and rangeSize().

**9.79.5.5   FiniteAlphabet Map::theRangeAlphabet   [private]**

Definition at line 259 of file Map.h.

Referenced by rangeAlphabet().

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

- Maps/include/Map.h

## 9.80 `__gnu_cxx::map_hash` Struct Reference

```
#include <WhiteheadAutoSet.h>
```

### Public Member Functions

- `size_t operator() (Map m) const`

#### 9.80.1 Detailed Description

Definition at line 24 of file `WhiteheadAutoSet.h`.

#### 9.80.2 Member Function Documentation

##### 9.80.2.1 `size_t __gnu_cxx::map_hash::operator() (Map m) const` `[inline]`

Definition at line 26 of file `WhiteheadAutoSet.h`.

References `Map::generatingImages()`.

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

- `Maps/include/WhiteheadAutoSet.h`

## 9.81 PC::Marking Class Reference

```
#include <PowerCircuit.h>
```

### Public Member Functions

- **Marking** ()
- **Marking** (**PowerCircuit** \*p, int i)
- **Marking** (const **Marking** &m)
- **Marking** & **operator=** (const **Marking** &rhs)
- **~Marking** ()
- bool **isReduced** ()
- **Marking** clone ()
- **Marking** copy ()
- std::list< **Node** > **getNodes** ()
- int **getSign** (**Node** n)
- **Marking** **operator++** (int dummy)
- **Marking** **operator-** ()
- **Marking** **operator+** (const **Marking** &m)
- **Marking** **operator&** (const **Marking** &m)
- bool **operator<** (const **Marking** &m)
- bool **operator>** (const **Marking** &m)
- bool **operator<=** (const **Marking** &m)
- bool **operator>=** (const **Marking** &m)
- bool **operator==** (const **Marking** &m)
- bool **operator!=** (const **Marking** &m)
- bool **isSuccessorMarking** ()
- bool **isDefined** ()
- **Node** **getIncidentNode** ()
- **Node** **getSmallestNode** ()

### Public Attributes

- **PowerCircuit** \* **pc**
- int **id**

### 9.81.1 Detailed Description

Wrapper class for markings in power circuits. In reality, markings are handled by (derivatives of) the class **PowerCircuit** (p. 395). This class is used to make handling of markings type-safe. It contains only a pointer to the corresponding power circuit and an integer id, which is -1 if the marking has not (yet) been set. If any more data is needed for a marking, the power circuit class has to hold it itself. The methods call similarly-named methods of **PowerCircuit** (p. 395).

Markings work like references to the respective internal representations. Thus the assignment operator does not create a copy of a marking. However all arithmetic operations create new markings, so the old ones can be maintained. Therefore a marking (i.e. the handle of a internal representation) only changes its value through the assignment operator and when calling the `connect()` or `connectInv()` methods of **PowerCircuit** (p. 395).

It is also not necessary (and not possible) to delete markings (in the **PowerCircuit-Graph** (p. 418) class, once there is no marking pointing onto an internal marking, its memory is automatically deallocated).

Definition at line 132 of file `PowerCircuit.h`.

### 9.81.2 Constructor & Destructor Documentation

#### 9.81.2.1 `PC::Marking::Marking () [inline]`

Definition at line 138 of file `PowerCircuit.h`.



9.81.2.2 PC::Marking::Marking (PowerCircuit \* *p*, int *i*)

9.81.2.3 PC::Marking::Marking (const Marking & *m*)

9.81.2.4 PC::Marking::~~Marking ()

### 9.81.3 Member Function Documentation

9.81.3.1 Marking PC::Marking::clone ()

9.81.3.2 Marking PC::Marking::copy ()

9.81.3.3 Node PC::Marking::getIncidentNode ()

9.81.3.4 std::list<Node> PC::Marking::getNodes ()

9.81.3.5 int PC::Marking::getSign (Node *n*)

9.81.3.6 Node PC::Marking::getSmallestNode ()

9.81.3.7 bool PC::Marking::isDefined ()

9.81.3.8 bool PC::Marking::isReduced ()

9.81.3.9 bool PC::Marking::isSuccessorMarking ()

9.81.3.10 bool PC::Marking::operator!= (const Marking & *m*)

9.81.3.11 Marking PC::Marking::operator& (const Marking & *m*)

9.81.3.12 Marking PC::Marking::operator+ (const Marking & *m*)

9.81.3.13 Marking PC::Marking::operator++ (int *dummy*)

9.81.3.14 Marking PC::Marking::operator- ()

9.81.3.15 bool PC::Marking::operator< (const Marking & *m*)

9.81.3.16 bool PC::Marking::operator<= (const Marking & *m*)

9.81.3.17 Marking& PC::Marking::operator= (const Marking & *rhs*)

9.81.3.18 bool PC::Marking::operator== (const Marking & *m*)

9.81.3.19 bool PC::Marking::operator> (const Marking & *m*)

9.81.3.20 bool PC::Marking::operator>= (const Marking & *m*)

### 9.81.4 Member Data Documentation

9.81.4.1 int PC::Marking::id

Definition at line 136 of file PowerCircuit.h.

#### 9.81.4.2 **PowerCircuit\* PC::Marking::pc**

Definition at line 135 of file PowerCircuit.h.

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

- HigmanGroup/include/**PowerCircuit.h**

## 9.82 MatrixFp Class Reference

```
#include <AEProtocol.h>
```

### Public Member Functions

- **MatrixFp** (int n, int p)
- **~MatrixFp** ()
- **MatrixFp** (const **MatrixFp** &m)
- **MatrixFp & operator=** (const **MatrixFp** &m)
- **MatrixFp operator+** (const **MatrixFp** &w) const
- **MatrixFp operator\*** (const **MatrixFp** &w) const
- **MatrixFp scalar\_mult** (int l) const
- **MatrixFp getPower** (int e) const
- void **set** (int i, int j, int v)

### Static Public Member Functions

- static **MatrixFp random** (int n, int p)
- static **MatrixFp ID** (int n, int p)

### Private Member Functions

- void **init** ()
- void **clean** ()

### Private Attributes

- int **the\_n**
- int **the\_p**
- vector< vector< int > > **theMatrix**

#### 9.82.1 Detailed Description

Definition at line 167 of file AEProtocol.h.

## 9.82.2 Constructor & Destructor Documentation

### 9.82.2.1 `MatrixFp::MatrixFp (int $n$ , int $p$ )` `[inline]`

Definition at line 170 of file AEProtocol.h.

References `init()`.

### 9.82.2.2 `MatrixFp::~~MatrixFp ()` `[inline]`

Definition at line 171 of file AEProtocol.h.

References `clean()`.

### 9.82.2.3 `MatrixFp::MatrixFp (const MatrixFp & $m$ )`

## 9.82.3 Member Function Documentation

### 9.82.3.1 `void MatrixFp::clean ()` `[private]`

Referenced by `~MatrixFp()`.

### 9.82.3.2 `MatrixFp MatrixFp::getPower (int $e$ ) const`

### 9.82.3.3 `static MatrixFp MatrixFp::ID (int $n$ , int $p$ )` `[static]`

### 9.82.3.4 `void MatrixFp::init ()` `[private]`

Referenced by `MatrixFp()`.

**9.82.3.5** `MatrixFp MatrixFp::operator* (const MatrixFp & w) const` `[inline]`

**9.82.3.6** `MatrixFp MatrixFp::operator+ (const MatrixFp & w) const` `[inline]`

**9.82.3.7** `MatrixFp& MatrixFp::operator= (const MatrixFp & m)`

**9.82.3.8** `static MatrixFp MatrixFp::random (int n, int p)` `[static]`

**9.82.3.9** `MatrixFp MatrixFp::scalar_mult (int l) const` `[inline]`

**9.82.3.10** `void MatrixFp::set (int i, int j, int v)` `[inline]`

Definition at line 183 of file AEProtocol.h.

References theMatrix.

## 9.82.4 Member Data Documentation

**9.82.4.1** `int MatrixFp::the_n` `[private]`

Definition at line 192 of file AEProtocol.h.

**9.82.4.2** `int MatrixFp::the_p` `[private]`

Definition at line 193 of file AEProtocol.h.

**9.82.4.3** `vector< vector<int> > MatrixFp::theMatrix` `[private]`

Definition at line 195 of file AEProtocol.h.

Referenced by set().

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

- CryptoAE/include/AEProtocol.h

## 9.83 MaxCommutePartition Class Reference

```
#include <DCBraidReduction.h>
```

### Public Member Functions

- **MaxCommutePartition** (int *n*, const **Word** &*w*)
- const vector< **CommDivider** > &**getPartition** () const

### Private Member Functions

- void **findCommutParts** (const **Word** &*w*)

### Private Attributes

- int *N*
- vector< **CommDivider** > *partition*

### 9.83.1 Detailed Description

Definition at line 77 of file DCBraidReduction.h.

### 9.83.2 Constructor & Destructor Documentation

#### 9.83.2.1 MaxCommutePartition::MaxCommutePartition (int *n*, const **Word** &*w*) [**inline**]

Definition at line 80 of file DCBraidReduction.h.

References **findCommutParts**().

### 9.83.3 Member Function Documentation

#### 9.83.3.1 void MaxCommutePartition::findCommutParts (const **Word** &*w*) [**inline**, **private**]

Definition at line 88 of file DCBraidReduction.h.

References **CommDivider::begin**, **Word::begin**(), **CommDivider::end**, **Word::end**(), **CommDivider::gen**, **CommDivider::length**, *N*, and *partition*.

Referenced by **MaxCommutePartition**().

### 9.83.3.2 `const vector<CommDivider>& MaxCommutePartition::getPartition()` `const` `[inline]`

Definition at line 85 of file DCBraidReduction.h.

References `partition`.

## 9.83.4 Member Data Documentation

### 9.83.4.1 `int MaxCommutePartition::N` `[private]`

Definition at line 121 of file DCBraidReduction.h.

Referenced by `findCommutParts()`.

### 9.83.4.2 `vector<CommDivider> MaxCommutePartition::partition` `[private]`

Definition at line 122 of file DCBraidReduction.h.

Referenced by `findCommutParts()`, and `getPartition()`.

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

- `Experiments/include/DCBraidReduction.h`

## 9.84 MotivePattern Struct Reference

```
#include <MotivePatternWrapper.h>
```

### Public Attributes

- int occurrences
- int sequences
- Word thePattern

### 9.84.1 Detailed Description

Definition at line 31 of file MotivePatternWrapper.h.

### 9.84.2 Member Data Documentation

#### 9.84.2.1 int MotivePattern::occurrences

Definition at line 33 of file MotivePatternWrapper.h.

#### 9.84.2.2 int MotivePattern::sequences

Definition at line 34 of file MotivePatternWrapper.h.

#### 9.84.2.3 Word MotivePattern::thePattern

Definition at line 36 of file MotivePatternWrapper.h.

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

- StringSimilarity/include/MotivePatternWrapper.h



## 9.85 MotivePatternWrapper Class Reference

```
#include <MotivePatternWrapper.h>
```

### Public Member Functions

- **MotivePatternWrapper** (const **FreeGroup** &f)
- void **initializeSequence** (const vector< **Word** > &)
- bool **compute** ()
- vector< **MotivePattern** > **getPatterns** ()

### Private Attributes

- string **input\_file**
- string **output\_file**
- **FreeGroup** **theGroup**

#### 9.85.1 Detailed Description

Definition at line 45 of file MotivePatternWrapper.h.

#### 9.85.2 Constructor & Destructor Documentation

##### 9.85.2.1 MotivePatternWrapper::MotivePatternWrapper (const FreeGroup &f) [inline]

Definition at line 48 of file MotivePatternWrapper.h.

### 9.85.3 Member Function Documentation

**9.85.3.1** `bool MotivePatternWrapper::compute ()`

**9.85.3.2** `vector<MotivePattern> MotivePatternWrapper::getPatterns ()`

**9.85.3.3** `void MotivePatternWrapper::initializeSequence (const vector< Word  
> &)`

### 9.85.4 Member Data Documentation

**9.85.4.1** `string MotivePatternWrapper::input_file [private]`

Definition at line 54 of file MotivePatternWrapper.h.

**9.85.4.2** `string MotivePatternWrapper::output_file [private]`

Definition at line 55 of file MotivePatternWrapper.h.

**9.85.4.3** `FreeGroup MotivePatternWrapper::theGroup [private]`

Definition at line 58 of file MotivePatternWrapper.h.

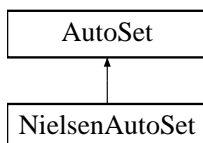
The documentation for this class was generated from the following file:

- StringSimilarity/include/MotivePatternWrapper.h

## 9.86 NielsenAutoSet Class Reference

Implements a set of Nielsen automorphisms of a free group.

`#include <WhiteheadAutoSet.h>`Inheritance diagram for NielsenAutoSet::



### Public Member Functions

- **NielsenAutoSet** (int n)  
*Constructor. Generates a set of Nielsen automorphisms.*
- **~NielsenAutoSet** ()
- const **Map** & **getRandomAuto** () const  
*Returns a random Nielsen automorphism.*
- const **SetOfMaps** & **getSet** () const  
*Returns the set of Nielsen automorphisms.*

### Private Attributes

- **SetOfMaps** theSet
- int nGens

#### 9.86.1 Detailed Description

Implements a set of Nielsen automorphisms of a free group.

Definition at line 54 of file WhiteheadAutoSet.h.

#### 9.86.2 Constructor & Destructor Documentation

##### 9.86.2.1 NielsenAutoSet::NielsenAutoSet (int n)

Constructor. Generates a set of Nielsen automorphisms. The set of Nielsen automorphisms for a free group of rank n is generated.

Note, the whole set is generated by enumeration.  $(n^2)$  automorphisms.

**Parameters:**

$n$  - rank of a free group.

**9.86.2.2 NielsenAutoSet::~~NielsenAutoSet () [inline]**

Definition at line 74 of file WhiteheadAutoSet.h.

**9.86.3 Member Function Documentation****9.86.3.1 const Map& NielsenAutoSet::getRandomAuto () const**

Returns a random Nielsen automorphism.

**Returns:**

Nielsen automorphisms selected uniformly randomly from the set.

**9.86.3.2 const SetOfMaps& NielsenAutoSet::getSet () const [inline, virtual]**

Returns the set of Nielsen automorphisms.

**Returns:**

the set of Nielsen automorphisms.

Implements **AutoSet** (p. 110).

Definition at line 93 of file WhiteheadAutoSet.h.

References theSet.

**9.86.4 Member Data Documentation****9.86.4.1 int NielsenAutoSet::nGens [private]**

Definition at line 96 of file WhiteheadAutoSet.h.

#### 9.86.4.2 SetOfMaps NielsenAutoSet::theSet [private]

Definition at line 95 of file WhiteheadAutoSet.h.

Referenced by `getSet()`.

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

- Maps/include/**WhiteheadAutoSet.h**

## 9.87 PC::Node Class Reference

```
#include <PowerCircuit.h>
```

### Public Member Functions

- **Node** ()
- **Node** (**PowerCircuit** \*p, int i)
- bool **isReduced** ()
- bool **isDefined** ()
- **Node** clone ()
- int **getOrder** ()
- **Marking** **getSuccessorMarking** ()
- bool **operator==** (**Node** n)

### Public Attributes

- **PowerCircuit** \* **pc**
- int **id**

#### 9.87.1 Detailed Description

Wrapper class for nodes in power circuits. In reality, nodes are handled by (derivatives of) the class **PowerCircuit** (p. 395). This class is used to make handling of nodes type-safe. It contains only a pointer to the corresponding power circuit and an integer id, which is -1 if the node has not (yet) been set. If any more data is needed for a node, the power circuit class has to hold it itself. The methods call similarly-named methods of **PowerCircuit** (p. 395).

Definition at line 89 of file PowerCircuit.h.

#### 9.87.2 Constructor & Destructor Documentation

##### 9.87.2.1 PC::Node::Node () [inline]

Definition at line 95 of file PowerCircuit.h.

##### 9.87.2.2 PC::Node::Node (PowerCircuit \*p, int i) [inline]

Definition at line 96 of file PowerCircuit.h.

### 9.87.3 Member Function Documentation

**9.87.3.1** Node PC::Node::clone ()

**9.87.3.2** int PC::Node::getOrder ()

**9.87.3.3** Marking PC::Node::getSuccessorMarking ()

**9.87.3.4** bool PC::Node::isDefined ()

**9.87.3.5** bool PC::Node::isReduced ()

**9.87.3.6** bool PC::Node::operator== (Node *n*)

### 9.87.4 Member Data Documentation

**9.87.4.1** int PC::Node::id

Definition at line 93 of file PowerCircuit.h.

**9.87.4.2** PowerCircuit\* PC::Node::pc

Definition at line 92 of file PowerCircuit.h.

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

- HigmanGroup/include/PowerCircuit.h

## 9.88 PC::PowerCircuitGraph::NodeUsedByType Struct Reference

### Public Member Functions

- **NodeUsedByType** (std::list< std::pair< **Node**, **Sign** > > \*l, std::list< std::pair< **Node**, **Sign** > >::iterator iter, int i)
- bool **operator==** (const **NodeUsedByType** &t2)

### Public Attributes

- std::list< std::pair< **Node**, **Sign** > > \* **nodeList**
- std::list< std::pair< **Node**, **Sign** > >::iterator **position**
- int **id**

### 9.88.1 Detailed Description

Definition at line 41 of file PowerCircuitGraph.h.

### 9.88.2 Constructor & Destructor Documentation

- 9.88.2.1 PC::PowerCircuitGraph::NodeUsedByType::NodeUsedByType**  
(std::list< std::pair< **Node**, **Sign** > > \*l, std::list< std::pair< **Node**, **Sign** > >::iterator *iter*, int *i*) [**inline**]

Definition at line 46 of file PowerCircuitGraph.h.

### 9.88.3 Member Function Documentation

- 9.88.3.1 bool PC::PowerCircuitGraph::NodeUsedByType::operator==** (const **NodeUsedByType** &*t2*) [**inline**]

Definition at line 49 of file PowerCircuitGraph.h.

References id.

### 9.88.4 Member Data Documentation

- 9.88.4.1 int PC::PowerCircuitGraph::NodeUsedByType::id**

Definition at line 45 of file PowerCircuitGraph.h.



Referenced by operator==( ).

**9.88.4.2** `std::list< std::pair<Node,Sign> >*`  
`PC::PowerCircuitGraph::NodeUsedByType::nodeList`

Definition at line 43 of file PowerCircuitGraph.h.

**9.88.4.3** `std::list< std::pair<Node,Sign> >::iterator`  
`PC::PowerCircuitGraph::NodeUsedByType::position`

Definition at line 44 of file PowerCircuitGraph.h.

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

- HigmanGroup/include/**PowerCircuitGraph.h**

## 9.89 ObjectOf< Rep > Class Template Reference

```
#include <ObjectOf.h>
```

### Public Member Functions

- **ObjectOf** (const **ObjectOf** &o)
- **~ObjectOf** ()
- **ObjectOf** & **operator=** (const **ObjectOf** &o)

### Protected Member Functions

- const Rep \* **look** () const
- Rep \* **enhance** () const
- Rep \* **change** ()
- void **acquireRep** (const Rep \*rep)
- **ObjectOf** (Rep \*newrep)

### Private Member Functions

- void **force\_derivation** ()

### Private Attributes

- Rep \* **theRep**

### 9.89.1 Detailed Description

```
template<class Rep> class ObjectOf< Rep >
```

Definition at line 12 of file ObjectOf.h.

### 9.89.2 Constructor & Destructor Documentation

**9.89.2.1** `template<class Rep> ObjectOf< Rep >::ObjectOf (const ObjectOf< Rep > &o) [inline]`

Definition at line 23 of file ObjectOf.h.

**9.89.2.2    template<class Rep> ObjectOf< Rep >::~~ObjectOf ()    [inline]**

Definition at line 25 of file ObjectOf.h.

**9.89.2.3    template<class Rep> ObjectOf< Rep >::ObjectOf (Rep \* *newrep*)  
                 [inline, protected]**

Definition at line 101 of file ObjectOf.h.

**9.89.3    Member Function Documentation****9.89.3.1    template<class Rep> void ObjectOf< Rep >::acquireRep (const Rep  
                 \* *rep*)    [inline, protected]**

Definition at line 80 of file ObjectOf.h.

**9.89.3.2    template<class Rep> Rep\* ObjectOf< Rep >::change ()    [inline,  
                 protected]**

Definition at line 74 of file ObjectOf.h.

Referenced by Word::cyclicallyPermute(), PowerWord::cyclicallyPermute(),  
Word::cyclicallyReduce(), PowerWord::cyclicallyReduce(),  
Word::cyclicallyReduceWord(), PowerWord::cyclicallyReduceWord(),  
Word::getPower(), PowerWord::getPower(), Word::initialSegment(), Power-  
Word::initialSegment(), Word::inverse(), PowerWord::inverse(), Word::operator-  
(), PowerWord::operator-(), Word::segment(), PowerWord::segment(),  
Word::terminalSegment(), and PowerWord::terminalSegment().

**9.89.3.3    template<class Rep> Rep\* ObjectOf< Rep >::enhance () const  
                 [inline, protected]**

Definition at line 70 of file ObjectOf.h.

**9.89.3.4    template<class Rep> void ObjectOf< Rep >::force\_derivation ()  
                 [inline, private]**

Definition at line 120 of file ObjectOf.h.

### 9.89.3.5 `template<class Rep> const Rep* ObjectOf< Rep >::look () const [inline, protected]`

Definition at line 67 of file ObjectOf.h.

Referenced by `FSA::addFSA()`, `PowerWord::insert()`, `Word::operator!=()`, `PowerWord::operator!=()`, `Word::operator*=(())`, `PowerWord::operator*=(())`, `Word::operator<()`, `PowerWord::operator<()`, `Word::operator==(())`, `PowerWord::operator==(())`, `Word::operator>()`, `PowerWord::operator>()`, and `Word::operator^=(())`.

### 9.89.3.6 `template<class Rep> ObjectOf& ObjectOf< Rep >::operator= (const ObjectOf< Rep > & o) [inline]`

Definition at line 34 of file ObjectOf.h.

## 9.89.4 Member Data Documentation

### 9.89.4.1 `template<class Rep> Rep* ObjectOf< Rep >::theRep [private]`

Definition at line 112 of file ObjectOf.h.

Referenced by `ObjectOf< WordRep >::acquireRep()`, `ObjectOf< WordRep >::change()`, `ObjectOf< WordRep >::enhance()`, `ObjectOf< WordRep >::force_derivation()`, `ObjectOf< WordRep >::look()`, `ObjectOf< WordRep >::ObjectOf()`, `ObjectOf< WordRep >::operator=()`, and `ObjectOf< WordRep >::~~ObjectOf()`.

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

- `general/include/ObjectOf.h`

## 9.90 PairDistanceSimilarityTest Class Reference

Implements a hypothesis testing that two words are similar according to some given criteria.

```
#include <PairDistanceTest.h>
```

### Public Member Functions

- **PairDistanceSimilarityTest** (int *n*, **PairGenerator** \**g*, **StringSimilarityMeasure** \**sm*)

*Constructor.*

- void **estimateTrueDistribution** ()
- double **testTruePair** ()
- double **testFalsePair** ()
- double **testPair** (const pair< **Word**, **Word** > &*p*)

### Private Attributes

- vector< double > **measureDistr**
- **StringSimilarityMeasure** \* **theSimilarity**
- int **sampleSize**
- **PairGenerator** \* **theGenerator**

#### 9.90.1 Detailed Description

Implements a hypothesis testing that two words are similar according to some given criteria.

Definition at line 97 of file PairDistanceTest.h.

#### 9.90.2 Constructor & Destructor Documentation

##### 9.90.2.1 PairDistanceSimilarityTest::PairDistanceSimilarityTest (int *n*, **PairGenerator** \**g*, **StringSimilarityMeasure** \**sm*) [inline]

Constructor.

Definition at line 101 of file PairDistanceTest.h.

### 9.90.3 Member Function Documentation

**9.90.3.1** `void PairDistanceSimilarityTest::estimateTrueDistribution ()`

**9.90.3.2** `double PairDistanceSimilarityTest::testFalsePair ()`

**9.90.3.3** `double PairDistanceSimilarityTest::testPair (const pair< Word, Word  
> & p)`

**9.90.3.4** `double PairDistanceSimilarityTest::testTruePair ()`

### 9.90.4 Member Data Documentation

**9.90.4.1** `vector<double> PairDistanceSimilarityTest::measureDistr  
[private]`

Definition at line 115 of file PairDistanceTest.h.

**9.90.4.2** `int PairDistanceSimilarityTest::sampleSize [private]`

Definition at line 118 of file PairDistanceTest.h.

**9.90.4.3** `PairGenerator* PairDistanceSimilarityTest::theGenerator  
[private]`

Definition at line 119 of file PairDistanceTest.h.

**9.90.4.4** `StringSimilarityMeasure* PairDistanceSimilarityTest::theSimilarity  
[private]`

Definition at line 116 of file PairDistanceTest.h.

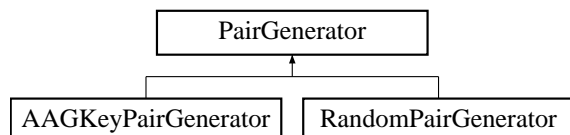
The documentation for this class was generated from the following file:

- StringSimilarity/include/**PairDistanceTest.h**

## 9.91 PairGenerator Class Reference

Abstract interface for classes defining sword pairs similarities.

`#include <PairDistanceTest.h>` Inheritance diagram for PairGenerator::



### Public Member Functions

- virtual `pair< Word, Word > getTruePair ()=0`  
*Returns a pair of words similar under specified criteria.*
- virtual `pair< Word, Word > getFalsePair ()=0`  
*Returns a pair of words that are not similar under specified criteria.*

#### 9.91.1 Detailed Description

Abstract interface for classes defining sword pairs similarities.

Definition at line 36 of file PairDistanceTest.h.

#### 9.91.2 Member Function Documentation

##### 9.91.2.1 virtual `pair<Word,Word> PairGenerator::getFalsePair ()` [pure virtual]

Returns a pair of words that are not similar under specified criteria.

Implemented in **AAGKeyPairGenerator** (p. 88), and **RandomPairGenerator** (p. 464).

##### 9.91.2.2 virtual `pair<Word,Word> PairGenerator::getTruePair ()` [pure virtual]

Returns a pair of words similar under specified criteria.

Implemented in **AAGKeyPairGenerator** (p. 88), and **RandomPairGenerator** (p. 464).

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

- StringSimilarity/include/**PairDistanceTest.h**



## 9.92 Parser Class Reference

```
#include <Parser.h>
```

### Public Member Functions

- **Parser** (istream &in, const **Alphabet** \*a)
- **~Parser** ()
- void **parse** ()
- char **getWordTerminalSymbol** () const
- const list< int > & **getWord** () const

### Private Attributes

- class WordFlexLexer \* **localFlexLexer**
- list< int > **theWord**
- char **theTS**

#### 9.92.1 Detailed Description

Definition at line 16 of file Parser.h.

#### 9.92.2 Constructor & Destructor Documentation

**9.92.2.1** **Parser::Parser** (istream & *in*, const Alphabet \* *a*)

**9.92.2.2** **Parser::~~Parser** ()

#### 9.92.3 Member Function Documentation

**9.92.3.1** const list<int>& **Parser::getWord** () const

**9.92.3.2** char **Parser::getWordTerminalSymbol** () const

**9.92.3.3** void **Parser::parse** ()

#### 9.92.4 Member Data Documentation

**9.92.4.1** class WordFlexLexer\* **Parser::localFlexLexer** [**private**]

Definition at line 27 of file Parser.h.

**9.92.4.2 char Parser::theTS [private]**

Definition at line 29 of file Parser.h.

**9.92.4.3 list<int> Parser::theWord [private]**

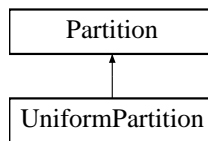
Definition at line 28 of file Parser.h.

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

- Alphabet/include/**Parser.h**

## 9.93 Partition Class Reference

#include <DCBraidReduction.h> Inheritance diagram for Partition::



### Public Member Functions

- virtual list< int > **getPartition** (const **Word** &w)=0

#### 9.93.1 Detailed Description

Definition at line 19 of file DCBraidReduction.h.

#### 9.93.2 Member Function Documentation

##### 9.93.2.1 virtual list<int> Partition::getPartition (const Word & w) [pure virtual]

Implemented in **UniformPartition** (p. 614).

Referenced by DCBraidReduction::shorten().

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

- Experiments/include/**DCBraidReduction.h**

## 9.94 PBar Struct Reference

```
#include <FormatOutput.h>
```

### Public Member Functions

- **PBar** (int *i*)
- **PBar** (double *i*)
- **PBar** (int *i*)
- **PBar** (double *i*)

### Public Attributes

- double **progress**

### Friends

- ostream & **operator**<< (ostream &out, const **PBar** &pb)
- ostream & **operator**<< (ostream &out, const **PBar** &pb)

#### 9.94.1 Detailed Description

Definition at line 7 of file FormatOutput.h.

#### 9.94.2 Constructor & Destructor Documentation

##### 9.94.2.1 PBar::PBar (int *i*) [inline]

Definition at line 9 of file FormatOutput.h.

##### 9.94.2.2 PBar::PBar (double *i*) [inline]

Definition at line 10 of file FormatOutput.h.

##### 9.94.2.3 PBar::PBar (int *i*) [inline]

Definition at line 9 of file ProgressBar.h.

#### 9.94.2.4 PBar::PBar (double *i*) [inline]

Definition at line 10 of file ProgressBar.h.

### 9.94.3 Friends And Related Function Documentation

#### 9.94.3.1 ostream& operator<< (ostream & *out*, const PBar & *pb*) [friend]

Definition at line 12 of file ProgressBar.h.

#### 9.94.3.2 ostream& operator<< (ostream & *out*, const PBar & *pb*) [friend]

Definition at line 12 of file FormatOutput.h.

### 9.94.4 Member Data Documentation

#### 9.94.4.1 double PBar::progress

Definition at line 19 of file FormatOutput.h.

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

- general/include/FormatOutput.h
- general/include/ProgressBar.h

## 9.95 PDFPage Class Reference

Class implements a page of a pdf document.

```
#include <PDFgraphing.h>
```

### Public Member Functions

- **PDFPage ()**
- **~PDFPage ()**
- void **writeContents** (ostream &os)  
*Write the contents of the page into a stream using PDF format.*
- void **addObject** (PDFPageObject \*obj)  
*Add a pdf object to a page.*

### Static Public Member Functions

- static int **width** ()  
*The actual width of the page (no margins).*
- static int **height** ()  
*The actual height of the page (no margins).*
- static int **mWidth** ()  
*Width of the page excluding margins.*
- static int **mHeight** ()  
*Height of the page excluding margins.*
- static int **mRightEnd** ()  
*Coordinate of the right end of the page (excluding right margin).*
- static int **mBottomEnd** ()  
*Coordinate of the bottom end of the page (excluding the margin).*
- static int **lMargin** ()  
*Left margin.*
- static int **rMargin** ()  
*Right margin.*

- static int **tMargin** ()  
*Top margin.*
- static int **bMargin** ()  
*Bottom margin.*

## Private Attributes

- vector< PDFPageObject \* > **theObjects**

### 9.95.1 Detailed Description

Class implements a page of a pdf document.

Definition at line 375 of file PDFgraphing.h.

### 9.95.2 Constructor & Destructor Documentation

#### 9.95.2.1 PDFPage::PDFPage () [inline]

Definition at line 385 of file PDFgraphing.h.

#### 9.95.2.2 PDFPage::~~PDFPage () [inline]

Definition at line 386 of file PDFgraphing.h.

References theObjects.

### 9.95.3 Member Function Documentation

#### 9.95.3.1 void PDFPage::addObject (PDFPageObject \* *obj*) [inline]

Add a pdf object to a page.

Definition at line 398 of file PDFgraphing.h.

References theObjects.

**9.95.3.2 static int PDFPage::bMargin () [inline, static]**

Bottom margin.

Definition at line 421 of file PDFgraphing.h.

Referenced by mHeight().

**9.95.3.3 static int PDFPage::height () [inline, static]**

The actual height of the page (no margins).

Definition at line 403 of file PDFgraphing.h.

Referenced by mHeight().

**9.95.3.4 static int PDFPage::lMargin () [inline, static]**

Left margin.

Definition at line 415 of file PDFgraphing.h.

Referenced by BraidDrawPDF::drawGenerator(), BraidDrawPDF::drawGrid(), mRightEnd(), and mWidth().

**9.95.3.5 static int PDFPage::mBottomEnd () [inline, static]**

Coordinate of the bottom end of the page (excluding the margin).

Definition at line 412 of file PDFgraphing.h.

References mHeight(), and tMargin().

**9.95.3.6 static int PDFPage::mHeight () [inline, static]**

Height of the page excluding margins.

Definition at line 408 of file PDFgraphing.h.

References bMargin(), height(), and tMargin().

Referenced by mBottomEnd(), and BraidDrawPDF::stripesInPage().

**9.95.3.7 static int PDFPage::mRightEnd () [inline, static]**

Coordinate of the right end of the page (excluding right margin).

Definition at line 410 of file PDFgraphing.h.



References lMargin(), and mWidth().

#### **9.95.3.8 static int PDFPage::mWidth () [inline, static]**

Width of the page excluding margins.

Definition at line 406 of file PDFgraphing.h.

References lMargin(), rMargin(), and width().

Referenced by BraidDrawPDF::drawGrid(), mRightEnd(), and BraidDrawPDF::posInPage().

#### **9.95.3.9 static int PDFPage::rMargin () [inline, static]**

Right margin.

Definition at line 417 of file PDFgraphing.h.

Referenced by mWidth().

#### **9.95.3.10 static int PDFPage::tMargin () [inline, static]**

Top margin.

Definition at line 419 of file PDFgraphing.h.

Referenced by BraidDrawPDF::drawGrid(), mBottomEnd(), mHeight(), and BraidDrawPDF::stripPosition().

#### **9.95.3.11 static int PDFPage::width () [inline, static]**

The actual width of the page (no margins).

Definition at line 401 of file PDFgraphing.h.

Referenced by mWidth().

#### **9.95.3.12 void PDFPage::writeContents (ostream & os) [inline]**

Write the contents of the page into a stream using PDF format.

Definition at line 392 of file PDFgraphing.h.

References theObjects.

## 9.95.4 Member Data Documentation

### 9.95.4.1 `vector< PDFPageObject* > PDFPage::theObjects` `[private]`

Definition at line 430 of file PDFgraphing.h.

Referenced by `addObject()`, `writeContents()`, and `~PDFPage()`.

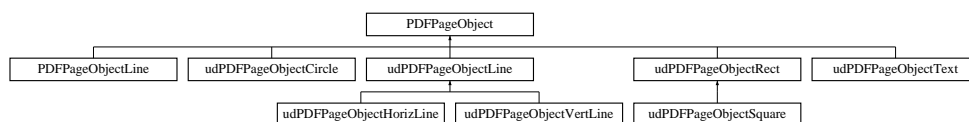
The documentation for this class was generated from the following file:

- Graphics/include/PDFgraphing.h

## 9.96 PDFPageObject Class Reference

Implements interface for PDF drawing objects.

`#include <PDFgraphing.h>` Inheritance diagram for PDFPageObject::



### Public Member Functions

- virtual void **write** (ostream &os)

#### 9.96.1 Detailed Description

Implements interface for PDF drawing objects.

Definition at line 28 of file PDFgraphing.h.

#### 9.96.2 Member Function Documentation

##### 9.96.2.1 virtual void PDFPageObject::write (ostream & os) [inline, virtual]

Reimplemented in **udPDFPageObjectText** (p. 611), **PDFPageObjectLine** (p. 367), **udPDFPageObjectLine** (p. 603), **udPDFPageObjectRect** (p. 606), and **udPDFPageObjectCircle** (p. 599).

Definition at line 47 of file PDFgraphing.h.

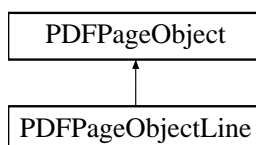
The documentation for this class was generated from the following file:

- Graphics/include/**PDFgraphing.h**

## 9.97 PDFPageObjectLine Class Reference

Class for pdf line object.

#include <PDFgraphing.h> Inheritance diagram for PDFPageObjectLine::



### Public Member Functions

- **PDFPageObjectLine** (double x1, double y1, double x2, double y2)  
*Constructor of a line pdf object.*
- void **write** (ostream &os)

### Private Attributes

- double **theX1**
- double **theY1**
- double **theX2**
- double **theY2**

### 9.97.1 Detailed Description

Class for pdf line object.

Definition at line 103 of file PDFgraphing.h.

### 9.97.2 Constructor & Destructor Documentation

#### 9.97.2.1 PDFPageObjectLine::PDFPageObjectLine (double x1, double y1, double x2, double y2) [inline]

Constructor of a line pdf object.

#### Parameters:

*x1,y1,x2,y2* - coordinates of the two endpoints of a line

Definition at line 116 of file PDFgraphing.h.

### 9.97.3 Member Function Documentation

#### 9.97.3.1 void PDFPageObjectLine::write (ostream & os) [inline, virtual]

Reimplemented from **PDFPageObject** (p. 365).

Definition at line 126 of file PDFgraphing.h.

References theX1, theX2, theY1, and theY2.

### 9.97.4 Member Data Documentation

#### 9.97.4.1 double PDFPageObjectLine::theX1 [private]

Definition at line 139 of file PDFgraphing.h.

Referenced by write().

#### 9.97.4.2 double PDFPageObjectLine::theX2 [private]

Definition at line 140 of file PDFgraphing.h.

Referenced by write().

#### 9.97.4.3 double PDFPageObjectLine::theY1 [private]

Definition at line 139 of file PDFgraphing.h.

Referenced by write().

#### 9.97.4.4 double PDFPageObjectLine::theY2 [private]

Definition at line 140 of file PDFgraphing.h.

Referenced by write().

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

- Graphics/include/PDFgraphing.h

## 9.98 PDFStructure Class Reference

Implements a pdf document consisting of several pages.

```
#include <PDFgraphing.h>
```

### Public Member Functions

- **PDFStructure** ()
- **~PDFStructure** ()
- void **save** (const char \*filename)  
*Save pdf document into a file "filename".*
- void **newPage** ()  
*Create a new page.*
- void **addObject** (int p, **PDFPageObject** \*obj)  
*Add a new object to the page.*

### Private Member Functions

- pair< const char \*, int > **preparePageContents** (int p) const

### Private Attributes

- vector< **PDFPage** \* > **thePages**

#### 9.98.1 Detailed Description

Implements a pdf document consisting of several pages.

Definition at line 441 of file PDFgraphing.h.

#### 9.98.2 Constructor & Destructor Documentation

##### 9.98.2.1 PDFStructure::PDFStructure () [inline]

Definition at line 452 of file PDFgraphing.h.

### 9.98.2.2 PDFStructure::~~PDFStructure () [inline]

Definition at line 453 of file PDFgraphing.h.

References thePages.

## 9.98.3 Member Function Documentation

### 9.98.3.1 void PDFStructure::addObject (int *p*, PDFPageObject \* *obj*) [inline]

Add a new object to the page.

#### Parameters:

*p* - page number

*obj* - pointer to the new pdf object instance

Definition at line 479 of file PDFgraphing.h.

References thePages.

Referenced by BraidDrawPDF::drawGenerator(), and BraidDrawPDF::drawGrid().

### 9.98.3.2 void PDFStructure::newPage () [inline]

Create a new page.

Definition at line 470 of file PDFgraphing.h.

References thePages.

Referenced by BraidDrawPDF::draw(), and BraidDrawPDF::drawCompressedBraid().

### 9.98.3.3 pair< const char\*, int > PDFStructure::preparePageContents (int *p*) const [private]

### 9.98.3.4 void PDFStructure::save (const char \* *filename*)

Save pdf document into a file "filename".

Referenced by BraidDrawPDF::save().

## 9.98.4 Member Data Documentation

### 9.98.4.1 `vector< PDFPage* > PDFStructure::thePages` `[private]`

Definition at line 499 of file PDFgraphing.h.

Referenced by `addObject()`, `newPage()`, and `~PDFStructure()`.

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

- Graphics/include/**PDFgraphing.h**



## 9.99 Permutation Class Reference

**Permutation** (p. 371).

```
#include <Permutation.h>
```

### Classes

- struct **triple**

### Public Member Functions

- **Permutation** (int size=0)  
*Default constructor (size specifies the size of the permutation).*
- **Permutation** (int size, const int \*p)  
*Construct a permutation of the specified size given by the array p.*
- **Permutation** (const vector< int > &p)  
*Construct a permutation by a vector of numbers.*
- template<class IntIterator >  
**Permutation** (const IntIterator &B, const IntIterator &E)  
*Construct a permutation by a sequence of numbers.*
- int **operator**[ ] (int ind) const  
*Get the ith element of the permutation.*
- int & **operator**[ ] (int ind)  
*Get the reference to the ith element of the permutation.*
- bool **operator**== (const **Permutation** &p) const  
*Check if 2 permutations are equal.*
- bool **operator**!= (const **Permutation** &p) const  
*Check if 2 permutations are not equal.*
- bool **operator**< (const **Permutation** &p) const  
*Check if \*this is strictly less than p (compared lexicographically).*
- **Permutation operator**\* (const **Permutation** &p) const  
*Multiple 2 permutations.*

- **Permutation & operator\*=** (const **Permutation** &p)  
*Multiple 1 permutation by another on the right.*
- **Permutation operator-** () const  
*Invert the permutation.*
- **Permutation inverse** () const  
*Invert the permutation.*
- **Permutation & left\_mult\_by\_cycle** (const vector< int > &cycle)  
*A function used for computation of BKL normal forms of braids.*
- void **change** (int i, int j)  
*Swap the values at uth and jth position (multiply this by a cycle (i,j) on the left).*
- const vector< int > & **getVector** () const  
*Get the vector of numbers which represents the permutation.*
- **Permutation power** (int p) const  
*Raise the permutation into the power p.*
- int **size** () const  
*Get the size of the permutation.*
- **Permutation increaseSize** (int N) const  
*Increase the size of a permutation.*
- int **length** () const  
*Compute the length of a permutation (length of a geodesic). Current version is slow, need to update.*
- int **difference** (const **Permutation** &p) const  
*Compute the number of positions with different elements.*
- **Permutation computeConjugacyClassRepresentative** (**Permutation** &conj) const  
*Compute the conjugacy class representative (2 elements conjugate iff they have the same result of this function).*
- **Permutation computeConjugator** (const **Permutation** &p) const  
*Compute a conjugator for a couple of permutations (if permutations are not conjugate then the result makes no sense).*

- **Permutation flip** () const  
*Flip the permutation (conjugate by a half-twist permutation  $\Delta$ ).*
- **Permutation tinyFlip** (int sh) const  
*Perform a tiny flip (conjugate by  $\delta$ ).*
- **bool isTrivial** () const  
*Check if the permutation is trivial.*
- **vector< int > geodesic** () const  
*Find a geodesic word representing the permutation (indices start with 0).*
- **vector< int > geodesicWord** () const  
*Find a geodesic word representing the permutation (indices start with 1).*
- **vector< int > getWordPresentation** () const  
*Compute the word presentation for a permutation which is "parallel descending cycles" (for other permutations it makes no sense).*
- **Permutation RightGCD** (const **Permutation** &p) const  
*Compute RightGCD of 2 permutations.*
- **Permutation RightLCM** (const **Permutation** &p) const  
*Compute RightLCM of 2 permutations.*
- **Permutation LeftGCD** (const **Permutation** &p) const  
*Compute LeftGCD of 2 permutations.*
- **Permutation LeftLCM** (const **Permutation** &p) const  
*Compute LeftLCM of 2 permutations.*
- **Permutation meet2** (const **Permutation** &p) const  
*Compute GCD for 2 permutations that are "parallel descending cycles" (used for BKL Normal Forms of braids).*
- **Permutation join2** (const **Permutation** &p) const  
*Compute LCM for 2 permutations that are "parallel descending cycles" (used for BKL Normal Forms of braids).*

## Static Public Member Functions

- static **Permutation CYCLE** (int N, const vector< int > &cycle)  
*Copy constructor provided by compiler.*
- static void **lr\_multiply\_by\_cycles** (**Permutation** &P, **Permutation** &I, const vector< int > &M1, const vector< int > &M2)  
*A function used for computation of BKL normal forms of braids.*
- static **Permutation random** (int size)  
*Generate a random permutation of specified size.*
- static **Permutation getHalfTwistPermutation** (int size)  
*Get half twist permutation  $\Delta = (n - 1, n - 2, \dots, 2, 1, 0)$ .*
- static **Permutation getCyclePermutation** (int size)  
*Get half twist permutation  $\Delta = (1, 2, \dots, n - 2, n - 1, 0)$ .*
- static bool **mixable** (const **Permutation** &p1, const **Permutation** &p2)  
*Check if 2 permutations are mixable (it is not used anywhere in the system, god knows that means).*

## Private Member Functions

- void **\_sub\_meet** (const **Permutation** &p, const **Permutation** &ip1, const **Permutation** &ip2, **Permutation** &cur, int \*left\_indeces\_a, int \*left\_indeces\_b, int \*right\_indeces\_a, int \*right\_indeces\_b, int beg, int end) const  
*(Aux) The main operation to compute RightGCD and all other lattice functions*
- void **prepare\_pairs** (int N, **Permutation** &P, vector< pair< int, int > > &pairs) const

## Private Attributes

- vector< int > **theValue**  
*A vector representing the permutation.*

## Friends

- class **BraidGroup**
- ostream & **operator<<** (ostream &os, const **Permutation** &p)

## 9.99.1 Detailed Description

**Permutation** (p. 371). We use the standard representation of a presentation on  $n$  symbols - a sequence of numbers  $(x_0, \dots, x_{n-1})$ . Notice that indexes start at 0.

Definition at line 33 of file Permutation.h.

## 9.99.2 Constructor & Destructor Documentation

### 9.99.2.1 Permutation::Permutation (int *size* = 0)

Default constructor (size specifies the size of the permutation).

### 9.99.2.2 Permutation::Permutation (int *size*, const int \* *p*)

Construct a permutation of the specified size given by the array *p*.

### 9.99.2.3 Permutation::Permutation (const vector< int > & *p*)

Construct a permutation by a vector of numbers.

### 9.99.2.4 template<class IntIterator > Permutation::Permutation (const IntIterator & *B*, const IntIterator & *E*) [inline]

Construct a permutation by a sequence of numbers.

Definition at line 55 of file Permutation.h.

## 9.99.3 Member Function Documentation

### 9.99.3.1 void Permutation::\_sub\_meet (const Permutation & *p*, const Permutation & *ip1*, const Permutation & *ip2*, Permutation & *cur*, int \* *left\_indeces\_a*, int \* *left\_indeces\_b*, int \* *right\_indeces\_a*, int \* *right\_indeces\_b*, int *beg*, int *end*) const [private]

(Aux) The main operation to compute RightGCD and all other lattice functions

### 9.99.3.2 void Permutation::change (int *i*, int *j*) [inline]

Swap the values at *uth* and *jth* position (multiply this by a cycle (i,j) on the left).

Definition at line 130 of file Permutation.h.

References theValue.

### 9.99.3.3 **Permutation** **Permutation::computeConjugacyClassRepresentative** (Permutation & *conj*) const

Compute the conjugacy class representative (2 elements conjugate iff they have the same result of this function).

### 9.99.3.4 **Permutation** **Permutation::computeConjugator** (const Permutation & *p*) const

Compute a conjugator for a couple of permutations (if permutations are not conjugate then the result makes no sense).

### 9.99.3.5 **static** **Permutation** **Permutation::CYCLE** (int *N*, const vector< int > & *cycle*) [**static**]

Copy constructor provided by compiler. What is it?

### 9.99.3.6 **int** **Permutation::difference** (const Permutation & *p*) const

Compute the number of positions with different elements.

### 9.99.3.7 **Permutation** **Permutation::flip** () const

Flip the permutation (conjugate by a half-twist permutation  $\Delta$ ).

### 9.99.3.8 **vector<int>** **Permutation::geodesic** () const

Find a geodesic word representing the permutation (indices start with 0).

### 9.99.3.9 **vector<int>** **Permutation::geodesicWord** () const

Find a geodesic word representing the permutation (indices start with 1).

### 9.99.3.10 **static** **Permutation** **Permutation::getCyclePermutation** (int *size*) [**static**]

Get half twist permutation  $\Delta = (1, 2, \dots, n-2, n-1, 0)$ .

**9.99.3.11 static Permutation Permutation::getHalfTwistPermutation (int *size*) [static]**

Get half twist permutation  $\Delta = (n - 1, n - 2, \dots, 2, 1, 0)$ .

Referenced by ThLeftNormalForm::ThLeftNormalForm(), and ThRightNormalForm::ThRightNormalForm().

**9.99.3.12 const vector< int >& Permutation::getVector () const [inline]**

Get the vector of numbers which represents the permutation.

Definition at line 134 of file Permutation.h.

References theValue.

**9.99.3.13 vector< int > Permutation::getWordPresentation () const**

Compute the word presentation for a permutation which is "parallel descending cycles" (for other permutations it makes no sense).

**9.99.3.14 Permutation Permutation::increaseSize (int *N*) const**

Increase the size of a permutation.

**9.99.3.15 Permutation Permutation::inverse () const**

Invert the permutation.

Referenced by operator-().

**9.99.3.16 bool Permutation::isTrivial () const**

Check if the permutation is trivial.

Referenced by ThLeftNormalForm::ThLeftNormalForm(), and ThRightNormalForm::ThRightNormalForm().

**9.99.3.17 Permutation Permutation::join2 (const Permutation & *p*) const**

Compute LCM for 2 permutations that are "parallel descending cycles" (used for BKL Normal Forms of braids).

### 9.99.3.18 **Permutation& Permutation::left\_mult\_by\_cycle (const vector< int > & cycle)**

A function used for computation of BKL normal forms of braids.

### 9.99.3.19 **Permutation Permutation::LeftGCD (const Permutation & p) const**

Compute LeftGCD of 2 permutations. Let  $p_1$  and  $p_2$  be two permutations. The maximal permutation  $P$  which ends  $p_1$  and  $p_2$  is called the left greatest common divisor of  $p_1$  and  $p_2$ , i.e.,  $P$  is maximal such that  $p_1 = P \circ d_1$  and  $p_2 = P \circ d_1$  for some permutations  $d_1$  and  $d_2$ .

### 9.99.3.20 **Permutation Permutation::LeftLCM (const Permutation & p) const**

Compute LeftLCM of 2 permutations. Let  $p_1$  and  $p_2$  be two permutations. The minimal permutation  $P$  which starts with  $p_1$  and  $p_2$  is called the left least common multiple, i.e.,  $P$  is minimal such that  $P = p_1 \circ d_1$  and  $P = p_2 \circ d_2$ .

### 9.99.3.21 **int Permutation::length () const**

Compute the length of a permutation (length of a geodesic). Current version is slow, need to update.

### 9.99.3.22 **static void Permutation::lr\_multiply\_by\_cycles (Permutation & P, Permutation & I, const vector< int > & M1, const vector< int > & M2) [static]**

A function used for computation of BKL normal forms of braids.

### 9.99.3.23 **Permutation Permutation::meet2 (const Permutation & p) const**

Compute GCD for 2 permutations that are "parallel descending cycles" (used for BKL Normal Forms of braids).

### 9.99.3.24 **static bool Permutation::mixable (const Permutation & p1, const Permutation & p2) [static]**

Check if 2 permutations are mixable (it is not used anywhere in the system, god knows that means).



**9.99.3.25** `bool Permutation::operator!=(const Permutation & p) const`

Check if 2 permutations are not equal.

**9.99.3.26** `Permutation Permutation::operator*(const Permutation & p) const`

Multiple 2 permutations.

**9.99.3.27** `Permutation& Permutation::operator*=(const Permutation & p)`

Multiple 1 permutation by another on the right.

**9.99.3.28** `Permutation Permutation::operator- () const [inline]`

Invert the permutation.

Definition at line 103 of file Permutation.h.

References `inverse()`.

**9.99.3.29** `bool Permutation::operator<(const Permutation & p) const`

Check if \*this is strictly less than p (compared lexicographically).

**9.99.3.30** `bool Permutation::operator==(const Permutation & p) const`

Check if 2 permutations are equal.

**9.99.3.31** `int& Permutation::operator[] (int ind) [inline]`

Get the reference to the ith element of the permutation.

Definition at line 80 of file Permutation.h.

References `theValue`.

**9.99.3.32** `int Permutation::operator[] (int ind) const [inline]`

Get the ith element of the permutation.

Definition at line 76 of file Permutation.h.

References `theValue`.

**9.99.3.33 Permutation Permutation::power (int  $p$ ) const**

Raise the permutation into the power  $p$ .

**9.99.3.34 void Permutation::prepare\_pairs (int  $N$ , Permutation &  $P$ , vector< pair< int, int > > &  $pairs$ ) const [private]****9.99.3.35 static Permutation Permutation::random (int  $size$ ) [static]**

Generate a random permutation of specified size.

**9.99.3.36 Permutation Permutation::RightGCD (const Permutation &  $p$ ) const**

Compute RightGCD of 2 permutations. Let  $p_1$  and  $p_2$  be two permutations. The maximal permutation  $P$  which ends  $p_1$  and  $p_2$  is called the right greatest common divisor of  $p_1$  and  $p_2$ , i.e.,  $P$  is maximal such that  $p_1 = d_1 \circ P$  and  $p_2 = d_2 \circ P$  for some permutations  $d_1$  and  $d_2$ .

**9.99.3.37 Permutation Permutation::RightLCM (const Permutation &  $p$ ) const**

Compute RightLCM of 2 permutations. Let  $p_1$  and  $p_2$  be two permutations. The minimal permutation  $P$  which ends with  $p_1$  and  $p_2$  is called the right least common multiple, i.e.,  $P$  is minimal such that  $P = d_1 \circ p_1$  and  $P = d_2 \circ p_2$ .

**9.99.3.38 int Permutation::size () const [inline]**

Get the size of the permutation.

Definition at line 142 of file Permutation.h.

References theValue.

Referenced by ThLeftNormalForm::ThLeftNormalForm(), and ThRightNormalForm::ThRightNormalForm().

**9.99.3.39 Permutation Permutation::tinyFlip (int  $sh$ ) const**

Perform a tiny flip (conjugate by  $\delta$ ).

## 9.99.4 Friends And Related Function Documentation

### 9.99.4.1 friend class BraidGroup [friend]

Definition at line 303 of file Permutation.h.

### 9.99.4.2 ostream& operator<< (ostream & *os*, const Permutation & *p*) [friend]

## 9.99.5 Member Data Documentation

### 9.99.5.1 vector< int > Permutation::theValue [private]

A vector representing the permutation.

Definition at line 308 of file Permutation.h.

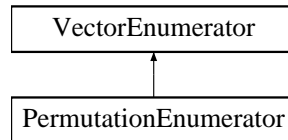
Referenced by change(), getVector(), operator[](), and size().

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

- general/include/**Permutation.h**

## 9.100 PermutationEnumerator Class Reference

#include <PermutationEnumerator.h> Inheritance diagram for PermutationEnumerator::



### Public Member Functions

- **PermutationEnumerator** (int L)  
*Constructor.*
- **PermutationEnumerator operator=** (const **PermutationEnumerator** &)  
*No assignment operator.*
- **PermutationEnumerator** (const **PermutationEnumerator** &)  
*No copy constructor.*
- **Permutation getPermutation** () const
- virtual bool **seqOK** () const  
*Check if the currently constructed sequence is legitimate.*
- virtual bool **seqLimit** () const  
*Returns true if the length-limit on the sequence is passed.*
- virtual bool **seqComplete** () const  
*Returns true if the current sequence is complete.*
- virtual int **start** () const  
*Returns the initial value for the current element of the sequence (usually 0).*
- virtual int **next** (int cur) const  
*Returns the next value for the current element of the sequence (usually cur+1).*
- virtual bool **finish** (int cur) const
- virtual void **stepTo** ()  
*Function is invoked when the current value of the current element of the sequence is accepted and we go for the next element.*

- virtual void **stepBack** (int cur)

*Function is invoked when we go backward from the current element to the previous element.*

## Private Attributes

- int **theLength**

*The length of a permutation.*

- vector< bool > **theUsedElements**

*Array of flags - elements used in already constructed part of a permutation.*

### 9.100.1 Detailed Description

Definition at line 14 of file PermutationEnumerator.h.

### 9.100.2 Constructor & Destructor Documentation

#### 9.100.2.1 PermutationEnumerator::PermutationEnumerator (int *L*) [inline]

Constructor.

Definition at line 26 of file PermutationEnumerator.h.

#### 9.100.2.2 PermutationEnumerator::PermutationEnumerator (const PermutationEnumerator &)

No copy constructor.

### 9.100.3 Member Function Documentation

#### 9.100.3.1 virtual bool PermutationEnumerator::finish (int *cur*) const [inline, virtual]

Implements **VectorEnumerator** (p. 622).

Definition at line 64 of file PermutationEnumerator.h.

References **theLength**.

**9.100.3.2** `PermutationEnumerator::getPermutation () const`  
`[inline]`

Definition at line 44 of file `PermutationEnumerator.h`.

References `VectorEnumerator::getSeq()`, and `theLength`.

**9.100.3.3** `virtual int PermutationEnumerator::next (int cur) const` `[inline, virtual]`

Returns the next value for the current element of the sequence (usually `cur+1`).

Implements **VectorEnumerator** (p. 622).

Definition at line 61 of file `PermutationEnumerator.h`.

**9.100.3.4** `PermutationEnumerator PermutationEnumerator::operator= (const PermutationEnumerator &)`

No assignment operator.

**9.100.3.5** `virtual bool PermutationEnumerator::seqComplete () const`  
`[inline, virtual]`

Returns true if the current sequence is complete.

Implements **VectorEnumerator** (p. 623).

Definition at line 54 of file `PermutationEnumerator.h`.

References `VectorEnumerator::getLength()`, and `theLength`.

**9.100.3.6** `virtual bool PermutationEnumerator::seqLimit () const` `[inline, virtual]`

Returns true if the length-limit on the sequence is passed.

Implements **VectorEnumerator** (p. 623).

Definition at line 51 of file `PermutationEnumerator.h`.

References `VectorEnumerator::getLength()`, and `theLength`.

**9.100.3.7** `virtual bool PermutationEnumerator::seqOK () const` `[inline, virtual]`

Check if the currently constructed sequence is legitimate.

Implements **VectorEnumerator** (p. 623).

Definition at line 46 of file PermutationEnumerator.h.

References `VectorEnumerator::getLength()`, `VectorEnumerator::getSeq()`, `theLength`, and `theUsedElements`.

#### 9.100.3.8 `virtual int PermutationEnumerator::start () const [inline, virtual]`

Returns the initial value for the current element of the sequence (usually 0).

Implements **VectorEnumerator** (p. 623).

Definition at line 58 of file PermutationEnumerator.h.

#### 9.100.3.9 `virtual void PermutationEnumerator::stepBack (int cur) [inline, virtual]`

Function is invoked when we go backward from the current element to the previous element. The value of `curElement` is already decreased, so you are at the previous element

Reimplemented from **VectorEnumerator** (p. 624).

Definition at line 72 of file PermutationEnumerator.h.

References `VectorEnumerator::getLength()`, `VectorEnumerator::getSeq()`, and `theUsedElements`.

#### 9.100.3.10 `virtual void PermutationEnumerator::stepTo () [inline, virtual]`

Function is invoked when the current value of the current element of the sequence is accepted and we go for the next element. The value of `curElement` is already increased, so you are at the next element.

Reimplemented from **VectorEnumerator** (p. 624).

Definition at line 68 of file PermutationEnumerator.h.

References `VectorEnumerator::getLength()`, `VectorEnumerator::getSeq()`, and `theUsedElements`.

## 9.100.4 Member Data Documentation

### 9.100.4.1 `int PermutationEnumerator::theLength` `[private]`

The length of a permutation.

Definition at line 87 of file `PermutationEnumerator.h`.

Referenced by `finish()`, `getPermutation()`, `seqComplete()`, `seqLimit()`, and `seqOK()`.

### 9.100.4.2 `vector< bool > PermutationEnumerator::theUsedElements` `[private]`

Array of flags - elements used in already constructed part of a permutation.

Definition at line 90 of file `PermutationEnumerator.h`.

Referenced by `seqOK()`, `stepBack()`, and `stepTo()`.

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

- `general/include/PermutationEnumerator.h`



## 9.101 Perturbation Class Reference

```
#include <DCBraidReduction.h>
```

### Public Member Functions

- virtual **Word** perturb (const **Word** &w)=0

#### 9.101.1 Detailed Description

Definition at line 25 of file DCBraidReduction.h.

#### 9.101.2 Member Function Documentation

##### 9.101.2.1 virtual **Word** Perturbation::perturb (const **Word** & w) [pure virtual]

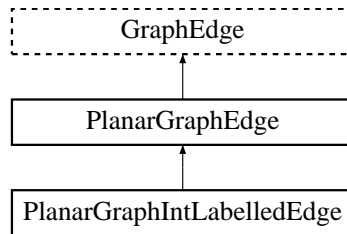
The documentation for this class was generated from the following file:

- Experiments/include/**DCBraidReduction.h**

## 9.102 PlanarGraphEdge Struct Reference

Defines non-labelled edge of the directed planar graph.

`#include <GraphType.h>` Inheritance diagram for PlanarGraphEdge::



### Public Member Functions

- **PlanarGraphEdge** (int **target**, int cell1=-1, int cell2=-1)  
*Constructor for an edge. Argument t is a number of the target.*
- **PlanarGraphEdge** ()  
*Dummy constructor, not to be used, required for STL::map. Also, can be used to denote "failure" or "dead-end" edge.*
- **PlanarGraphEdge inverse** (int origin)  
*Invert an edge.*
- bool **operator<** (const **PlanarGraphEdge** &e) const  
*Check if one edge is less than the other.*
- bool **operator==** (const **PlanarGraphEdge** &e) const  
*Check if two edges equal.*
- bool **operator!=** (const **PlanarGraphEdge** &e) const  
*Check if two edges are not equal.*

### Public Attributes

- int **theCell1**  
*The left neighbor-cell of the edge. Numbering of cells starts from 0, -1 denotes the outer face.*

- int **theCell2**

*The right neighbor-cell of the edge. Numbering of cells starts from 0, -1 denotes the outer face.*

### 9.102.1 Detailed Description

Defines non-labelled edge of the directed planar graph.

Definition at line 150 of file GraphType.h.

### 9.102.2 Constructor & Destructor Documentation

#### 9.102.2.1 PlanarGraphEdge::PlanarGraphEdge (int *target*, int *cell1* = -1, int *cell2* = -1) [inline]

Constructor for an edge. Argument t is a number of the target.

Definition at line 154 of file GraphType.h.

#### 9.102.2.2 PlanarGraphEdge::PlanarGraphEdge () [inline]

Dummy constructor, not to be used, required for STL::map. Also, can be used to denote "failure" or "dead-end" edge.

Definition at line 158 of file GraphType.h.

Referenced by inverse().

### 9.102.3 Member Function Documentation

#### 9.102.3.1 PlanarGraphEdge PlanarGraphEdge::inverse (int *origin*) [inline]

Invert an edge.

Reimplemented from **GraphEdge** (p. 253).

Reimplemented in **PlanarGraphIntLabelledEdge** (p. 393).

Definition at line 162 of file GraphType.h.

References PlanarGraphEdge(), theCell1, and theCell2.

### 9.102.3.2 **bool PlanarGraphEdge::operator!= (const PlanarGraphEdge & e)** **const [inline]**

Check if two edges are not equal.

Reimplemented from **GraphEdge** (p. 253).

Reimplemented in **PlanarGraphIntLabelledEdge** (p. 393).

Definition at line 180 of file GraphType.h.

References theCell1, theCell2, and GraphEdge::theTarget.

### 9.102.3.3 **bool PlanarGraphEdge::operator< (const PlanarGraphEdge & e)** **const [inline]**

Check if one edge is less than the other.

Reimplemented from **GraphEdge** (p. 253).

Reimplemented in **PlanarGraphIntLabelledEdge** (p. 394).

Definition at line 166 of file GraphType.h.

References theCell1, theCell2, and GraphEdge::theTarget.

### 9.102.3.4 **bool PlanarGraphEdge::operator== (const PlanarGraphEdge & e)** **const [inline]**

Check if two edges equal.

Reimplemented from **GraphEdge** (p. 254).

Reimplemented in **PlanarGraphIntLabelledEdge** (p. 394).

Definition at line 176 of file GraphType.h.

References theCell1, theCell2, and GraphEdge::theTarget.

## 9.102.4 Member Data Documentation

### 9.102.4.1 **int PlanarGraphEdge::theCell1**

The left neighbor-cell of the edge. Numbering of cells starts from 0, -1 denotes the outer face.

Definition at line 184 of file GraphType.h.

Referenced by PlanarGraphIntLabelledEdge::inverse(), inverse(), PlanarGraphIntLabelledEdge::operator!=(), operator!==( ), PlanarGraphIntLabelledEdge::operator<(), operator<=(), PlanarGraphIntLabelledEdge::operator==( ), and operator==( ).

#### 9.102.4.2 int PlanarGraphEdge::theCell2

The right neighbor-cell of the edge. Numbering of cells starts from 0, -1 denotes the outer face.

Definition at line 188 of file GraphType.h.

Referenced by PlanarGraphIntLabelledEdge::inverse(), inverse(), PlanarGraphIntLabelledEdge::operator!=(), operator!=(), PlanarGraphIntLabelledEdge::operator<(), operator<(), PlanarGraphIntLabelledEdge::operator==(), and operator==().

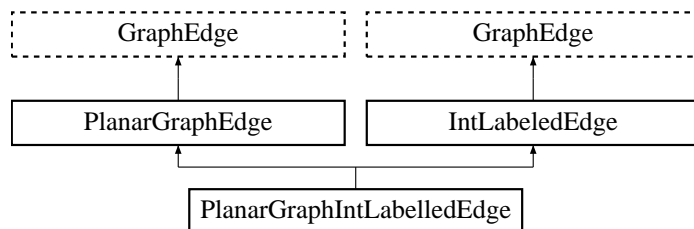
The documentation for this struct was generated from the following file:

- Graph/include/**GraphType.h**

### 9.103 PlanarGraphIntLabelledEdge Struct Reference

Defines non-labelled edge of the directed planar graph.

#include <GraphType.h> Inheritance diagram for PlanarGraphIntLabelledEdge::



#### Public Member Functions

- **PlanarGraphIntLabelledEdge** (int **target**, int label, int cell1=-1, int cell2=-1)  
*Constructor for an edge. Argument t is a number of the target.*
- **PlanarGraphIntLabelledEdge** ()  
*Dummy constructor, not to be used, required for STL::map. Also, can be used to denote "failure" or "dead-end" edge.*
- **PlanarGraphIntLabelledEdge inverse** (int origin)  
*Invert an edge.*
- bool **operator<** (const **PlanarGraphIntLabelledEdge** &e) const  
*Check if one edge is less than the other.*
- bool **operator==** (const **PlanarGraphIntLabelledEdge** &e) const  
*Check if two edges equal.*
- bool **operator!=** (const **PlanarGraphIntLabelledEdge** &e) const  
*Check if two edges are not equal.*

#### 9.103.1 Detailed Description

Defines non-labelled edge of the directed planar graph.

Definition at line 198 of file GraphType.h.

## 9.103.2 Constructor & Destructor Documentation

### 9.103.2.1 PlanarGraphIntLabelledEdge::PlanarGraphIntLabelledEdge (int *target*, int *label*, int *cell1* = -1, int *cell2* = -1) [inline]

Constructor for an edge. Argument *t* is a number of the target.

Definition at line 202 of file GraphType.h.

### 9.103.2.2 PlanarGraphIntLabelledEdge::PlanarGraphIntLabelledEdge () [inline]

Dummy constructor, not to be used, required for STL::map. Also, can be used to denote "failure" or "dead-end" edge.

Definition at line 206 of file GraphType.h.

Referenced by `inverse()`.

## 9.103.3 Member Function Documentation

### 9.103.3.1 PlanarGraphIntLabelledEdge PlanarGraphIntLabelledEdge::inverse (int *origin*) [inline]

Invert an edge.

Reimplemented from `IntLabeledEdge` (p. 278).

Definition at line 210 of file GraphType.h.

References `PlanarGraphIntLabelledEdge()`, `PlanarGraphEdge::theCell1`, `PlanarGraphEdge::theCell2`, and `IntLabeledEdge::theLabel`.

### 9.103.3.2 bool PlanarGraphIntLabelledEdge::operator!= (const PlanarGraphIntLabelledEdge & *e*) const [inline]

Check if two edges are not equal.

Reimplemented from `IntLabeledEdge` (p. 278).

Definition at line 230 of file GraphType.h.

References `PlanarGraphEdge::theCell1`, `PlanarGraphEdge::theCell2`, `IntLabeledEdge::theLabel`, and `GraphEdge::theTarget`.

### 9.103.3.3 **bool PlanarGraphIntLabelledEdge::operator< (const PlanarGraphIntLabelledEdge & e) const [inline]**

Check if one edge is less than the other.

Reimplemented from **IntLabeledEdge** (p. 279).

Definition at line 214 of file GraphType.h.

References `PlanarGraphEdge::theCell1`, `PlanarGraphEdge::theCell2`, `IntLabeledEdge::theLabel`, and `GraphEdge::theTarget`.

### 9.103.3.4 **bool PlanarGraphIntLabelledEdge::operator== (const PlanarGraphIntLabelledEdge & e) const [inline]**

Check if two edges equal.

Reimplemented from **IntLabeledEdge** (p. 279).

Definition at line 226 of file GraphType.h.

References `PlanarGraphEdge::theCell1`, `PlanarGraphEdge::theCell2`, `IntLabeledEdge::theLabel`, and `GraphEdge::theTarget`.

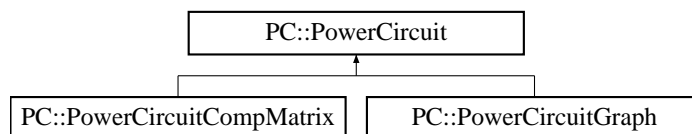
The documentation for this struct was generated from the following file:

- `Graph/include/GraphType.h`



## 9.104 PC::PowerCircuit Class Reference

#include <PowerCircuit.h> Inheritance diagram for PC::PowerCircuit::



### Public Member Functions

- **PowerCircuit** ()
- virtual **~PowerCircuit** ()
- virtual void **reduce** ()=0
- virtual void **reduce** (std::vector< **Node** > &nodeVector, std::vector< **Marking** > &markingVector)=0
- virtual **Marking** **createMarking** (int i=0)=0
- virtual **Marking** **createMarking** (const std::list< **Node** > &nodes)=0
- virtual **Marking** **createMarking** (const std::list< **Node** > &nodeList, const std::list< int > &signList)=0
- virtual **Marking** **createMarkingFromNodes** (unsigned int numNodes,...)
- virtual **Node** **createNode** (const **Marking** &succ)=0
- virtual void **connect** (const **Marking** &m, const **Marking** &p)=0
- virtual void **connectInv** (const **Marking** &m, const **Marking** &p)=0
- virtual void **remove** (const **Marking** &m)=0
- virtual **Node** **getReducedNode** (unsigned int ord)=0
- virtual std::list< **Node** > **getNodes** ()=0
- virtual std::list< **Marking** > **getMarkings** ()=0
- virtual std::list< **Node** > **getMarkingNodes** (const **Marking** &m)=0
- virtual **PowerCircuit** \* **clone** (std::vector< **Marking** > &markingsToClone)=0
- virtual void **draw** (std::string filename, **Marking** highlight1=**Marking**(), **Marking** highlight2=**Marking**(), **Marking** highlight3=**Marking**(), **Marking** highlight4=**Marking**(), **Marking** highlight5=**Marking**(), **Marking** highlight6=**Marking**(), **Marking** highlight7=**Marking**(), **Marking** highlight8=**Marking**(), **Marking** highlight9=**Marking**())
- int **getNumEdges** ()
- int **getNumNodes** ()
- int **getNumMarkings** ()
- virtual void **printStatistics** (std::ostream &os=std::cout)
- virtual void **print** (std::ostream &os=std::cout)=0

## Private Member Functions

- virtual void **incMarkingRefCount** (const **Marking** &mark)=0
- virtual void **decMarkingRefCount** (**Marking** &mark)=0
- virtual **Marking** **addMarkings** (const **Marking** &m1, const **Marking** &m2)=0
- virtual **Marking** **invMarking** (const **Marking** &m)=0
- virtual **Marking** **incMarking** (const **Marking** &m)=0
- virtual **Marking** **intersectMarkings** (const **Marking** &m1, const **Marking** &m2)=0
- virtual **Marking** **cloneMarking** (const **Marking** &m)=0
- virtual **Node** **cloneNode** (**Node** n)=0
- virtual **Marking** **copyMarking** (const **Marking** &m)=0
- virtual bool **isMarkingReduced** (const **Marking** &m) const =0
- virtual int **compareMarkings** (const **Marking** &m1, const **Marking** &m2)=0
- virtual int **getRedNodeOrd** (**Node** n)=0
- virtual int **getNodeSignInMarking** (**Node** n, const **Marking** &m) const =0
- virtual bool **isSuccessorMarking** (const **Marking** &m) const =0
- virtual **Node** **getIncidentNode** (const **Marking** &m)=0
- virtual **Node** **getSmallestNodeInMarking** (const **Marking** &m)=0
- virtual **Marking** **getSuccMarking** (**Node** n)=0

## Friends

- class **Node**
- class **Marking**

### 9.104.1 Detailed Description

Abstract base class for all implementations of power circuits. The **PowerCircuit** (p. 395) class provides methods to create markings, perform the reduction procedure, draw edges in the graph (connect), and some additional tasks.

Definition at line 170 of file PowerCircuit.h.

### 9.104.2 Constructor & Destructor Documentation

#### 9.104.2.1 **PC::PowerCircuit::PowerCircuit ()** [**inline**]

Definition at line 200 of file PowerCircuit.h.

**9.104.2.2** virtual PC::PowerCircuit::~~PowerCircuit () [inline, virtual]

Definition at line 201 of file PowerCircuit.h.

### 9.104.3 Member Function Documentation

**9.104.3.1** virtual Marking PC::PowerCircuit::addMarkings (const Marking & *m1*, const Marking & *m2*) [private, pure virtual]

Implemented in PC::PowerCircuitCompMatrix (p. 407), and PC::PowerCircuitGraph (p. 421).

**9.104.3.2** virtual PowerCircuit\* PC::PowerCircuit::clone (std::vector< Marking > & *markingsToClone*) [pure virtual]

Makes a copy of a **PowerCircuit** (p. 395). Only the markings in the *markingsToClone* list are available in the cloned circuit. Reduced nodes and markings are remain reduced. The clone has to be deallocated via delete, once it is not needed anymore!

Implemented in PC::PowerCircuitCompMatrix (p. 408), and PC::PowerCircuitGraph (p. 421).

**9.104.3.3** virtual Marking PC::PowerCircuit::cloneMarking (const Marking & *m*) [private, pure virtual]

Implemented in PC::PowerCircuitCompMatrix (p. 408), and PC::PowerCircuitGraph (p. 421).

**9.104.3.4** virtual Node PC::PowerCircuit::cloneNode (Node *n*) [private, pure virtual]

Implemented in PC::PowerCircuitCompMatrix (p. 408), and PC::PowerCircuitGraph (p. 422).

**9.104.3.5** virtual int PC::PowerCircuit::compareMarkings (const Marking & *m1*, const Marking & *m2*) [private, pure virtual]

Implemented in PC::PowerCircuitCompMatrix (p. 409), and PC::PowerCircuitGraph (p. 422).

#### 9.104.3.6 **virtual void PC::PowerCircuit::connect (const Marking & *m*, const Marking & *p*) [pure virtual]**

Adds the marking *p* to all successor markings of nodes in the support of *m*. (So, if  $\text{supp}(m)$  and  $\text{supp}(p)$  are disjoint, that means, that arrows are drawn from all nodes in  $\text{supp}(m)$  to *p*). The user has to assure that, for each node, the sum lies in the range  $[-1, +1]$  (for example using **clone()** (p. 397))! The user is required to check that the resulting graph has no cycles and no successor marking becomes negative. All nodes in  $\text{supp}(m)$  have to be non-reduced (use **clone()** (p. 397) to assure that).

Implemented in **PC::PowerCircuitCompMatrix** (p. 409), and **PC::PowerCircuitGraph** (p. 422).

#### 9.104.3.7 **virtual void PC::PowerCircuit::connectInv (const Marking & *m*, const Marking & *p*) [pure virtual]**

The same as **connect**, only that the marking *p* is subtracted instead of added.

Implemented in **PC::PowerCircuitCompMatrix** (p. 409), and **PC::PowerCircuitGraph** (p. 422).

#### 9.104.3.8 **virtual Marking PC::PowerCircuit::copyMarking (const Marking & *m*) [private, pure virtual]**

Implemented in **PC::PowerCircuitCompMatrix** (p. 409), and **PC::PowerCircuitGraph** (p. 423).

#### 9.104.3.9 **virtual Marking PC::PowerCircuit::createMarking (const std::list< Node > & *nodeList*, const std::list< int > & *signList*) [pure virtual]**

Creates a new marking, with the nodes in *nodeList* set to the respective sign in *signList*.

Implemented in **PC::PowerCircuitCompMatrix** (p. 409), and **PC::PowerCircuitGraph** (p. 423).

#### 9.104.3.10 **virtual Marking PC::PowerCircuit::createMarking (const std::list< Node > & *nodes*) [pure virtual]**

Creates a new marking, with the nodes in *nodeList* set to 1.

Implemented in **PC::PowerCircuitCompMatrix** (p. 410), and **PC::PowerCircuitGraph** (p. 423).

### 9.104.3.11 virtual Marking PC::PowerCircuit::createMarking (int *i* = 0) [pure virtual]

Creates a new marking with value *val*. The therefore required nodes are inserted into the reduced part of the PC (p. 80).

Implemented in **PC::PowerCircuitCompMatrix** (p. 410), and **PC::PowerCircuitGraph** (p. 423).

### 9.104.3.12 virtual Marking PC::PowerCircuit::createMarkingFromNodes (unsigned int *numNodes*, ...) [virtual]

Creates a new marking, with the nodes set to 1. As first argument it requires the number of nodes set to one. The other arguments are the respective nodes.

### 9.104.3.13 virtual Node PC::PowerCircuit::createNode (const Marking & *succ*) [pure virtual]

Creates a new node with a copy of *m* as successor-marking.

Implemented in **PC::PowerCircuitCompMatrix** (p. 410), and **PC::PowerCircuitGraph** (p. 423).

### 9.104.3.14 virtual void PC::PowerCircuit::decMarkingRefCount (Marking & *mark*) [private, pure virtual]

Implemented in **PC::PowerCircuitCompMatrix** (p. 410), and **PC::PowerCircuitGraph** (p. 423).

### 9.104.3.15 virtual void PC::PowerCircuit::draw (std::string *filename*, Marking *highlight1* = Marking (), Marking *highlight2* = Marking (), Marking *highlight3* = Marking (), Marking *highlight4* = Marking (), Marking *highlight5* = Marking (), Marking *highlight6* = Marking (), Marking *highlight7* = Marking (), Marking *highlight8* = Marking (), Marking *highlight9* = Marking ()) [virtual]

### 9.104.3.16 virtual Node PC::PowerCircuit::getIncidentNode (const Marking & *m*) [private, pure virtual]

Implemented in **PC::PowerCircuitCompMatrix** (p. 411), and **PC::PowerCircuitGraph** (p. 424).

**9.104.3.17** `virtual std::list<Node> PC::PowerCircuit::getMarkingNodes (const Marking & m) [pure virtual]`

Returns a list of all nodes which are non-zero in the marking *m*.

Implemented in **PC::PowerCircuitCompMatrix** (p. 411), and **PC::PowerCircuitGraph** (p. 424).

**9.104.3.18** `virtual std::list<Marking> PC::PowerCircuit::getMarkings () [pure virtual]`

Returns a list of all markings in the **PowerCircuit** (p. 395).

Implemented in **PC::PowerCircuitCompMatrix** (p. 411), and **PC::PowerCircuitGraph** (p. 424).

**9.104.3.19** `virtual std::list<Node> PC::PowerCircuit::getNodes () [pure virtual]`

Returns a list of all nodes in the **PowerCircuit** (p. 395).

Implemented in **PC::PowerCircuitCompMatrix** (p. 411), and **PC::PowerCircuitGraph** (p. 424).

**9.104.3.20** `virtual int PC::PowerCircuit::getNodeSignInMarking (Node n, const Marking & m) const [private, pure virtual]`

Implemented in **PC::PowerCircuitCompMatrix** (p. 412), and **PC::PowerCircuitGraph** (p. 425).

**9.104.3.21** `int PC::PowerCircuit::getNumEdges ()`

**9.104.3.22** `int PC::PowerCircuit::getNumMarkings ()`

**9.104.3.23** `int PC::PowerCircuit::getNumNodes ()`

**9.104.3.24** `virtual int PC::PowerCircuit::getRedNodeOrd (Node n) [private, pure virtual]`

Implemented in **PC::PowerCircuitCompMatrix** (p. 412), and **PC::PowerCircuitGraph** (p. 425).

**9.104.3.25** virtual Node PC::PowerCircuit::getReducedNode (unsigned int *ord*)  
[pure virtual]

Returns the node with index *ord* (that means there are *ord* smaller nodes in the reduced part of the PC (p. 80)) If there are less than *ord* nodes, the undefined node is returned.

Implemented in PC::PowerCircuitCompMatrix (p. 412), and PC::PowerCircuitGraph (p. 425).

**9.104.3.26** virtual Node PC::PowerCircuit::getSmallestNodeInMarking (const Marking & *m*) [private, pure virtual]

Implemented in PC::PowerCircuitCompMatrix (p. 412), and PC::PowerCircuitGraph (p. 425).

**9.104.3.27** virtual Marking PC::PowerCircuit::getSuccMarking (Node *n*)  
[private, pure virtual]

Implemented in PC::PowerCircuitCompMatrix (p. 412), and PC::PowerCircuitGraph (p. 425).

**9.104.3.28** virtual Marking PC::PowerCircuit::incMarking (const Marking & *m*) [private, pure virtual]

Implemented in PC::PowerCircuitCompMatrix (p. 412), and PC::PowerCircuitGraph (p. 425).

**9.104.3.29** virtual void PC::PowerCircuit::incMarkingRefCount (const Marking & *mark*) [private, pure virtual]

Implemented in PC::PowerCircuitCompMatrix (p. 412), and PC::PowerCircuitGraph (p. 425).

**9.104.3.30** virtual Marking PC::PowerCircuit::intersectMarkings (const Marking & *m1*, const Marking & *m2*) [private, pure virtual]

Implemented in PC::PowerCircuitCompMatrix (p. 413), and PC::PowerCircuitGraph (p. 426).

**9.104.3.31** `virtual Marking PC::PowerCircuit::invMarking (const Marking & m) [private, pure virtual]`

Implemented in **PC::PowerCircuitCompMatrix** (p. 413), and **PC::PowerCircuitGraph** (p. 426).

**9.104.3.32** `virtual bool PC::PowerCircuit::isMarkingReduced (const Marking & m) const [private, pure virtual]`

Implemented in **PC::PowerCircuitCompMatrix** (p. 413), and **PC::PowerCircuitGraph** (p. 426).

**9.104.3.33** `virtual bool PC::PowerCircuit::isSuccessorMarking (const Marking & m) const [private, pure virtual]`

Implemented in **PC::PowerCircuitCompMatrix** (p. 413), and **PC::PowerCircuitGraph** (p. 426).

**9.104.3.34** `virtual void PC::PowerCircuit::print (std::ostream & os = std::cout) [pure virtual]`

Prints the adjacency matrix of the underlying graph.

Implemented in **PC::PowerCircuitCompMatrix** (p. 414), and **PC::PowerCircuitGraph** (p. 427).

**9.104.3.35** `virtual void PC::PowerCircuit::printStatistics (std::ostream & os = std::cout) [virtual]`

Prints some statistical data.

Reimplemented in **PC::PowerCircuitCompMatrix** (p. 414), and **PC::PowerCircuitGraph** (p. 428).

**9.104.3.36** `virtual void PC::PowerCircuit::reduce (std::vector< Node > & nodeVector, std::vector< Marking > & markingVector) [pure virtual]`

Moves all nodes in nodeList and markings in markingList to the reduced part of the **PC** (p. 80). Nodes in nodeList MUST NOT be reduced! The support of markings in markingList has to be in nodeList or the already reduced part of the **PC** (p. 80).

Implemented in **PC::PowerCircuitCompMatrix** (p. 415), and **PC::PowerCircuitGraph** (p. 428).



**9.104.3.37 virtual void PC::PowerCircuit::reduce () [pure virtual]**

Reduces the whole **PC** (p. 80) and all its markings.

Implemented in **PC::PowerCircuitCompMatrix** (p. 415), and **PC::PowerCircuitGraph** (p. 428).

**9.104.3.38 virtual void PC::PowerCircuit::remove (const Marking & *m*) [pure virtual]**

Deletes all nodes in the given marking. (So if other markings use those nodes, their values may be changed!)

Implemented in **PC::PowerCircuitCompMatrix** (p. 415), and **PC::PowerCircuitGraph** (p. 428).

**9.104.4 Friends And Related Function Documentation****9.104.4.1 friend class Marking [friend]**

Definition at line 196 of file PowerCircuit.h.

**9.104.4.2 friend class Node [friend]**

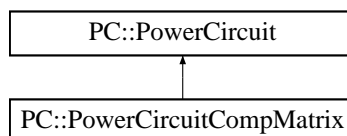
Definition at line 195 of file PowerCircuit.h.

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

- HigmanGroup/include/**PowerCircuit.h**

## 9.105 PC::PowerCircuitCompMatrix Class Reference

#include <PowerCircuitCompMatrix.h> Inheritance diagram for PC::PowerCircuitCompMatrix::



### Classes

- struct **ColHdr**
- struct **IntMarking**
- struct **IntNode**
- struct **RowHdr**

### Public Member Functions

- **PowerCircuitCompMatrix** (int numInitNodes=0, int numInitCols=0)
- virtual **~PowerCircuitCompMatrix** ()
- virtual void **reduce** ()
- virtual void **reduce** (std::vector< **Node** > &nodeVector, std::vector< **Marking** > &markingVector)
- virtual **Marking** **createMarking** (int i=0)
- virtual **Marking** **createMarking** (const std::list< **Node** > &nodes)
- virtual **Marking** **createMarking** (const std::list< **Node** > &nodeList, const std::list< int > &signList)
- virtual **Node** **createNode** (const **Marking** &succ)
- virtual void **connect** (const **Marking** &m, const **Marking** &p)
- virtual void **connectInv** (const **Marking** &m, const **Marking** &q)
- virtual void **remove** (const **Marking** &m)
- virtual **Node** **getReducedNode** (unsigned int ord)
- virtual std::list< **Node** > **getNodes** ()
- virtual std::list< **Marking** > **getMarkings** ()
- virtual std::list< **Node** > **getMarkingNodes** (const **Marking** &m)
- virtual **PowerCircuit** \* **clone** (std::vector< **Marking** > &markingsToKeep)
- virtual void **print** (std::ostream &os=std::cout)
- double **getMatrixUsage** ()
- virtual void **printStatistics** (std::ostream &os=std::cout)

## Private Types

- enum **MarkingType** { **NORMAL**, **LAMBDA** }
- typedef std::vector< **IntMarking** > **MarkingVector**
- typedef std::vector< **IntNode** > **NodeVector**

## Private Member Functions

- bool **checkMarkingValid** (const **Marking** &m) const
- bool **checkNodeValid** (**Node** n) const
- void **deleteNode** (**Row** node)
- void **deleteCol** (**Col** col)
- void **deleteMarking** (**Marking** &mark)
- **Marking** **allocateNewMarking** (**Col** col)
- **Row** **newNodeFromMarking** (const **Marking** &mark)
- **Row** **newOneNode** ()
- **Row** **cloneNode** (**Row** node)
- **Marking** **newOneMarking** ()
- **Marking** **newZeroMarking** ()
- **Marking** **newUnitMarking** (int onePos)
- **Marking** **newCopyColMarking** (const **Marking** &mark)
- void **separateMarkingFromCol** (const **Marking** &m)
- void **mergeCols** (**Col** targetCol, **Col** col2)
- int **getTreedNodePosToGivenCol** (**Col** col)
- void **moveNodeIntoTreedPartOfMatrix** (**Row** node, int newPos)
- int **findNewPosOfCompactCol** (**Col** col, bool &equal)
- void **compactifyFromBottom** (**Col** col)
- void **insertNewPowerOfTwoNode** (unsigned int power)
- unsigned int **calculateCompactRepresentation** (int n, **Sign** result[32])
- void **setBV** (**Row** node)
- void **newDoubleNode** (**Row** node)
- void **compactify** (**Col** col)
- void **removeDoubleNodesFromMarkings** (**Row** oldNode, **Row** newNode)
- int **insertCompactMarkingIntoTreed** (const **Marking** &mark)
- int **insertCompactColIntoTreed** (**Col** col)
- int **insertNodeIntoTreed** (**Row** node)
- void **moveColsToTreed** (std::list< **Col** > &colList)
- void **topSortNode** (unsigned int nodeIndex, std::vector< **Row** > &nodesToSort, std::vector< bool > &visited, std::vector< **Row** > &topSortPerm)
- void **extendTree** (std::vector< **Row** > &nodeList, std::vector< **Col** > &colList)
- void **checkCyclesRecursive** (**Row** node, std::vector< bool > visited)
- void **checkCycles** ()

- virtual void **incMarkingRefCount** (const **Marking** &mark)
- virtual void **decMarkingRefCount** (**Marking** &mark)
- virtual **Marking** **addMarkings** (const **Marking** &m1, const **Marking** &m2)
- virtual **Marking** **invMarking** (const **Marking** &m)
- virtual **Marking** **incMarking** (const **Marking** &mark)
- virtual **Marking** **intersectMarkings** (const **Marking** &m1, const **Marking** &m2)
- virtual **Marking** **cloneMarking** (const **Marking** &mark)
- virtual **Node** **cloneNode** (**Node** n)
- virtual **Marking** **copyMarking** (const **Marking** &m)
- virtual bool **isMarkingReduced** (const **Marking** &m) const
- virtual int **compareMarkings** (const **Marking** &m1, const **Marking** &m2)
- virtual int **getRedNodeOrd** (**Node** n)
- virtual int **getNodeSignInMarking** (**Node** n, const **Marking** &m) const
- virtual bool **isSuccessorMarking** (const **Marking** &m) const
- virtual **Node** **getIncidentNode** (const **Marking** &m)
- virtual **Node** **getSmallestNodeInMarking** (const **Marking** &m)
- virtual **Marking** **getSuccMarking** (**Node** n)

### Private Attributes

- **SignMatrix**< **RowHdr**, **ColHdr**, unsigned int > **matrix**
- unsigned int **numTreedCols**
- unsigned int **numTreedNodes**

### Static Private Attributes

- static **MarkingVector** **markings**
- static **NodeVector** **nodes**
- static unsigned int **totalNumMarkings**
- static unsigned int **totalNumNodes**
- static int **firstDeletedMarking**
- static int **firstDeletedNode**

## 9.105.1 Detailed Description

Definition at line 14 of file PowerCircuitCompMatrix.h.

## 9.105.2 Member Typedef Documentation

**9.105.2.1** `typedef std::vector<IntMarking>  
PC::PowerCircuitCompMatrix::MarkingVector  
[private]`

Definition at line 65 of file PowerCircuitCompMatrix.h.

**9.105.2.2** `typedef std::vector<IntNode>  
PC::PowerCircuitCompMatrix::NodeVector  
[private]`

Definition at line 66 of file PowerCircuitCompMatrix.h.

## 9.105.3 Member Enumeration Documentation

**9.105.3.1** `enum PC::PowerCircuitCompMatrix::MarkingType [private]`

Enumerator:

*NORMAL*

*LAMBDA*

Definition at line 42 of file PowerCircuitCompMatrix.h.

## 9.105.4 Constructor & Destructor Documentation

**9.105.4.1** `PC::PowerCircuitCompMatrix::PowerCircuitCompMatrix (int  
numInitNodes = 0, int numInitCols = 0)`

**9.105.4.2** `virtual PC::PowerCircuitCompMatrix::~~PowerCircuitCompMatrix  
() [virtual]`

## 9.105.5 Member Function Documentation

**9.105.5.1** `virtual Marking PC::PowerCircuitCompMatrix::addMarkings (const  
Marking & m1, const Marking & m2) [private, virtual]`

Implements **PC::PowerCircuit** (p. 397).

- 9.105.5.2 **Marking** **PC::PowerCircuitCompMatrix::allocateNewMarking** (*Col col*) [**private**]
- 9.105.5.3 **unsigned int**  
**PC::PowerCircuitCompMatrix::calculateCompactRepresentation** (*int n*, *Sign result[32]*) [**private**]
- 9.105.5.4 **void** **PC::PowerCircuitCompMatrix::checkCycles** () [**private**]
- 9.105.5.5 **void** **PC::PowerCircuitCompMatrix::checkCyclesRecursive** (*Row node*, *std::vector< bool > visited*) [**private**]
- 9.105.5.6 **bool** **PC::PowerCircuitCompMatrix::checkMarkingValid** (*const Marking & m*) **const** [**private**]
- 9.105.5.7 **bool** **PC::PowerCircuitCompMatrix::checkNodeValid** (*Node n*) **const** [**private**]
- 9.105.5.8 **virtual PowerCircuit\*** **PC::PowerCircuitCompMatrix::clone** (*std::vector< Marking > & markingsToClone*) [**virtual**]

Makes a copy of a **PowerCircuit** (p. 395). Only the markings in the markingsToClone list are available in the cloned circuit. Reduced nodes and markings are remain reduced. The clone has to be deallocated via delete, once it is not needed anymore!

Implements **PC::PowerCircuit** (p. 397).

- 9.105.5.9 **virtual Marking** **PC::PowerCircuitCompMatrix::cloneMarking** (*const Marking & mark*) [**private**, **virtual**]

Implements **PC::PowerCircuit** (p. 397).

- 9.105.5.10 **virtual Node** **PC::PowerCircuitCompMatrix::cloneNode** (*Node n*) [**private**, **virtual**]

Implements **PC::PowerCircuit** (p. 397).

**9.105.5.11** Row PC::PowerCircuitCompMatrix::cloneNode (Row *node*)  
[private]

**9.105.5.12** void PC::PowerCircuitCompMatrix::compactify (Col *col*)  
[private]

**9.105.5.13** void PC::PowerCircuitCompMatrix::compactifyFromBottom (Col  
*col*) [private]

**9.105.5.14** virtual int PC::PowerCircuitCompMatrix::compareMarkings (const  
Marking & *m1*, const Marking & *m2*) [private, virtual]

Implements PC::PowerCircuit (p. 397).

**9.105.5.15** virtual void PC::PowerCircuitCompMatrix::connect (const  
Marking & *m*, const Marking & *p*) [virtual]

Adds the marking *p* to all successor markings of nodes in the support of *m*. (So, if  $\text{supp}(m)$  and  $\text{supp}(p)$  are disjoint, that means, that arrows are drawn from all nodes in  $\text{supp}(m)$  to *p*). The user has to assure that, for each node, the sum lies in the range  $[-1, +1]$  (for example using **clone()** (p. 408))! The user is required to check that the resulting graph has no cycles and no successor marking becomes negative. All nodes in  $\text{supp}(m)$  have to be non-reduced (use **clone()** (p. 408) to assure that).

Implements PC::PowerCircuit (p. 398).

**9.105.5.16** virtual void PC::PowerCircuitCompMatrix::connectInv (const  
Marking & *m*, const Marking & *p*) [virtual]

The same as connect, only that the marking *p* is subtracted instead of added.

Implements PC::PowerCircuit (p. 398).

**9.105.5.17** virtual Marking PC::PowerCircuitCompMatrix::copyMarking  
(const Marking & *m*) [private, virtual]

Implements PC::PowerCircuit (p. 398).

**9.105.5.18** virtual Marking PC::PowerCircuitCompMatrix::createMarking  
(const std::list< Node > & *nodeList*, const std::list< int > &  
*signList*) [virtual]

Creates a new marking, with the nodes in *nodeList* set to the respective sign in *signList*.

Implements **PC::PowerCircuit** (p. 398).

**9.105.5.19 virtual Marking PC::PowerCircuitCompMatrix::createMarking  
(const std::list< Node > & nodes) [virtual]**

Creates a new marking, with the nodes in nodeList set to 1.

Implements **PC::PowerCircuit** (p. 398).

**9.105.5.20 virtual Marking PC::PowerCircuitCompMatrix::createMarking  
(int i = 0) [virtual]**

Creates a new marking with value val. The therefore required nodes are inserted into the reduced part of the **PC** (p. 80).

Implements **PC::PowerCircuit** (p. 399).

**9.105.5.21 virtual Node PC::PowerCircuitCompMatrix::createNode (const  
Marking & succ) [virtual]**

Creates a new node with a copy of m as successor-marking.

Implements **PC::PowerCircuit** (p. 399).

**9.105.5.22 virtual void PC::PowerCircuitCompMatrix::decMarkingRefCount  
(Marking & mark) [private, virtual]**

Implements **PC::PowerCircuit** (p. 399).



- 9.105.5.23 void PC::PowerCircuitCompMatrix::deleteCol (Col *col*)  
[private]
- 9.105.5.24 void PC::PowerCircuitCompMatrix::deleteMarking (Marking & *mark*) [private]
- 9.105.5.25 void PC::PowerCircuitCompMatrix::deleteNode (Row *node*)  
[private]
- 9.105.5.26 void PC::PowerCircuitCompMatrix::extendTree (std::vector< Row  
> & *nodeList*, std::vector< Col > & *colList*) [private]
- 9.105.5.27 int PC::PowerCircuitCompMatrix::findNewPosOfCompactCol (Col  
*col*, bool & *equal*) [private]
- 9.105.5.28 virtual Node PC::PowerCircuitCompMatrix::getIncidentNode  
(const Marking & *m*) [private, virtual]

Implements **PC::PowerCircuit** (p. 399).

- 9.105.5.29 virtual std::list<Node>  
PC::PowerCircuitCompMatrix::getMarkingNodes  
(const Marking & *m*) [virtual]

Returns a list of all nodes which are non-zero in the marking *m*.

Implements **PC::PowerCircuit** (p. 400).

- 9.105.5.30 virtual std::list<Marking>  
PC::PowerCircuitCompMatrix::getMarkings ()  
[virtual]

Returns a list of all markings in the **PowerCircuit** (p. 395).

Implements **PC::PowerCircuit** (p. 400).

- 9.105.5.31 double PC::PowerCircuitCompMatrix::getMatrixUsage ()
- 9.105.5.32 virtual std::list<Node> PC::PowerCircuitCompMatrix::getNodes ()  
[virtual]

Returns a list of all nodes in the **PowerCircuit** (p. 395).

Implements **PC::PowerCircuit** (p. 400).

**9.105.5.33** `virtual int PC::PowerCircuitCompMatrix::getNodeSignInMarking (Node n, const Marking & m) const` [`private`, `virtual`]

Implements **PC::PowerCircuit** (p. 400).

**9.105.5.34** `virtual int PC::PowerCircuitCompMatrix::getRedNodeOrd (Node n)` [`private`, `virtual`]

Implements **PC::PowerCircuit** (p. 400).

**9.105.5.35** `virtual Node PC::PowerCircuitCompMatrix::getReducedNode (unsigned int ord)` [`virtual`]

Returns the node with index *ord* (that means there are *ord* smaller nodes in the reduced part of the **PC** (p. 80)) If there are less than *ord* nodes, the undefined node is returned.

Implements **PC::PowerCircuit** (p. 401).

**9.105.5.36** `virtual Node PC::PowerCircuitCompMatrix::getSmallestNodeInMarking (const Marking & m)` [`private`, `virtual`]

Implements **PC::PowerCircuit** (p. 401).

**9.105.5.37** `virtual Marking PC::PowerCircuitCompMatrix::getSuccMarking (Node n)` [`private`, `virtual`]

Implements **PC::PowerCircuit** (p. 401).

**9.105.5.38** `int PC::PowerCircuitCompMatrix::getTreedNodePosToGivenCol (Col col)` [`private`]

**9.105.5.39** `virtual Marking PC::PowerCircuitCompMatrix::incMarking (const Marking & mark)` [`private`, `virtual`]

Implements **PC::PowerCircuit** (p. 401).

**9.105.5.40** `virtual void PC::PowerCircuitCompMatrix::incMarkingRefCount (const Marking & mark)` [`private`, `virtual`]

Implements **PC::PowerCircuit** (p. 401).

- 9.105.5.41** `int PC::PowerCircuitCompMatrix::insertCompactColIntoTreed (Col col) [private]`
- 9.105.5.42** `int PC::PowerCircuitCompMatrix::insertCompactMarkingIntoTreed (const Marking & mark) [private]`
- 9.105.5.43** `void PC::PowerCircuitCompMatrix::insertNewPowerOfTwoNode (unsigned int power) [private]`
- 9.105.5.44** `int PC::PowerCircuitCompMatrix::insertNodeIntoTreed (Row node) [private]`
- 9.105.5.45** `virtual Marking PC::PowerCircuitCompMatrix::intersectMarkings (const Marking & m1, const Marking & m2) [private, virtual]`

Implements **PC::PowerCircuit** (p. 401).

- 9.105.5.46** `virtual Marking PC::PowerCircuitCompMatrix::invMarking (const Marking & m) [private, virtual]`

Implements **PC::PowerCircuit** (p. 402).

- 9.105.5.47** `virtual bool PC::PowerCircuitCompMatrix::isMarkingReduced (const Marking & m) const [private, virtual]`

Implements **PC::PowerCircuit** (p. 402).

- 9.105.5.48** `virtual bool PC::PowerCircuitCompMatrix::isSuccessorMarking (const Marking & m) const [private, virtual]`

Implements **PC::PowerCircuit** (p. 402).

- 9.105.5.49 `void PC::PowerCircuitCompMatrix::mergeCols (Col targetCol, Col col2) [private]`
- 9.105.5.50 `void PC::PowerCircuitCompMatrix::moveColsToTreed (std::list< Col > & colList) [private]`
- 9.105.5.51 `void PC::PowerCircuitCompMatrix::moveNodeIntoTreedPartOfMatrix (Row node, int newPos) [private]`
- 9.105.5.52 `Marking PC::PowerCircuitCompMatrix::newCopyColMarking (const Marking & mark) [private]`
- 9.105.5.53 `void PC::PowerCircuitCompMatrix::newDoubleNode (Row node) [private]`
- 9.105.5.54 `Row PC::PowerCircuitCompMatrix::newNodeFromMarking (const Marking & mark) [private]`
- 9.105.5.55 `Marking PC::PowerCircuitCompMatrix::newOneMarking () [private]`
- 9.105.5.56 `Row PC::PowerCircuitCompMatrix::newOneNode () [private]`
- 9.105.5.57 `Marking PC::PowerCircuitCompMatrix::newUnitMarking (int onePos) [private]`
- 9.105.5.58 `Marking PC::PowerCircuitCompMatrix::newZeroMarking () [private]`
- 9.105.5.59 `virtual void PC::PowerCircuitCompMatrix::print (std::ostream & os = std::cout) [virtual]`

Prints the adjacency matrix of the underlying graph.

Implements **PC::PowerCircuit** (p. 402).

- 9.105.5.60 `virtual void PC::PowerCircuitCompMatrix::printStatistics (std::ostream & os = std::cout) [virtual]`

Prints some statistical data.

Reimplemented from **PC::PowerCircuit** (p. 402).

**9.105.5.61** `virtual void PC::PowerCircuitCompMatrix::reduce (std::vector< Node > & nodeVector, std::vector< Marking > & markingVector) [virtual]`

Moves all nodes in *nodeList* and markings in *markingList* to the reduced part of the **PC** (p. 80). Nodes in *nodeList* MUST NOT be reduced! The support of markings in *markingList* has to be in *nodeList* or the already reduced part of the **PC** (p. 80).

Implements **PC::PowerCircuit** (p. 402).

**9.105.5.62** `virtual void PC::PowerCircuitCompMatrix::reduce () [virtual]`

Reduces the whole **PC** (p. 80) and all its markings.

Implements **PC::PowerCircuit** (p. 403).

**9.105.5.63** `virtual void PC::PowerCircuitCompMatrix::remove (const Marking & m) [virtual]`

Deletes all nodes in the given marking. (So if other markings use those nodes, their values may be changed!)

Implements **PC::PowerCircuit** (p. 403).

**9.105.5.64** `void PC::PowerCircuitCompMatrix::removeDoubleNodesFromMarkings (Row oldNode, Row newNode) [private]`

**9.105.5.65** `void PC::PowerCircuitCompMatrix::separateMarkingFromCol (const Marking & m) [private]`

**9.105.5.66** `void PC::PowerCircuitCompMatrix::setBV (Row node) [private]`

**9.105.5.67** `void PC::PowerCircuitCompMatrix::topSortNode (unsigned int nodeIndex, std::vector< Row > & nodesToSort, std::vector< bool > & visited, std::vector< Row > & topSortPerm) [private]`

## 9.105.6 Member Data Documentation

**9.105.6.1** `int PC::PowerCircuitCompMatrix::firstDeletedMarking [static, private]`

Definition at line 72 of file *PowerCircuitCompMatrix.h*.

**9.105.6.2** `int PC::PowerCircuitCompMatrix::firstDeletedNode` `[static, private]`

Definition at line 73 of file PowerCircuitCompMatrix.h.

**9.105.6.3** `MarkingVector PC::PowerCircuitCompMatrix::markings` `[static, private]`

Definition at line 68 of file PowerCircuitCompMatrix.h.

**9.105.6.4** `SignMatrix<RowHdr,ColHdr,unsigned int>`  
`PC::PowerCircuitCompMatrix::matrix` `[private]`

Definition at line 35 of file PowerCircuitCompMatrix.h.

**9.105.6.5** `NodeVector PC::PowerCircuitCompMatrix::nodes` `[static, private]`

Definition at line 69 of file PowerCircuitCompMatrix.h.

**9.105.6.6** `unsigned int PC::PowerCircuitCompMatrix::numTreedCols` `[private]`

Definition at line 36 of file PowerCircuitCompMatrix.h.

**9.105.6.7** `unsigned int PC::PowerCircuitCompMatrix::numTreedNodes` `[private]`

Definition at line 37 of file PowerCircuitCompMatrix.h.

**9.105.6.8** `unsigned int PC::PowerCircuitCompMatrix::totalNumMarkings` `[static, private]`

Definition at line 70 of file PowerCircuitCompMatrix.h.

**9.105.6.9** `unsigned int PC::PowerCircuitCompMatrix::totalNumNodes` `[static, private]`

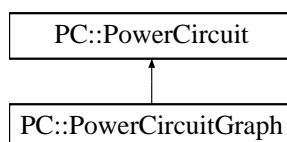
Definition at line 71 of file PowerCircuitCompMatrix.h.

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

- HigmanGroup/include/**PowerCircuitCompMatrix.h**

## 9.106 PC::PowerCircuitGraph Class Reference

#include <PowerCircuitGraph.h> Inheritance diagram for PC::PowerCircuitGraph::



### Classes

- struct **IntMarking**
- struct **IntNode**
- struct **NodeUsedByType**

### Public Member Functions

- **PowerCircuitGraph** ()
- virtual **~PowerCircuitGraph** ()
- virtual void **reduce** ()
- void **reduce** (std::list< **Marking** > markingList)
- virtual void **reduce** (std::vector< **Node** > &nodeVector, std::vector< **Marking** > &markingVector)
- virtual **Marking** **createMarking** (int i=0)
- virtual **Marking** **createMarking** (const std::list< **Node** > &nodes)
- virtual **Marking** **createMarking** (const std::list< **Node** > &nodeList, const std::list< int > &signList)
- virtual **Marking** **createMarking** (const std::list< std::pair< **Node**, **Sign** > > &nodeList)
- virtual **Node** **createNode** (const **Marking** &succ)
- virtual void **connect** (const **Marking** &m, const **Marking** &p)
- virtual void **connectInv** (const **Marking** &m, const **Marking** &q)
- virtual void **remove** (const **Marking** &m)
- virtual **Node** **getReducedNode** (unsigned int ord)
- virtual std::list< **Node** > **getNodes** ()
- virtual std::list< **Marking** > **getMarkings** ()
- virtual std::list< **Node** > **getMarkingNodes** (const **Marking** &m)
- virtual **PowerCircuit** \* **clone** (std::vector< **Marking** > &markingsToKeep)
- virtual void **print** (std::ostream &os=std::cout)



- void **printMarking** (const **Marking** &m, std::ostream &os=std::cout)
- void **printMarking** (const std::list< std::pair< **Node**, **Sign** > > &l, std::ostream &os=std::cout)
- double **getMatrixUsage** ()
- virtual void **printStatistics** (std::ostream &os=std::cout)
- bool **checkCyclesRecursive** (int n, std::vector< bool > &visited)
- bool **checkCycles** ()
- bool **checkConsistency** ()

## Private Member Functions

- bool **checkMarkingValid** (const **Marking** &m) const
- bool **checkNodeValid** (**Node** n) const
- void **deleteNode** (**Node** node)
- void **deleteMarking** (**Marking** &mark)
- **Marking** **newMarking** (std::list< std::pair< **Node**, **Sign** > > nodeList)
- **Node** **newNode** (std::list< std::pair< **Node**, **Sign** > > nodeList)
- **Node** **newOneNode** ()
- **Marking** **newOneMarking** ()
- **Marking** **newZeroMarking** ()
- **Marking** **newUnitMarking** (int onePos)
- **Marking** **newCopyMarking** (const **Marking** &mark)
- std::list< std::pair< **Node**, **Sign** > > **inc** (const std::list< std::pair< **Node**, **Sign** > > &nodeList)
- void **moveNodeIntoReducedPart** (**Node** node, int newPos)
- int **compare** (std::list< std::pair< **Node**, **Sign** > > &l1, std::list< std::pair< **Node**, **Sign** > > &l2)
- void **sortNodeList** (std::list< std::pair< **Node**, **Sign** > > &l)
- void **insertNewPowerOfTwoNode** (unsigned int power)
- std::list< std::pair< **Node**, **Sign** > > **calculateCompactRepresentation** (int n)
- void **markSucessors** (std::vector< bool > marked, std::list< std::pair< **Node**, **Sign** > > &nodeList)
- void **setBV** (**Node** node)
- **Node** **newDoubleNode** (**Node** node)
- std::list< std::pair< **Node**, **Sign** > >::iterator **findNodeInList** (unsigned int nodeIndex, std::list< std::pair< **Node**, **Sign** > > &nodeList)
- void **removeDoubleNodesFromMarkings** (**Node** oldNode, **Node** newNode, std::list< **NodeUsedByType** > \*nodeUsedBy)
- int **findNewPosOfReducedNode** (**Node** node, bool &equal)
- int **insertNodeIntoReduced** (**Node** node, std::list< **NodeUsedByType** > \*nodeUsedBy)
- void **topSortNode** (**Node** node, std::vector< bool > &visited, std::vector< **Node** > &topSortPerm)

- void **extendTree** (std::vector< **Node** > &nodeList, std::vector< **Marking** > &markingList)
- virtual void **incMarkingRefCount** (const **Marking** &mark)
- virtual void **decMarkingRefCount** (**Marking** &mark)
- virtual **Marking** **addMarkings** (const **Marking** &m1, const **Marking** &m2)
- virtual **Marking** **invMarking** (const **Marking** &m)
- virtual **Marking** **incMarking** (const **Marking** &mark)
- virtual **Marking** **intersectMarkings** (const **Marking** &m1, const **Marking** &m2)
- virtual **Marking** **cloneMarking** (const **Marking** &mark)
- virtual **Node** **cloneNode** (**Node** n)
- virtual **Marking** **copyMarking** (const **Marking** &m)
- virtual bool **isMarkingReduced** (const **Marking** &m) const
- virtual int **compareMarkings** (const **Marking** &m1, const **Marking** &m2)
- virtual int **getRedNodeOrd** (**Node** n)
- virtual int **getNodeSignInMarking** (**Node** n, const **Marking** &m) const
- virtual bool **isSuccessorMarking** (const **Marking** &m) const
- virtual **Node** **getIncidentNode** (const **Marking** &m)
- virtual **Node** **getSmallestNodeInMarking** (const **Marking** &m)
- virtual **Marking** **getSuccMarking** (**Node** n)

### Static Private Member Functions

- static bool **compareNodesLessThan** (const std::pair< **Node**, **Sign** > &n1, const std::pair< **Node**, **Sign** > &n2)

### Private Attributes

- unsigned int **numNodes**
- unsigned int **numMarkings**
- unsigned int **numReducedNodes**
- std::vector< **Node** > **nodeOrder**
- std::vector< **IntNode** > **nodes**
- std::vector< **IntMarking** > **markings**
- int **firstDeletedNode**
- int **firstDeletedMarking**

### 9.106.1 Detailed Description

Definition at line 15 of file PowerCircuitGraph.h.

## 9.106.2 Constructor & Destructor Documentation

**9.106.2.1** PC::PowerCircuitGraph::PowerCircuitGraph ()

**9.106.2.2** virtual PC::PowerCircuitGraph::~~PowerCircuitGraph ()  
[virtual]

## 9.106.3 Member Function Documentation

**9.106.3.1** virtual Marking PC::PowerCircuitGraph::addMarkings (const Marking & *m1*, const Marking & *m2*) [private, virtual]

Implements PC::PowerCircuit (p. 397).

**9.106.3.2** std::list< std::pair<Node,Sign> >  
PC::PowerCircuitGraph::calculateCompactRepresentation (int *n*)  
[private]

**9.106.3.3** bool PC::PowerCircuitGraph::checkConsistency ()

**9.106.3.4** bool PC::PowerCircuitGraph::checkCycles ()

**9.106.3.5** bool PC::PowerCircuitGraph::checkCyclesRecursive (int *n*,  
std::vector< bool > & *visited*)

**9.106.3.6** bool PC::PowerCircuitGraph::checkMarkingValid (const Marking &  
*m*) const [private]

**9.106.3.7** bool PC::PowerCircuitGraph::checkNodeValid (Node *n*) const  
[private]

**9.106.3.8** virtual PowerCircuit\* PC::PowerCircuitGraph::clone (std::vector<  
Marking > & *markingsToClone*) [virtual]

Makes a copy of a **PowerCircuit** (p. 395). Only the markings in the *markingsToClone* list are available in the cloned circuit. Reduced nodes and markings are remain reduced. The clone has to be deallocated via delete, once it is not needed anymore!

Implements PC::PowerCircuit (p. 397).

**9.106.3.9** virtual Marking PC::PowerCircuitGraph::cloneMarking (const Marking & *mark*) [private, virtual]

Implements PC::PowerCircuit (p. 397).

**9.106.3.10** `virtual Node PC::PowerCircuitGraph::cloneNode (Node n)`  
`[private, virtual]`

Implements **PC::PowerCircuit** (p. 397).

**9.106.3.11** `int PC::PowerCircuitGraph::compare (std::list< std::pair<`  
`Node, Sign > > & l1, std::list< std::pair< Node, Sign > > & l2)`  
`[private]`

**9.106.3.12** `virtual int PC::PowerCircuitGraph::compareMarkings (const`  
`Marking & m1, const Marking & m2)` `[private, virtual]`

Implements **PC::PowerCircuit** (p. 397).

**9.106.3.13** `static bool PC::PowerCircuitGraph::compareNodesLessThan (const`  
`std::pair< Node, Sign > & n1, const std::pair< Node, Sign > & n2)`  
`[inline, static, private]`

Definition at line 72 of file PowerCircuitGraph.h.

References nodes.

**9.106.3.14** `virtual void PC::PowerCircuitGraph::connect (const Marking & m,`  
`const Marking & p)` `[virtual]`

Adds the marking *p* to all successor markings of nodes in the support of *m*. (So, if *supp(m)* and *supp(p)* are disjoint, that means, that arrows are drawn from all nodes in *supp(m)* to *p*). The user has to assure that, for each node, the sum lies in the range [-1,+1] (for example using **clone()** (p. 421))! The user is required to check that the resulting graph has no cycles and no successor marking becomes negative. All nodes in *supp(m)* have to be non-reduced (use **clone()** (p. 421) to assure that).

Implements **PC::PowerCircuit** (p. 398).

**9.106.3.15** `virtual void PC::PowerCircuitGraph::connectInv (const Marking &`  
`m, const Marking & p)` `[virtual]`

The same as **connect**, only that the marking *p* is subtracted instead of added.

Implements **PC::PowerCircuit** (p. 398).

**9.106.3.16** virtual Marking PC::PowerCircuitGraph::copyMarking (const Marking & *m*) [**private**, **virtual**]

Implements **PC::PowerCircuit** (p. 398).

**9.106.3.17** virtual Marking PC::PowerCircuitGraph::createMarking (const std::list< std::pair< Node, Sign > > & *nodeList*) [**virtual**]

**9.106.3.18** virtual Marking PC::PowerCircuitGraph::createMarking (const std::list< Node > & *nodeList*, const std::list< int > & *signList*) [**virtual**]

Creates a new marking, with the nodes in *nodeList* set to the respective sign in *signList*.

Implements **PC::PowerCircuit** (p. 398).

**9.106.3.19** virtual Marking PC::PowerCircuitGraph::createMarking (const std::list< Node > & *nodes*) [**virtual**]

Creates a new marking, with the nodes in *nodeList* set to 1.

Implements **PC::PowerCircuit** (p. 398).

**9.106.3.20** virtual Marking PC::PowerCircuitGraph::createMarking (int *i* = 0) [**virtual**]

Creates a new marking with value *val*. The therefore required nodes are inserted into the reduced part of the **PC** (p. 80).

Implements **PC::PowerCircuit** (p. 399).

**9.106.3.21** virtual Node PC::PowerCircuitGraph::createNode (const Marking & *succ*) [**virtual**]

Creates a new node with a copy of *m* as successor-marking.

Implements **PC::PowerCircuit** (p. 399).

**9.106.3.22** virtual void PC::PowerCircuitGraph::decMarkingRefCount (Marking & *mark*) [**private**, **virtual**]

Implements **PC::PowerCircuit** (p. 399).

- 9.106.3.23 **void** **PC::PowerCircuitGraph::deleteMarking** (Marking & *mark*)  
[private]
- 9.106.3.24 **void** **PC::PowerCircuitGraph::deleteNode** (Node *node*)  
[private]
- 9.106.3.25 **void** **PC::PowerCircuitGraph::extendTree** (std::vector< Node > & *nodeList*, std::vector< Marking > & *markingList*) [private]
- 9.106.3.26 **int** **PC::PowerCircuitGraph::findNewPosOfReducedNode** (Node *node*, bool & *equal*) [private]
- 9.106.3.27 **std::list< std::pair<Node,Sign> >::iterator**  
**PC::PowerCircuitGraph::findNodeInList** (unsigned int *nodeIndex*,  
std::list< std::pair< Node, Sign > > & *nodeList*) [private]
- 9.106.3.28 **virtual Node** **PC::PowerCircuitGraph::getIncidentNode** (const  
Marking & *m*) [private, virtual]

Implements **PC::PowerCircuit** (p. 399).

- 9.106.3.29 **virtual std::list<Node>** **PC::PowerCircuitGraph::getMarkingNodes**  
(const Marking & *m*) [virtual]

Returns a list of all nodes which are non-zero in the marking *m*.

Implements **PC::PowerCircuit** (p. 400).

- 9.106.3.30 **virtual std::list<Marking>** **PC::PowerCircuitGraph::getMarkings**  
() [virtual]

Returns a list of all markings in the **PowerCircuit** (p. 395).

Implements **PC::PowerCircuit** (p. 400).

- 9.106.3.31 **double** **PC::PowerCircuitGraph::getMatrixUsage** ()

- 9.106.3.32 **virtual std::list<Node>** **PC::PowerCircuitGraph::getNodes** ()  
[virtual]

Returns a list of all nodes in the **PowerCircuit** (p. 395).

Implements **PC::PowerCircuit** (p. 400).

**9.106.3.33** `virtual int PC::PowerCircuitGraph::getNodeSignInMarking (Node n, const Marking & m) const [private, virtual]`

Implements **PC::PowerCircuit** (p. 400).

**9.106.3.34** `virtual int PC::PowerCircuitGraph::getRedNodeOrd (Node n) [private, virtual]`

Implements **PC::PowerCircuit** (p. 400).

**9.106.3.35** `virtual Node PC::PowerCircuitGraph::getReducedNode (unsigned int ord) [virtual]`

Returns the node with index *ord* (that means there are *ord* smaller nodes in the reduced part of the **PC** (p. 80)) If there are less than *ord* nodes, the undefined node is returned.

Implements **PC::PowerCircuit** (p. 401).

**9.106.3.36** `virtual Node PC::PowerCircuitGraph::getSmallestNodeInMarking (const Marking & m) [private, virtual]`

Implements **PC::PowerCircuit** (p. 401).

**9.106.3.37** `virtual Marking PC::PowerCircuitGraph::getSuccMarking (Node n) [private, virtual]`

Implements **PC::PowerCircuit** (p. 401).

**9.106.3.38** `std::list< std::pair<Node,Sign> > PC::PowerCircuitGraph::inc (const std::list< std::pair< Node, Sign > > & nodeList) [private]`

**9.106.3.39** `virtual Marking PC::PowerCircuitGraph::incMarking (const Marking & mark) [private, virtual]`

Implements **PC::PowerCircuit** (p. 401).

**9.106.3.40** `virtual void PC::PowerCircuitGraph::incMarkingRefCount (const Marking & mark) [private, virtual]`

Implements **PC::PowerCircuit** (p. 401).

- 9.106.3.41** void PC::PowerCircuitGraph::insertNewPowerOfTwoNode  
(unsigned int *power*) [private]
- 9.106.3.42** int PC::PowerCircuitGraph::insertNodeIntoReduced (Node *node*,  
std::list< NodeUsedByType > \* *nodeUsedBy*) [private]
- 9.106.3.43** virtual Marking PC::PowerCircuitGraph::intersectMarkings (const  
Marking & *m1*, const Marking & *m2*) [private, virtual]

Implements PC::PowerCircuit (p.401).

- 9.106.3.44** virtual Marking PC::PowerCircuitGraph::invMarking (const  
Marking & *m*) [private, virtual]

Implements PC::PowerCircuit (p.402).

- 9.106.3.45** virtual bool PC::PowerCircuitGraph::isMarkingReduced (const  
Marking & *m*) const [private, virtual]

Implements PC::PowerCircuit (p.402).

- 9.106.3.46** virtual bool PC::PowerCircuitGraph::isSuccessorMarking (const  
Marking & *m*) const [private, virtual]

Implements PC::PowerCircuit (p.402).



- 9.106.3.47 `void PC::PowerCircuitGraph::markSucessors (std::vector< bool > marked, std::list< std::pair< Node, Sign > > & nodeList)` `[private]`
- 9.106.3.48 `void PC::PowerCircuitGraph::moveNodeIntoReducedPart (Node node, int newPos)` `[private]`
- 9.106.3.49 `Marking PC::PowerCircuitGraph::newCopyMarking (const Marking & mark)` `[private]`
- 9.106.3.50 `Node PC::PowerCircuitGraph::newDoubleNode (Node node)` `[private]`
- 9.106.3.51 `Marking PC::PowerCircuitGraph::newMarking (std::list< std::pair< Node, Sign > > nodeList)` `[private]`
- 9.106.3.52 `Node PC::PowerCircuitGraph::newNode (std::list< std::pair< Node, Sign > > nodeList)` `[private]`
- 9.106.3.53 `Marking PC::PowerCircuitGraph::newOneMarking ()` `[private]`
- 9.106.3.54 `Node PC::PowerCircuitGraph::newOneNode ()` `[private]`
- 9.106.3.55 `Marking PC::PowerCircuitGraph::newUnitMarking (int onePos)` `[private]`
- 9.106.3.56 `Marking PC::PowerCircuitGraph::newZeroMarking ()` `[private]`
- 9.106.3.57 `virtual void PC::PowerCircuitGraph::print (std::ostream & os = std::cout)` `[virtual]`

Prints the adjacency matrix of the underlying graph.

Implements **PC::PowerCircuit** (p. 402).

- 9.106.3.58** `void PC::PowerCircuitGraph::printMarking (const std::list< std::pair< Node, Sign > > & l, std::ostream & os = std::cout)`
- 9.106.3.59** `void PC::PowerCircuitGraph::printMarking (const Marking & m, std::ostream & os = std::cout)`
- 9.106.3.60** `virtual void PC::PowerCircuitGraph::printStatistics (std::ostream & os = std::cout) [virtual]`

Prints some statistical data.

Reimplemented from **PC::PowerCircuit** (p. 402).

- 9.106.3.61** `virtual void PC::PowerCircuitGraph::reduce (std::vector< Node > & nodeVector, std::vector< Marking > & markingVector) [virtual]`

Moves all nodes in nodeList and markings in markingList to the reduced part of the **PC** (p. 80). Nodes in nodeList MUST NOT be reduced! The support of markings in markingList has to be in nodeList or the already reduced part of the **PC** (p. 80).

Implements **PC::PowerCircuit** (p. 402).

- 9.106.3.62** `void PC::PowerCircuitGraph::reduce (std::list< Marking > markingList)`

- 9.106.3.63** `virtual void PC::PowerCircuitGraph::reduce () [virtual]`

Reduces the whole **PC** (p. 80) and all its markings.

Implements **PC::PowerCircuit** (p. 403).

- 9.106.3.64** `virtual void PC::PowerCircuitGraph::remove (const Marking & m) [virtual]`

Deletes all nodes in the given marking. (So if other markings use those nodes, their values may be changed!)

Implements **PC::PowerCircuit** (p. 403).

- 9.106.3.65** void PC::PowerCircuitGraph::removeDoubleNodesFromMarkings (Node *oldNode*, Node *newNode*, std::list< NodeUsedByType > \* *nodeUsedBy*) [private]
- 9.106.3.66** void PC::PowerCircuitGraph::setBV (Node *node*) [private]
- 9.106.3.67** void PC::PowerCircuitGraph::sortNodeList (std::list< std::pair< Node, Sign > > & *l*) [private]
- 9.106.3.68** void PC::PowerCircuitGraph::topSortNode (Node *node*, std::vector< bool > & *visited*, std::vector< Node > & *topSortPerm*) [private]

## 9.106.4 Member Data Documentation

- 9.106.4.1** int PC::PowerCircuitGraph::firstDeletedMarking [private]

Definition at line 66 of file PowerCircuitGraph.h.

- 9.106.4.2** int PC::PowerCircuitGraph::firstDeletedNode [private]

Definition at line 65 of file PowerCircuitGraph.h.

- 9.106.4.3** std::vector<IntMarking> PC::PowerCircuitGraph::markings [private]

Definition at line 63 of file PowerCircuitGraph.h.

- 9.106.4.4** std::vector<Node> PC::PowerCircuitGraph::nodeOrder [private]

Definition at line 60 of file PowerCircuitGraph.h.

- 9.106.4.5** std::vector<IntNode> PC::PowerCircuitGraph::nodes [private]

Definition at line 62 of file PowerCircuitGraph.h.

Referenced by compareNodesLessThan().

**9.106.4.6 unsigned int PC::PowerCircuitGraph::numMarkings [private]**

Definition at line 58 of file PowerCircuitGraph.h.

**9.106.4.7 unsigned int PC::PowerCircuitGraph::numNodes [private]**

Definition at line 57 of file PowerCircuitGraph.h.

**9.106.4.8 unsigned int PC::PowerCircuitGraph::numReducedNodes  
[private]**

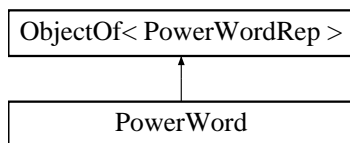
Definition at line 59 of file PowerCircuitGraph.h.

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

- HigmanGroup/include/**PowerCircuitGraph.h**

## 9.107 PowerWord Class Reference

#include <PowerWord.h> Inheritance diagram for PowerWord::



### Public Types

- typedef ConstPowerWordIterator **const\_iterator**
- typedef PowerWordIterator **iterator**
- typedef pair< int, int > **PII**

### Public Member Functions

- **PowerWord** ()
- **PowerWord** (const vector< int > &gens)
- **PowerWord** (const list< int > &gens)
- **PowerWord** (const vector< **PII** > &gens)
- **PowerWord** (const list< **PII** > &gens)
- **PowerWord** (int g, int p=1)
- bool **operator**< (const **PowerWord** &wr) const
- bool **operator**> (const **PowerWord** &wr) const
- bool **operator**== (const **PowerWord** &wr) const
- bool **operator**!= (const **PowerWord** &wr) const
- **PowerWord** & **operator**\*= (const **PowerWord** &w)
- **PowerWord** **operator**\* (const **PowerWord** &w) const
- **PowerWord** **operator**- () const
- **const\_iterator** **begin** () const
- **const\_iterator** **end** () const
- **iterator** **begin** ()
- **iterator** **end** ()
- **PowerWord** & **freelyReduce** ()
- **PowerWord** & **freelyReduce** (**const\_iterator** beg, **const\_iterator** end)
- const list< **PII** > & **getList** () const
- list< **PII** > & **getList** ()
- int **length** () const
- void **push\_back** (int gen, int power)

- void **push\_front** (int gen, int power)
- void **push\_back** (const pair< int, int > &g)
- void **push\_front** (const pair< int, int > &g)
- int **getPower** (**PowerWord** &base) const
- bool **doesContain** (const int &gen) const
- void **cyclicLeftShift** ()
- void **cyclicRightShift** ()
- **PowerWord** **cyclicallyReduce** () const
- void **cyclicallyReduceWord** ()
- **PowerWord** **cyclicallyReduce** (**PowerWord** &conjugator) const
- void **cyclicallyReduceWord** (**PowerWord** &conjugator)
- **PowerWord** **inverse** () const
- **PowerWord** **cyclicallyPermute** (int n) const
- **PowerWord** **initialSegment** (int len) const
- **PowerWord** **terminalSegment** (int len) const
- **PowerWord** **segment** (int from, int to) const
- int **exponentSum** (const int &gen) const
- int **isIn** (const int &gen) const
- **Word** **power** (int t) const
- void **insert** (const **PowerWord** &wr, int pos)

## Static Public Member Functions

- static **PowerWord** **randomWord** (int gens, int wLen)

## Private Member Functions

- ostream & **printOn** (ostream &os) const

## Friends

- ostream & **operator**<< (ostream &os, const **PowerWord** &w)

### 9.107.1 Detailed Description

Definition at line 27 of file PowerWord.h.

### 9.107.2 Member Typedef Documentation

#### 9.107.2.1 typedef ConstPowerWordIterator PowerWord::const\_iterator

Definition at line 32 of file PowerWord.h.

### 9.107.2.2 `typedef PowerWordIterator PowerWord::iterator`

Definition at line 33 of file PowerWord.h.

### 9.107.2.3 `typedef pair< int , int > PowerWord::PII`

Definition at line 43 of file PowerWord.h.

## 9.107.3 Constructor & Destructor Documentation

### 9.107.3.1 `PowerWord::PowerWord () [inline]`

Definition at line 45 of file PowerWord.h.

### 9.107.3.2 `PowerWord::PowerWord (const vector< int > & gens) [inline]`

Definition at line 46 of file PowerWord.h.

### 9.107.3.3 `PowerWord::PowerWord (const list< int > & gens) [inline]`

Definition at line 47 of file PowerWord.h.

### 9.107.3.4 `PowerWord::PowerWord (const vector< PII > & gens) [inline]`

Definition at line 48 of file PowerWord.h.

### 9.107.3.5 `PowerWord::PowerWord (const list< PII > & gens) [inline]`

Definition at line 49 of file PowerWord.h.

### 9.107.3.6 `PowerWord::PowerWord (int g, int p = 1) [inline]`

Definition at line 51 of file PowerWord.h.

## 9.107.4 Member Function Documentation

### 9.107.4.1 `iterator PowerWord::begin () [inline]`

Definition at line 107 of file PowerWord.h.

**9.107.4.2   `const_iterator PowerWord::begin () const   [inline]`**

Definition at line 105 of file PowerWord.h.

Referenced by `freelyReduce()`.

**9.107.4.3   `PowerWord PowerWord::cyclicallyPermute (int n) const   [inline]`**

Definition at line 173 of file PowerWord.h.

References `ObjectOf< Rep >::change()`.

**9.107.4.4   `PowerWord PowerWord::cyclicallyReduce (PowerWord & conjugator) const   [inline]`**

Definition at line 158 of file PowerWord.h.

References `ObjectOf< Rep >::change()`, and `cyclicallyReduce()`.

**9.107.4.5   `PowerWord PowerWord::cyclicallyReduce () const   [inline]`**

Definition at line 148 of file PowerWord.h.

References `ObjectOf< Rep >::change()`.

Referenced by `cyclicallyReduce()`, and `cyclicallyReduceWord()`.

**9.107.4.6   `void PowerWord::cyclicallyReduceWord (PowerWord & conjugator)  
[inline]`**

Definition at line 163 of file PowerWord.h.

References `ObjectOf< Rep >::change()`, `ObjectOf< PowerWordRep >::change()`, and `cyclicallyReduce()`.

**9.107.4.7   `void PowerWord::cyclicallyReduceWord ()   [inline]`**

Definition at line 155 of file PowerWord.h.

References `ObjectOf< PowerWordRep >::change()`, and `cyclicallyReduce()`.

**9.107.4.8   `void PowerWord::cyclicLeftShift ()   [inline]`**

Definition at line 143 of file PowerWord.h.



References `ObjectOf< PowerWordRep >::change()`, and `PowerWordRep::cyclicLeftShift()`.

#### 9.107.4.9 `void PowerWord::cyclicRightShift () [inline]`

Definition at line 145 of file `PowerWord.h`.

References `ObjectOf< PowerWordRep >::change()`, and `PowerWordRep::cyclicRightShift()`.

#### 9.107.4.10 `bool PowerWord::doesContain (const int & gen) const [inline]`

Definition at line 138 of file `PowerWord.h`.

References `PowerWordRep::doesContain()`, and `ObjectOf< PowerWordRep >::look()`.

#### 9.107.4.11 `iterator PowerWord::end () [inline]`

Definition at line 108 of file `PowerWord.h`.

#### 9.107.4.12 `const_iterator PowerWord::end () const [inline]`

Definition at line 106 of file `PowerWord.h`.

Referenced by `freelyReduce()`.

#### 9.107.4.13 `int PowerWord::exponentSum (const int & gen) const [inline]`

Definition at line 199 of file `PowerWord.h`.

References `PowerWordRep::exponentSum()`, and `ObjectOf< PowerWordRep >::look()`.

#### 9.107.4.14 `PowerWord& PowerWord::freelyReduce (const_iterator beg, const_iterator end)`

#### 9.107.4.15 `PowerWord& PowerWord::freelyReduce () [inline]`

Definition at line 110 of file `PowerWord.h`.

References `begin()`, `end()`, and `freelyReduce()`.

Referenced by `freelyReduce()`.

**9.107.4.16 list< PII >& PowerWord::getList () [inline]**

Definition at line 123 of file PowerWord.h.

References ObjectOf< PowerWordRep >::change(), and PowerWordRep::getList().

**9.107.4.17 const list< PII >& PowerWord::getList () const [inline]**

Definition at line 121 of file PowerWord.h.

References PowerWordRep::getList(), and ObjectOf< PowerWordRep >::look().

**9.107.4.18 int PowerWord::getPower (PowerWord & base) const [inline]**

Definition at line 134 of file PowerWord.h.

References ObjectOf< Rep >::change(), PowerWordRep::getPower(), and ObjectOf< PowerWordRep >::look().

**9.107.4.19 PowerWord PowerWord::initialSegment (int len) const [inline]**

Definition at line 181 of file PowerWord.h.

References ObjectOf< Rep >::change().

**9.107.4.20 void PowerWord::insert (const PowerWord & wr, int pos) [inline]**

Definition at line 209 of file PowerWord.h.

References ObjectOf< PowerWordRep >::change(), and ObjectOf< Rep >::look().

**9.107.4.21 PowerWord PowerWord::inverse () const [inline]**

Definition at line 167 of file PowerWord.h.

References ObjectOf< Rep >::change(), PowerWordRep::inverse(), and ObjectOf< PowerWordRep >::look().

**9.107.4.22 int PowerWord::isIn (const int & gen) const [inline]**

Definition at line 203 of file PowerWord.h.

References PowerWordRep::isIn(), and ObjectOf< PowerWordRep >::look().

**9.107.4.23 int PowerWord::length () const [inline]**

Definition at line 126 of file PowerWord.h.

References PowerWordRep::length(), and ObjectOf< PowerWordRep >::look().

**9.107.4.24 bool PowerWord::operator!= (const PowerWord & wr) const [inline]**

Definition at line 75 of file PowerWord.h.

References ObjectOf< Rep >::look(), and ObjectOf< PowerWordRep >::look().

**9.107.4.25 PowerWord PowerWord::operator\* (const PowerWord & w) const [inline]**

Definition at line 84 of file PowerWord.h.

**9.107.4.26 PowerWord& PowerWord::operator\*= (const PowerWord & w) [inline]**

Definition at line 79 of file PowerWord.h.

References ObjectOf< PowerWordRep >::change(), and ObjectOf< Rep >::look().

**9.107.4.27 PowerWord PowerWord::operator- () const [inline]**

Definition at line 90 of file PowerWord.h.

References ObjectOf< Rep >::change(), PowerWordRep::inverse(), and ObjectOf< PowerWordRep >::look().

**9.107.4.28 bool PowerWord::operator< (const PowerWord & wr) const [inline]**

Definition at line 65 of file PowerWord.h.

References ObjectOf< Rep >::look(), and ObjectOf< PowerWordRep >::look().

**9.107.4.29 bool PowerWord::operator== (const PowerWord & wr) const [inline]**

Definition at line 72 of file PowerWord.h.

References ObjectOf< Rep >::look(), and ObjectOf< PowerWordRep >::look().

**9.107.4.30** `bool PowerWord::operator> (const PowerWord & wr) const`  
`[inline]`

Definition at line 68 of file PowerWord.h.

References `ObjectOf< Rep >::look()`, and `ObjectOf< PowerWordRep >::look()`.

**9.107.4.31** `Word PowerWord::power (int t) const`

**9.107.4.32** `ostream& PowerWord::printOn (ostream & os) const` `[inline, private]`

Definition at line 233 of file PowerWord.h.

References `ObjectOf< PowerWordRep >::look()`, and `PowerWordRep::printOn()`.

**9.107.4.33** `void PowerWord::push_back (const pair< int, int > & g)`  
`[inline]`

Definition at line 131 of file PowerWord.h.

References `ObjectOf< PowerWordRep >::change()`, and `PowerWordRep::pushGeneratorBack()`.

**9.107.4.34** `void PowerWord::push_back (int gen, int power)` `[inline]`

Definition at line 129 of file PowerWord.h.

References `ObjectOf< PowerWordRep >::change()`, and `PowerWordRep::pushGeneratorBack()`.

**9.107.4.35** `void PowerWord::push_front (const pair< int, int > & g)`  
`[inline]`

Definition at line 132 of file PowerWord.h.

References `ObjectOf< PowerWordRep >::change()`, and `PowerWordRep::pushGeneratorFront()`.

**9.107.4.36** `void PowerWord::push_front (int gen, int power)` `[inline]`

Definition at line 130 of file PowerWord.h.

References `ObjectOf< PowerWordRep >::change()`, and `PowerWordRep::pushGeneratorFront()`.

**9.107.4.37** static PowerWord PowerWord::randomWord (int *gens*, int *wLen*)  
[static]

**9.107.4.38** PowerWord PowerWord::segment (int *from*, int *to*) const  
[inline]

Definition at line 193 of file PowerWord.h.

References ObjectOf< Rep >::change().

**9.107.4.39** PowerWord PowerWord::terminalSegment (int *len*) const  
[inline]

Definition at line 187 of file PowerWord.h.

References ObjectOf< Rep >::change().

## 9.107.5 Friends And Related Function Documentation

**9.107.5.1** ostream& operator<< (ostream & *os*, const PowerWord & *w*)  
[friend]

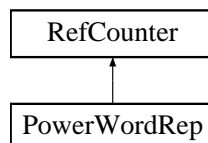
Definition at line 229 of file PowerWord.h.

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

- Elt/include/PowerWord.h

## 9.108 PowerWordRep Class Reference

#include <PowerWordRep.h> Inheritance diagram for PowerWordRep::



### Public Member Functions

- **PowerWordRep \* clone** () const
- const list< **PII** > & **getList** () const
- list< **PII** > & **getList** ()
- ostream & **printOn** (ostream &os) const

### Private Types

- typedef pair< int, int > **PII**

### Private Member Functions

- **PowerWordRep** ()
- **PowerWordRep** (const list< **PII** > &gens)
- **PowerWordRep** (const vector< **PII** > &gens)
- **PowerWordRep** (const **PowerWordRep** &wr)
- **PowerWordRep** (const list< int > &gens)
- **PowerWordRep** (const vector< int > &gens)
- **PowerWordRep** (int g, int p=1)
- **PowerWordRep operator=** (const **PowerWordRep** wr)
- **PowerWordRep & operator\*=** (const **PowerWordRep** &w)
- bool **operator<** (const **PowerWordRep** &wr) const
- bool **operator>** (const **PowerWordRep** &wr) const
- bool **operator==** (const **PowerWordRep** &wr) const
- **PowerWordRep operator\*** (const **PowerWordRep** &w) const
- bool **doesContain** (int gen) const
- int **length** () const
- int **exponentSum** (int gen) const
- int **isIn** (int gen) const
- int **getPower** (**PowerWordRep** &base) const

- **PowerWordRep** `inverse ()` `const`
- `void` **cyclicallyReduce** `()`
- `void` **cyclicallyReduce** (**PowerWordRep** &conjugator)
- `void` **cyclicLeftShift** `()`
- `void` **cyclicRightShift** `()`
- `void` **cyclicallyPermute** (int n)
- `void` **segment** (int from, int to)
- `void` **initialSegment** (int to)
- `void` **terminalSegment** (int from)
- `void` **insert** (`const` **PowerWordRep** &wr, int pos)
- `void` **pushGeneratorBack** (int g, int p)
- `void` **pushGeneratorFront** (int g, int p)

### Private Attributes

- `list< PII >` **theElements**
- `int` **theLength**

### Friends

- `class` **PowerWord**

## 9.108.1 Detailed Description

Definition at line 28 of file PowerWordRep.h.

## 9.108.2 Member Typedef Documentation

### 9.108.2.1 `typedef pair< int , int > PowerWordRep::PII` `[private]`

Definition at line 31 of file PowerWordRep.h.

## 9.108.3 Constructor & Destructor Documentation

### 9.108.3.1 `PowerWordRep::PowerWordRep ()` `[private]`

Referenced by `clone()`.

**9.108.3.2** `PowerWordRep::PowerWordRep (const list< PII > & gens)`  
[private]

**9.108.3.3** `PowerWordRep::PowerWordRep (const vector< PII > & gens)`  
[private]

**9.108.3.4** `PowerWordRep::PowerWordRep (const PowerWordRep & wr)`  
[private]

**9.108.3.5** `PowerWordRep::PowerWordRep (const list< int > & gens)`  
[private]

**9.108.3.6** `PowerWordRep::PowerWordRep (const vector< int > & gens)`  
[private]

**9.108.3.7** `PowerWordRep::PowerWordRep (int g, int p = 1)` [private]

## **9.108.4 Member Function Documentation**

**9.108.4.1** `PowerWordRep* PowerWordRep::clone () const` [inline]

Definition at line 83 of file PowerWordRep.h.

References PowerWordRep().

**9.108.4.2** `void PowerWordRep::cyclicallyPermute (int n)` [private]

**9.108.4.3** `void PowerWordRep::cyclicallyReduce (PowerWordRep & conjugator)` [private]

**9.108.4.4** `void PowerWordRep::cyclicallyReduce ()` [private]

**9.108.4.5** `void PowerWordRep::cyclicLeftShift ()` [private]

Referenced by PowerWord::cyclicLeftShift().

**9.108.4.6** `void PowerWordRep::cyclicRightShift ()` [private]

Referenced by PowerWord::cyclicRightShift().

**9.108.4.7** `bool PowerWordRep::doesContain (int gen) const` [private]

Referenced by PowerWord::doesContain().



**9.108.4.8 int PowerWordRep::exponentSum (int *gen*) const [private]**

Referenced by PowerWord::exponentSum().

**9.108.4.9 list< PII >& PowerWordRep::getList () [inline]**

Definition at line 86 of file PowerWordRep.h.

References theElements.

**9.108.4.10 const list< PII >& PowerWordRep::getList () const [inline]**

Definition at line 85 of file PowerWordRep.h.

References theElements.

Referenced by PowerWord::getList().

**9.108.4.11 int PowerWordRep::getPower (PowerWordRep & *base*) const [private]**

Referenced by PowerWord::getPower().

**9.108.4.12 void PowerWordRep::initialSegment (int *to*) [private]****9.108.4.13 void PowerWordRep::insert (const PowerWordRep & *wr*, int *pos*) [private]****9.108.4.14 PowerWordRep PowerWordRep::inverse () const [private]**

Referenced by PowerWord::inverse(), and PowerWord::operator-().

**9.108.4.15 int PowerWordRep::isIn (int *gen*) const [private]**

Referenced by PowerWord::isIn().

**9.108.4.16 int PowerWordRep::length () const [inline, private]**

Definition at line 93 of file PowerWordRep.h.

References theLength.

Referenced by PowerWord::length().

**9.108.4.17** `PowerWordRep PowerWordRep::operator* (const PowerWordRep & w) const [inline, private]`

Definition at line 69 of file PowerWordRep.h.

**9.108.4.18** `PowerWordRep& PowerWordRep::operator*= (const PowerWordRep & w) [private]`

**9.108.4.19** `bool PowerWordRep::operator< (const PowerWordRep & wr) const [private]`

**9.108.4.20** `PowerWordRep PowerWordRep::operator= (const PowerWordRep wr) [private]`

**9.108.4.21** `bool PowerWordRep::operator== (const PowerWordRep & wr) const [private]`

**9.108.4.22** `bool PowerWordRep::operator> (const PowerWordRep & wr) const [private]`

**9.108.4.23** `ostream& PowerWordRep::printOn (ostream & os) const [inline]`

Definition at line 141 of file PowerWordRep.h.

References theElements.

Referenced by PowerWord::printOn().

**9.108.4.24** `void PowerWordRep::pushGeneratorBack (int g, int p) [private]`

Referenced by PowerWord::push\_back().

**9.108.4.25** `void PowerWordRep::pushGeneratorFront (int g, int p) [private]`

Referenced by PowerWord::push\_front().

**9.108.4.26** void PowerWordRep::segment (int *from*, int *to*) [private]

**9.108.4.27** void PowerWordRep::terminalSegment (int *from*) [private]

## 9.108.5 Friends And Related Function Documentation

**9.108.5.1** friend class PowerWord [friend]

Definition at line 30 of file PowerWordRep.h.

## 9.108.6 Member Data Documentation

**9.108.6.1** list< PII > PowerWordRep::theElements [private]

Definition at line 173 of file PowerWordRep.h.

Referenced by getList(), and printOn().

**9.108.6.2** int PowerWordRep::theLength [private]

Definition at line 176 of file PowerWordRep.h.

Referenced by length().

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

- Elt/include/PowerWordRep.h

## 9.109 StraightLineProgramWord::Production Struct Reference

A production for a rule of a composition system.

### Public Member Functions

- **Production** (int t1=0, int t2=0, **LongInteger** l=0, int h=0)

### Public Attributes

- int **theTerm1**
- int **theTerm2**
- **LongInteger** **theLength**
- int **theHeight**
- bool **reduced**

### 9.109.1 Detailed Description

A production for a rule of a composition system. Each rule of a composition system is of the form  $X_i \rightarrow X_j^{\pm 1} \cdot X_k^{\pm 1}$  or  $X_i \rightarrow X_j^{\pm 1}$ . If  $j$  or  $k$  is zero then that part does not exist.

Definition at line 62 of file StraightLineProgramWord.h.

### 9.109.2 Constructor & Destructor Documentation

#### 9.109.2.1 StraightLineProgramWord::Production::Production (int t1 = 0, int t2 = 0, LongInteger l = 0, int h = 0) [inline]

Definition at line 64 of file StraightLineProgramWord.h.

### 9.109.3 Member Data Documentation

#### 9.109.3.1 bool StraightLineProgramWord::Production::reduced

Definition at line 76 of file StraightLineProgramWord.h.

#### 9.109.3.2 int StraightLineProgramWord::Production::theHeight

Definition at line 75 of file StraightLineProgramWord.h.

**9.109.3.3 LongInteger StraightLineProgramWord::Production::theLength**

Definition at line 74 of file StraightLineProgramWord.h.

**9.109.3.4 int StraightLineProgramWord::Production::theTerm1**

Definition at line 72 of file StraightLineProgramWord.h.

**9.109.3.5 int StraightLineProgramWord::Production::theTerm2**

Definition at line 73 of file StraightLineProgramWord.h.

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

- FreeGroup/include/**StraightLineProgramWord.h**

## 9.110 QuadEquationTransformationGraph Class Reference

```
#include <QuadEquatTransformationGraph.h>
```

### Public Member Functions

- **QuadEquationTransformationGraph** (const **Equation** &eq)
- void **extend** ()
- const **IntLabeledGraph** & **getGraph** () const  
*Get the current graph.*
- bool **isDone** () const  
*Determine if the graph is completely constructed.*
- bool **solutionFound** () const  
*Determine if the equation has solutions.*

### Private Member Functions

- set< **Word** > **getNeighbours** (const **Word** &eq)  
*Function returns all neighbours of the vertex in the graph.*
- **Word** **applyAdjointTransformation** (const **Word** &eq, int x, const **Word** &im)  
*Function is used in **getNeighbours**( ) (p. 449).*
- bool **isTrivialSolution** (const **Word** &e) const  
*Check if an equation e has trivial solution.*

### Private Attributes

- const **Equation** **theEquation**  
*Equation (p. 195) under consideration.*
- **IntLabeledGraph** **theGraph**  
*The transformation graph.*
- map< **Word**, int > **processedEquations**

*Already processed equations and their numbers in the graph.*

- `map< Word, int > equationsInProgress`

*New equations and their numbers in the graph.*

- `bool hasSolution`

*Flag. True if the equation with the trivial solution is found.*

### 9.110.1 Detailed Description

Definition at line 22 of file QuadEquatTransformationGraph.h.

### 9.110.2 Constructor & Destructor Documentation

#### 9.110.2.1 QuadEquationTransformationGraph::QuadEquationTransformationGraph (const Equation & eq)

### 9.110.3 Member Function Documentation

#### 9.110.3.1 Word QuadEquationTransformationGraph::applyAdjointTransformation (const Word & eq, int x, const Word & im) [private]

Function is used in `getNeighbours()` (p. 449).

#### 9.110.3.2 void QuadEquationTransformationGraph::extend ()

#### 9.110.3.3 const IntLabeledGraph& QuadEquationTransformationGraph::getGraph () const

Get the current graph.

#### 9.110.3.4 set< Word > QuadEquationTransformationGraph::getNeighbours (const Word & eq) [private]

Function returns all neighbours of the vertex in the graph.

#### 9.110.3.5 bool QuadEquationTransformationGraph::isDone () const

Determine if the graph is completely constructed.

### 9.110.3.6 `bool QuadEquationTransformationGraph::isTrivialSolution (const Word & e) const [private]`

Check if an equation *e* has trivial solution.

### 9.110.3.7 `bool QuadEquationTransformationGraph::solutionFound () const`

Determine if the equation has solutions.

## 9.110.4 Member Data Documentation

### 9.110.4.1 `map< Word , int > QuadEquationTransformationGraph::equationsInProgress [private]`

New equations and their numbers in the graph.

Definition at line 112 of file QuadEquatTransformationGraph.h.

### 9.110.4.2 `bool QuadEquationTransformationGraph::hasSolution [private]`

Flag. True if the equation with the trivial solution is found.

Definition at line 116 of file QuadEquatTransformationGraph.h.

### 9.110.4.3 `map< Word , int > QuadEquationTransformationGraph::processedEquations [private]`

Already processed equations and their numbers in the graph.

Definition at line 108 of file QuadEquatTransformationGraph.h.

### 9.110.4.4 `const Equation QuadEquationTransformationGraph::theEquation [private]`

**Equation** (p. 195) under consideration.

Definition at line 95 of file QuadEquatTransformationGraph.h.

### 9.110.4.5 `IntLabeledGraph QuadEquationTransformationGraph::theGraph [private]`

The transformation graph. A graph with edges labelled by integers. Labels of edges are all ones at the moment. In the future they will encode corresponding transformations.



Definition at line 104 of file QuadEquatTransformationGraph.h.

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

- Equation/include/**QuadEquatTransformationGraph.h**

## 9.111 `quadruple< T1, T2, T3, T4 >` Class Template Reference

```
#include <tuples.h>
```

### Public Member Functions

- **`quadruple`** (`const T1 &t1=T1()`, `const T2 &t2=T2()`, `const T3 &t3=T3()`, `const T4 &t4=T4()`)
- **`operator<`** (`const quadruple &q`) `const`

### Public Attributes

- **`T1 first`**
- **`T2 second`**
- **`T3 third`**
- **`T4 fourth`**

### Friends

- `ostream & operator<<` (`ostream &os`, `const quadruple &t`)

#### 9.111.1 Detailed Description

```
template<class T1, class T2, class T3, class T4> class quadruple< T1, T2, T3, T4 >
```

Definition at line 113 of file `tuples.h`.

#### 9.111.2 Constructor & Destructor Documentation

**9.111.2.1** `template<class T1, class T2, class T3, class T4> quadruple< T1, T2, T3, T4 >::quadruple (const T1 & t1 = T1 (), const T2 & t2 = T2 (), const T3 & t3 = T3 (), const T4 & t4 = T4 ()) [inline]`

Definition at line 124 of file `tuples.h`.

### 9.111.3 Member Function Documentation

**9.111.3.1** `template<class T1, class T2, class T3, class T4> bool quadruple< T1, T2, T3, T4 >::operator< (const quadruple< T1, T2, T3, T4 > & q)  
const [inline]`

Definition at line 135 of file tuples.h.

### 9.111.4 Friends And Related Function Documentation

**9.111.4.1** `template<class T1, class T2, class T3, class T4> ostream&  
operator<< (ostream & os, const quadruple< T1, T2, T3, T4 > & t)  
[friend]`

Definition at line 164 of file tuples.h.

### 9.111.5 Member Data Documentation

**9.111.5.1** `template<class T1, class T2, class T3, class T4> T1 quadruple< T1, T2, T3, T4 >::first`

Definition at line 177 of file tuples.h.

Referenced by quadruple< Word, Word, Word, Word >::operator<().

**9.111.5.2** `template<class T1, class T2, class T3, class T4> T4 quadruple< T1, T2, T3, T4 >::fourth`

Definition at line 180 of file tuples.h.

Referenced by quadruple< Word, Word, Word, Word >::operator<().

**9.111.5.3** `template<class T1, class T2, class T3, class T4> T2 quadruple< T1, T2, T3, T4 >::second`

Definition at line 178 of file tuples.h.

Referenced by quadruple< Word, Word, Word, Word >::operator<().

**9.111.5.4** `template<class T1, class T2, class T3, class T4> T3 quadruple< T1, T2, T3, T4 >::third`

Definition at line 179 of file tuples.h.

Referenced by `quadruple< Word, Word, Word, Word >::operator<()`.

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

- `general/include/tuples.h`

## 9.112 quintuple< T1, T2, T3, T4, T5 > Class Template Reference

```
#include <tuple.h>
```

### Public Member Functions

- **quintuple** (const T1 &t1=T1(), const T2 &t2=T2(), const T3 &t3=T3(), const T4 &t4=T4(), const T5 &t5=T5())
- bool **operator**< (const **quintuple** &q) const

### Public Attributes

- T1 **first**
- T2 **second**
- T3 **third**
- T4 **fourth**
- T4 **fifth**

### Friends

- ostream & **operator**<< (ostream &os, const **quintuple** &t)

#### 9.112.1 Detailed Description

**template**<class T1, class T2, class T3, class T4, class T5> class quintuple< T1, T2, T3, T4, T5 >

Definition at line 190 of file tuple.h.

#### 9.112.2 Constructor & Destructor Documentation

**9.112.2.1** **template**<class T1, class T2, class T3, class T4, class T5> quintuple< T1, T2, T3, T4, T5 >::quintuple (const T1 &*t1* = T1 (), const T2 &*t2* = T2 (), const T3 &*t3* = T3 (), const T4 &*t4* = T4 (), const T5 &*t5* = T5 ()) [**inline**]

Definition at line 201 of file tuple.h.

### 9.112.3 Member Function Documentation

**9.112.3.1** `template<class T1, class T2, class T3, class T4, class T5> bool  
quintuple< T1, T2, T3, T4, T5 >::operator< (const quintuple< T1,  
T2, T3, T4, T5 > & q) const [inline]`

Definition at line 212 of file tuples.h.

### 9.112.4 Friends And Related Function Documentation

**9.112.4.1** `template<class T1, class T2, class T3, class T4, class T5> ostream&  
operator<< (ostream & os, const quintuple< T1, T2, T3, T4, T5 > &  
t) [friend]`

Definition at line 246 of file tuples.h.

### 9.112.5 Member Data Documentation

**9.112.5.1** `template<class T1, class T2, class T3, class T4, class T5> T4  
quintuple< T1, T2, T3, T4, T5 >::fifth`

Definition at line 264 of file tuples.h.

Referenced by `quintuple< Word, Word, Word, Word, Word >::operator<()`.

**9.112.5.2** `template<class T1, class T2, class T3, class T4, class T5> T1  
quintuple< T1, T2, T3, T4, T5 >::first`

Definition at line 260 of file tuples.h.

Referenced by `quintuple< Word, Word, Word, Word, Word >::operator<()`.

**9.112.5.3** `template<class T1, class T2, class T3, class T4, class T5> T4  
quintuple< T1, T2, T3, T4, T5 >::fourth`

Definition at line 263 of file tuples.h.

Referenced by `quintuple< Word, Word, Word, Word, Word >::operator<()`.

**9.112.5.4** `template<class T1, class T2, class T3, class T4, class T5> T2  
quintuple< T1, T2, T3, T4, T5 >::second`

Definition at line 261 of file tuples.h.

Referenced by quintuple< Word, Word, Word, Word, Word >::operator<().

**9.112.5.5    template<class T1, class T2, class T3, class T4, class T5> T3  
              quintuple< T1, T2, T3, T4, T5 >::third**

Definition at line 262 of file tuples.h.

Referenced by quintuple< Word, Word, Word, Word, Word >::operator<().

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

- general/include/**tuples.h**

## 9.113 RandLib Class Reference

Static wrapper class for RANDLIB Library.

```
#include <RanlibCPP.h>
```

### Static Public Member Functions

- static double **chiPValue** (int idf, double alpha)  
*Returns a p-value for the chi distribution.*

### Static Public Attributes

- static **RandLibURG ur**  
*The static instance of the wrapper class.*

#### 9.113.1 Detailed Description

Static wrapper class for RANDLIB Library.

Definition at line 105 of file RanlibCPP.h.

#### 9.113.2 Member Function Documentation

##### 9.113.2.1 static double RandLib::chiPValue (int idf, double alpha) [inline, static]

Returns a p-value for the chi distribution.

Definition at line 114 of file RanlibCPP.h.

References cdfchi().

#### 9.113.3 Member Data Documentation

##### 9.113.3.1 RandLibURG RandLib::ur [static]

The static instance of the wrapper class. This is done to make sure that the random number generator is seeded and only once.

Definition at line 110 of file RanlibCPP.h.



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

- ranlib/include/**RanlibCPP.h**

## 9.114 RandLibURG Class Reference

A wrapper class for RANLIB Library.

```
#include <RandlibCPP.h>
```

### Public Member Functions

- **RandLibURG** (long i1, long i2)  
*Constructor.*
- **RandLibURG** ()  
*Default constructor. The seeds of the random number generator a set using current time values\*./*
- void **reset** (long i1, long i2)  
*Set the seeds of random generator to i1 and u2.*
- void **reset** ()  
*Set the seeds of the random number generator using current time values.*
- void **setSeedPID** ()  
*Set the seeds of the random number generator using current time and process id values.*
- void **getseed** (long &s1, long &s2)  
*Get the seeds of the random number generator.*
- float **rand** ()  
*Returns a float point pseudo random number in [0,1].*
- float **rand** (float low, float high)  
*Returns a float point pseudo random number in [low,high].*
- long **irand** (long low, long high)  
*Returns an integer pseudo random number in [low,high].*

### 9.114.1 Detailed Description

A wrapper class for RANLIB Library. This is a wrapper class for functions implemented in

RANLIB Library of Fortran Routines for Random Number Generation, Compiled and Written by:

Barry W. Brown and James Lovato

Department of Biomathematics, Box 237, The University of Texas, M.D. Anderson Cancer Center, 1515 Holcombe Boulevard, Houston, TX 77030

Definition at line 37 of file RanlibCPP.h.

## 9.114.2 Constructor & Destructor Documentation

### 9.114.2.1 RandLibURG::RandLibURG (long *i1*, long *i2*) [inline]

Constructor. Set the seeds of random generator to *i1* and *i2*

Definition at line 42 of file RanlibCPP.h.

References `setall()`.

### 9.114.2.2 RandLibURG::RandLibURG () [inline]

Default constructor. The seeds of the random number generator are set using current time values\*/\*.

Definition at line 47 of file RanlibCPP.h.

References `irand()`, and `setall()`.

## 9.114.3 Member Function Documentation

### 9.114.3.1 void RandLibURG::getseed (long & *s1*, long & *s2*) [inline]

Get the seeds of the random number generator.

Definition at line 79 of file RanlibCPP.h.

References `getsd()`.

### 9.114.3.2 long RandLibURG::irand (long *low*, long *high*) [inline]

Returns an integer pseudo random number in [*low*,*high*].

Definition at line 98 of file RanlibCPP.h.

References `ignuin()`.

Referenced by `RandLibURG()`, `reset()`, and `setSeedPID()`.

**9.114.3.3 float RandLibURG::rand (float *low*, float *high*) [inline]**

Returns a float point pseudo random number in [low,high].

Definition at line 92 of file RanlibCPP.h.

References `genunf()`.

**9.114.3.4 float RandLibURG::rand () [inline]**

Returns a float point pseudo random number in [0,1].

Definition at line 86 of file RanlibCPP.h.

References `ranf()`.

**9.114.3.5 void RandLibURG::reset () [inline]**

Set the seeds of the random number generator using current time values.

Definition at line 60 of file RanlibCPP.h.

References `irand()`, and `setall()`.

**9.114.3.6 void RandLibURG::reset (long *i1*, long *i2*) [inline]**

Set the seeds of random generator to `i1` and `u2`.

Definition at line 55 of file RanlibCPP.h.

References `setall()`.

**9.114.3.7 void RandLibURG::setSeedPID () [inline]**

Set the seeds of the random number generator using current time and process id values.

Definition at line 68 of file RanlibCPP.h.

References `irand()`, and `setall()`.

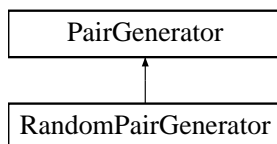
The documentation for this class was generated from the following file:

- `ranlib/include/RanlibCPP.h`

## 9.115 RandomPairGenerator Class Reference

Implements a class for generating pseudo-random pairs of words.

`#include <PairDistanceTest.h>`Inheritance diagram for RandomPairGenerator::



### Public Member Functions

- **RandomPairGenerator** (int N, int len)  
*Constructor.*
- pair< **Word**, **Word** > **getTruePair** ()  
*Returns a pair of pseudo-random, independently generated words.*
- pair< **Word**, **Word** > **getFalsePair** ()  
*Returns a pair of correlated words.*

### Private Attributes

- int **nGens**
- int **Len**

#### 9.115.1 Detailed Description

Implements a class for generating pseudo-random pairs of words.

Definition at line 48 of file PairDistanceTest.h.

#### 9.115.2 Constructor & Destructor Documentation

##### 9.115.2.1 RandomPairGenerator::RandomPairGenerator (int N, int len) [inline]

Constructor.

**Parameters:**

*N* - number of generators  
*len* - the length of each word

Definition at line 56 of file PairDistanceTest.h.

**9.115.3 Member Function Documentation****9.115.3.1 pair<Word,Word> RandomPairGenerator::getFalsePair ()  
[inline, virtual]**

Returns a pair of correlated words. There are many ways to create correlated words. This is just for testing. First word *w1* is generated randomly and the second *w2* = *w1*. Thereofre they are equal.

Implements **PairGenerator** (p. 353).

Definition at line 75 of file PairDistanceTest.h.

References *Len*, *nGens*, and *Word::randomWord()*.

**9.115.3.2 pair<Word,Word> RandomPairGenerator::getTruePair ()  
[inline, virtual]**

Returns a pair of pseudo-random, independently generated words.

Implements **PairGenerator** (p. 353).

Definition at line 61 of file PairDistanceTest.h.

References *Len*, *nGens*, and *Word::randomWord()*.

**9.115.4 Member Data Documentation****9.115.4.1 int RandomPairGenerator::Len [private]**

Definition at line 85 of file PairDistanceTest.h.

Referenced by *getFalsePair()*, and *getTruePair()*.

**9.115.4.2 int RandomPairGenerator::nGens [private]**

Definition at line 84 of file PairDistanceTest.h.

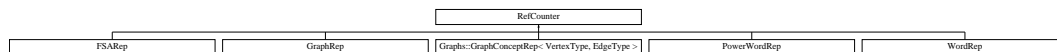
Referenced by *getFalsePair()*, and *getTruePair()*.

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

- StringSimilarity/include/**PairDistanceTest.h**

## 9.116 RefCounter Class Reference

`#include <RefCounter.h>` Inheritance diagram for RefCounter::



### Public Types

- typedef unsigned int **refCounterType**

### Public Member Functions

- **RefCounter** ()
- **RefCounter** (const **RefCounter** &rc)
- bool **lastRef** () const
- bool **sharedRef** () const
- void **addRef** () const
- void **delRef** () const

### Private Member Functions

- **RefCounter** & **operator=** (const **RefCounter** &)

### Private Attributes

- refCounterType xrefs

#### 9.116.1 Detailed Description

Definition at line 9 of file RefCounter.h.

#### 9.116.2 Member Typedef Documentation

##### 9.116.2.1 typedef unsigned int RefCounter::refCounterType

Definition at line 19 of file RefCounter.h.



### 9.116.3 Constructor & Destructor Documentation

#### 9.116.3.1 RefCounter::RefCounter () [inline]

Definition at line 27 of file RefCounter.h.

#### 9.116.3.2 RefCounter::RefCounter (const RefCounter & rc) [inline]

Definition at line 31 of file RefCounter.h.

### 9.116.4 Member Function Documentation

#### 9.116.4.1 void RefCounter::addRef () const [inline]

Definition at line 48 of file RefCounter.h.

#### 9.116.4.2 void RefCounter::delRef () const [inline]

Definition at line 51 of file RefCounter.h.

#### 9.116.4.3 bool RefCounter::lastRef () const [inline]

Definition at line 44 of file RefCounter.h.

References xrefs.

#### 9.116.4.4 RefCounter& RefCounter::operator= (const RefCounter &) [private]

#### 9.116.4.5 bool RefCounter::sharedRef () const [inline]

Definition at line 46 of file RefCounter.h.

References xrefs.

### 9.116.5 Member Data Documentation

#### 9.116.5.1 refCounterType RefCounter::xrefs [private]

Definition at line 66 of file RefCounter.h.

Referenced by lastRef(), and sharedRef().

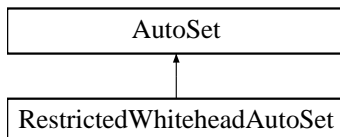
The documentation for this class was generated from the following file:

- `general/include/RefCounter.h`

## 9.117 RestrictedWhiteheadAutoSet Class Reference

Implements a so-called restricted set of Whitehead automorphisms of a free group.

`#include <WhiteheadAutoSet.h>` Inheritance diagram for RestrictedWhiteheadAutoSet::



### Public Member Functions

- **RestrictedWhiteheadAutoSet** (int n, bool use\_conj=true)  
*Constructor. Generates a set of restricted Whitehead automorphisms.*
- **~RestrictedWhiteheadAutoSet** ()
- const **Map** & **getRandomAuto** () const  
*Returns a random restricted Whitehead automorphism.*
- const **SetOfMaps** & **getSet** () const  
*Returns the set of restricted Whitehead automorphisms.*

### Private Attributes

- **SetOfMaps** theSet
- int nGens

#### 9.117.1 Detailed Description

Implements a so-called restricted set of Whitehead automorphisms of a free group. In general the Restricted set of Whitehead automorphisms contains automorphisms of the type:

$$x_i \rightarrow x_i x_j$$

$$x_i \rightarrow x_j^{-1} x_i$$

$$x_i \rightarrow x_j^{-1} x_i x_j,$$

where  $x_j, x_j \in X$  - generating set of the free group.

Definition at line 111 of file WhiteheadAutoSet.h.

## 9.117.2 Constructor & Destructor Documentation

### 9.117.2.1 `RestrictedWhiteheadAutoSet::RestrictedWhiteheadAutoSet (int n, bool use_conj = true)`

Constructor. Generates a set of restricted Whitehead automorphisms. The set of restricted Whitehead automorphisms for a free group of rank *n* is generated.

Note, the whole set is generated by enumeration.

#### Parameters:

*n* - rank of a free group.

*use\_conj* - if `false` the conjugation is excluded from the set. `true` is the default value.

### 9.117.2.2 `RestrictedWhiteheadAutoSet::~~RestrictedWhiteheadAutoSet ()` [inline]

Definition at line 133 of file WhiteheadAutoSet.h.

## 9.117.3 Member Function Documentation

### 9.117.3.1 `const Map& RestrictedWhiteheadAutoSet::getRandomAuto () const`

Returns a random restricted Whitehead automorphism.

#### Returns:

restricted Whitehead automorphisms selected uniformly randomly from the set.

### 9.117.3.2 `const SetOfMaps& RestrictedWhiteheadAutoSet::getSet () const` [inline, virtual]

Returns the set of restricted Whitehead automorphisms.

#### Returns:

the set of restricted Whitehead automorphisms.

Implements **AutoSet** (p. 110).

Definition at line 153 of file WhiteheadAutoSet.h.

References theSet.

#### 9.117.4 Member Data Documentation

##### 9.117.4.1 `int RestrictedWhiteheadAutoSet::nGens` `[private]`

Definition at line 156 of file WhiteheadAutoSet.h.

##### 9.117.4.2 `SetOfMaps RestrictedWhiteheadAutoSet::theSet` `[private]`

Definition at line 155 of file WhiteheadAutoSet.h.

Referenced by `getSet()`.

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

- `Maps/include/WhiteheadAutoSet.h`

## 9.118 RGB Class Reference

```
#include <WordDraw.h>
```

### Public Member Functions

- **RGB** (double *r*, double *g*, double *b*)
- **RGB** (double *gray*)

### Public Attributes

- double **R**
- double **G**
- double **B**

#### 9.118.1 Detailed Description

Definition at line 174 of file WordDraw.h.

#### 9.118.2 Constructor & Destructor Documentation

##### 9.118.2.1 RGB::RGB (double *r*, double *g*, double *b*) [inline]

Definition at line 177 of file WordDraw.h.

##### 9.118.2.2 RGB::RGB (double *gray*) [inline]

Definition at line 178 of file WordDraw.h.

#### 9.118.3 Member Data Documentation

##### 9.118.3.1 double RGB::B

Definition at line 183 of file WordDraw.h.

##### 9.118.3.2 double RGB::G

Definition at line 182 of file WordDraw.h.

### 9.118.3.3 double RGB::R

Definition at line 181 of file WordDraw.h.

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

- Graphics/include/**WordDraw.h**

## 9.119 PC::Row Struct Reference

```
#include <SignMatrix.h>
```

### Public Member Functions

- **Row ()**

### Public Attributes

- unsigned int **rowid**

### Static Public Attributes

- static const unsigned int **UNDEFINED** = ~0

#### 9.119.1 Detailed Description

Definition at line 24 of file SignMatrix.h.

#### 9.119.2 Constructor & Destructor Documentation

##### 9.119.2.1 PC::Row::Row () [inline]

Definition at line 24 of file SignMatrix.h.

#### 9.119.3 Member Data Documentation

##### 9.119.3.1 unsigned int PC::Row::rowid

Definition at line 24 of file SignMatrix.h.

Referenced by PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::deleteRow(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::getRow(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::getLineNumber(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::insertRow(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::insertRowCopy(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::insertUnitCol(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::moveRow(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE



>::operator>(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::rowHdr(), and PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::setEntry().

#### 9.119.3.2 `const unsigned int PC::Row::UNDEFINED = ~0` `[static]`

Definition at line 24 of file SignMatrix.h.

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

- HigmanGroup/include/**SignMatrix.h**

## 9.120 PC::PowerCircuitCompMatrix::RowHdr Struct Reference

### Public Member Functions

- **RowHdr** ()
- **RowHdr** (const **Marking** &l, bool bv, int i)

### Public Attributes

- **Marking** lambdaMarking
- bool **BV**
- unsigned int **indexInNodes**

#### 9.120.1 Detailed Description

Definition at line 19 of file PowerCircuitCompMatrix.h.

#### 9.120.2 Constructor & Destructor Documentation

##### 9.120.2.1 PC::PowerCircuitCompMatrix::RowHdr::RowHdr () [inline]

Definition at line 25 of file PowerCircuitCompMatrix.h.

##### 9.120.2.2 PC::PowerCircuitCompMatrix::RowHdr::RowHdr (const **Marking** &l, bool bv, int i) [inline]

Definition at line 26 of file PowerCircuitCompMatrix.h.

#### 9.120.3 Member Data Documentation

##### 9.120.3.1 bool PC::PowerCircuitCompMatrix::RowHdr::BV

Definition at line 22 of file PowerCircuitCompMatrix.h.

##### 9.120.3.2 unsigned int PC::PowerCircuitCompMatrix::RowHdr::indexInNodes

Definition at line 23 of file PowerCircuitCompMatrix.h.

**9.120.3.3 Marking PC::PowerCircuitCompMatrix::RowHdr::lambdaMarking**

Definition at line 21 of file PowerCircuitCompMatrix.h.

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

- HigmanGroup/include/**PowerCircuitCompMatrix.h**

## 9.121 ShftConjKeyInstance Class Reference

```
#include <ShftConjKeyGeneration.h>
```

### Public Member Functions

- **ShftConjKeyInstance** (int braid\_rank, **Word** publicKeyA, **Word** privateKey)  
*Create an instance of the protocol.*
- int **getBraidRank** () const  
*(accessor function) get the rank of the braid group*
- **Word** **getPrivateKey** () const  
*(accessor function) get the private key*
- pair< **Word**, **Word** > **getPublicKey** () const  
*(accessor function) get the public key*

### Static Public Member Functions

- static **ShftConjKeyInstance** **random** (int braid\_rank, int baseLenth, int keyLength)  
*Generate a random instance of the protocol.*

### Private Attributes

- int **theRank**  
*the rank of the braid group*
- pair< **Word**, **Word** > **thePublicKey**  
*the public key of the first party  $P_B = D(sh(A)^{-1} \sigma_1 sh(w)A)$*
- **Word** **thePrivateKey**  
*the private key of the second party (denoted by  $B$ )*

#### 9.121.1 Detailed Description

Definition at line 23 of file ShftConjKeyGeneration.h.

## 9.121.2 Constructor & Destructor Documentation

### 9.121.2.1 ShftConjKeyInstance::ShftConjKeyInstance (int *braid\_rank*, Word *publicKeyA*, Word *privateKey*)

Create an instance of the protocol.

## 9.121.3 Member Function Documentation

### 9.121.3.1 int ShftConjKeyInstance::getBraidRank () const [inline]

(accessor function) get the rank of the braid group

Definition at line 55 of file ShftConjKeyGeneration.h.

References theRank.

### 9.121.3.2 Word ShftConjKeyInstance::getPrivateKey () const [inline]

(accessor function) get the private key

Definition at line 57 of file ShftConjKeyGeneration.h.

References thePrivateKey.

### 9.121.3.3 pair< Word , Word > ShftConjKeyInstance::getPublicKey () const [inline]

(accessor function) get the public key

Definition at line 59 of file ShftConjKeyGeneration.h.

References thePublicKey.

### 9.121.3.4 static ShftConjKeyInstance ShftConjKeyInstance::random (int *braid\_rank*, int *baseLenth*, int *keyLength*) [static]

Generate a random instance of the protocol.

#### Parameters:

*braid\_rank* - rank of the braid group;

*baseLenth* - the length of thePublicKey.first

*keyLength* - the length of thePrivateKey

## 9.121.4 Member Data Documentation

### 9.121.4.1 Word ShftConjKeyInstance::thePrivateKey [private]

the private key of the second party (denoted by  $B$ )

Definition at line 77 of file ShftConjKeyGeneration.h.

Referenced by getPrivateKey().

### 9.121.4.2 pair< Word , Word > ShftConjKeyInstance::thePublicKey [private]

the public key of the first party  $P_B = D(sh(A)^{-1}\sigma_1 sh(w)A)$

Definition at line 74 of file ShftConjKeyGeneration.h.

Referenced by getPublicKey().

### 9.121.4.3 int ShftConjKeyInstance::theRank [private]

the rank of the braid group

Definition at line 71 of file ShftConjKeyGeneration.h.

Referenced by getBraidRank().

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

- CryptoShftConj/include/ShftConjKeyGeneration.h

## 9.122 ShftConjKeyInstanceGarside Class Reference

Container for public and private information in Dehornoy's authentication protocol.

```
#include <ShftConjKeyGenerationGarside.h>
```

### Public Member Functions

- **ShftConjKeyInstanceGarside** (int braid\_rank, **ThRightNormalForm** publicKeyA, **ThRightNormalForm** privateKey)  
*Create an instance of the protocol.*
- int **getBraidRank** () const  
*(accessor function) get the rank of the braid group*
- **ThRightNormalForm** **getPrivateKey** () const  
*(accessor function) get the private key*
- pair< **ThRightNormalForm**, **ThRightNormalForm** > **getPublicKey** () const  
*(accessor function) get the public key*

### Static Public Member Functions

- static **ShftConjKeyInstanceGarside** **random** (int braid\_rank, int baseLenth, int keyLength)  
*Generate a random instance of the protocol.*

### Private Attributes

- int **theRank**  
*the rank of the braid group*
- **ThRightNormalForm** **thePrivateKey**  
*the private key of the first party (denoted by  $A$ )*
- pair< **ThRightNormalForm**, **ThRightNormalForm** > **thePublicKey**  
*the public key of the first party  $P_B = D(sh(A)^{-1}\sigma_1 sh(w)A)$*

### 9.122.1 Detailed Description

Container for public and private information in Dehornoy's authentication protocol. Use this class to generate and keep an instance of Dehornoy's authentication protocol. Definition at line 31 of file ShftConjKeyGenerationGarside.h.

### 9.122.2 Constructor & Destructor Documentation

#### 9.122.2.1 ShftConjKeyInstanceGarside::ShftConjKeyInstanceGarside (int *braid\_rank*, ThRightNormalForm *publicKeyA*, ThRightNormalForm *privateKey*)

Create an instance of the protocol.

### 9.122.3 Member Function Documentation

#### 9.122.3.1 int ShftConjKeyInstanceGarside::getBraidRank () const [inline]

(accessor function) get the rank of the braid group

Definition at line 66 of file ShftConjKeyGenerationGarside.h.

References theRank.

#### 9.122.3.2 ThRightNormalForm ShftConjKeyInstanceGarside::getPrivateKey () const [inline]

(accessor function) get the private key

Definition at line 68 of file ShftConjKeyGenerationGarside.h.

References thePrivateKey.

#### 9.122.3.3 pair< ThRightNormalForm , ThRightNormalForm > ShftConjKeyInstanceGarside::getPublicKey () const [inline]

(accessor function) get the public key

Definition at line 70 of file ShftConjKeyGenerationGarside.h.

References thePublicKey.



#### 9.122.3.4 static ShftConjKeyInstanceGarside ShftConjKeyInstanceGarside::random (int *braid\_rank*, int *baseLenth*, int *keyLength*) [static]

Generate a random instance of the protocol.

##### Parameters:

- braid\_rank* - rank of the braid group;
- baseLenth* - the length of the base word
- keyLength* - the length of the key

### 9.122.4 Member Data Documentation

#### 9.122.4.1 ThRightNormalForm ShftConjKeyInstanceGarside::thePrivateKey [private]

the private key of the first party (denoted by  $A$ )

Definition at line 85 of file ShftConjKeyGenerationGarside.h.

Referenced by getPrivateKey().

#### 9.122.4.2 pair< ThRightNormalForm , ThRightNormalForm > ShftConjKeyInstanceGarside::thePublicKey [private]

the public key of the first party  $P_B = D(sh(A)^{-1}\sigma_1 sh(w)A)$

Definition at line 88 of file ShftConjKeyGenerationGarside.h.

Referenced by getPublicKey().

#### 9.122.4.3 int ShftConjKeyInstanceGarside::theRank [private]

the rank of the braid group

Definition at line 82 of file ShftConjKeyGenerationGarside.h.

Referenced by getBraidRank().

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

- CryptoShftConj/include/ShftConjKeyGenerationGarside.h

### 9.123 PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE > Class Template Reference

```
#include <SignMatrix.h>
```

#### Classes

- struct **intColHdr**
- struct **intRowHdr**

#### Public Member Functions

- **SignMatrix** (unsigned int rows=0, unsigned int cols=0, unsigned int maxR=16, unsigned int maxC=16)
- virtual **~SignMatrix** ()
- unsigned int **getNumRows** () const
- unsigned int **getNumCols** () const
- **Row** **getRow** (unsigned int rowNo) const
- unsigned int **getRowNumber** (**Row** row) const
- **Col** **getCol** (unsigned int colNo) const
- unsigned int **getColNumber** (**Col** col) const
- **Sign operator()** (unsigned int rowNo, unsigned int colNo) const
- **Sign operator()** (**Row** row, **Col** col) const
- **Sign operator()** (**Row** row, unsigned int colNo) const
- **Sign operator()** (unsigned int rowNo, **Col** col) const
- void **setEntry** (unsigned int rowNo, unsigned int colNo, **Sign** val)
- void **setEntry** (**Row** row, **Col** col, **Sign** val)
- void **setEntry** (**Row** row, unsigned int colNo, **Sign** val)
- void **setEntry** (unsigned int rowNo, **Col** col, **Sign** val)
- **RowHdr** & **rowHdr** (unsigned int rowNo)
- **RowHdr** & **rowHdr** (**Row** col)
- **ColHdr** & **colHdr** (unsigned int colNo)
- **ColHdr** & **colHdr** (**Col** col)
- bool **columnsEqual** (**Col** col1, **Col** col2)
- **Col** **insertZeroCol** (unsigned int colNo, **ColHdr** colHdr=**ColHdr**())
- **Col** **insertUnitCol** (unsigned int colNo, unsigned int i, **ColHdr** colHdr=**ColHdr**())
- **Col** **insertInvCol** (**Col** sourceCol, unsigned int targetColNo, **ColHdr** targetColHdr=**ColHdr**())

- **Col insertColSum** (**Col** sourceCol1, **Col** sourceCol2, unsigned int targetColNo, ColHdr targetColHdr=ColHdr())
- **Col insertColCopy** (**Col** sourceCol, unsigned int targetColNo)
- **Col insertIntersectionCol** (**Col** sourceCol1, **Col** sourceCol2, unsigned int targetColNo, ColHdr targetColHdr=ColHdr())
- void **addToCol** (**Col** sourceCol, **Col** targetCol)
- void **deleteCol** (**Col** col)
- void **moveCol** (**Col** col, unsigned int newColNo)
- **Row insertZeroRow** (unsigned int rowNo, RowHdr rowHdr=RowHdr())
- **Row insertRowCopy** (**Row** sourceRow, unsigned int targetRowNo)
- void **deleteRow** (**Row** row)
- void **moveRow** (**Row** row, unsigned int newRowNo)
- void **clone** (const **SignMatrix**< RowHdr, ColHdr, INTERNAL\_TYPE > &mat, std::list< **Col** > &colList)
- void **insertMatrix** (const **SignMatrix**< RowHdr, ColHdr, INTERNAL\_TYPE > &mat, unsigned int rowOffset, unsigned int colOffset)
- void **printMatrix** (std::ostream &os=std::cout) const

## Private Member Functions

- void **extend** (unsigned int rows, unsigned int cols)
- **Row insertRow** (unsigned int rowNo, RowHdr rowHdr)
- **Col insertCol** (unsigned int colNo, ColHdr colHdr)
- void **free** ()

## Private Attributes

- unsigned int **numRows**
- unsigned int **numCols**
- unsigned int **maxNumRows**
- unsigned int **maxNumCols**
- int **firstUnusedRow**
- int **firstUnusedCol**
- INTERNAL\_TYPE \* **matrix**
- std::vector< **intRowHdr** > **rowHdrs**
- std::vector< **intColHdr** > **colHdrs**
- const unsigned int **INTERNAL\_SIZE**
- INTERNAL\_TYPE **mask1**
- INTERNAL\_TYPE **mask2**

## Static Private Attributes

- static const double **EXTEND\_CONDITION** = 1.2
- static const double **EXTEND\_FACTOR** = 2.0

### 9.123.1 Detailed Description

**template<class RowHdr, class ColHdr, typename INTERNAL\_TYPE> class PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >**

Definition at line 30 of file SignMatrix.h.

### 9.123.2 Constructor & Destructor Documentation

**9.123.2.1** **template<class RowHdr , class ColHdr , typename INTERNAL\_TYPE > PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::SignMatrix (unsigned int *rows* = 0, unsigned int *cols* = 0, unsigned int *maxR* = 16, unsigned int *maxC* = 16) [inline]**

Constructor

Creates an instance of **SignMatrix** (p. 484) having the indicated number of rows and columns (initially filled with zeros). If there is an estimate on the final size of the matrix, it can be given in order to avoid later memory reallocations.

Definition at line 111 of file SignMatrix.h.

References PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::colHdrs, PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::firstUnusedCol, PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::firstUnusedRow, PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::INTERNAL\_SIZE, PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::mask1, PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::mask2, PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::matrix, PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::maxNumCols, PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::maxNumRows, PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::numCols, PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::numRows, and PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::rowHdrs.

**9.123.2.2** **template<class RowHdr , class ColHdr , typename INTERNAL\_TYPE > PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::~~SignMatrix () [inline, virtual]**

Destructor

Destroys the instance and deallocates any used memory.

Definition at line 185 of file SignMatrix.h.

References PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::free().

### 9.123.3 Member Function Documentation

**9.123.3.1** `template<class RowHdr , class ColHdr , typename INTERNAL_TYPE > void PC::SignMatrix< RowHdr, ColHdr, INTERNAL_TYPE >::addToCol (Col sourceCol, Col targetCol) [inline]`

Adds the entries of one column to another. Assumes that the addition can be done without getting values outside {-1,0,+1}.

Definition at line 477 of file SignMatrix.h.

References PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::colHdrs, PC::Col::colid, PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::INTERNAL\_SIZE, PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::mask1, PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::mask2, PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::matrix, PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::maxNumCols, PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::maxNumRows, and PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::numCols.

**9.123.3.2** `template<class RowHdr, class ColHdr, typename INTERNAL_TYPE> void PC::SignMatrix< RowHdr, ColHdr, INTERNAL_TYPE >::clone (const SignMatrix< RowHdr, ColHdr, INTERNAL_TYPE > & mat, std::list< Col > & colList) [inline]`

Produces a copy of the matrix using only the given columns. The row and column headers are NOT copied.

Definition at line 623 of file SignMatrix.h.

References PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::colHdrs, PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::firstUnusedCol, PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::firstUnusedRow, PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::free(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::INTERNAL\_SIZE, PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::matrix, PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::maxNumCols, PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::maxNumRows, PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::numCols, PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::-

TYPE >::numRows, and PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::rowHdrs.

**9.123.3.3** `template<class RowHdr , class ColHdr , typename  
INTERNAL_TYPE > ColHdr & PC::SignMatrix< RowHdr, ColHdr,  
INTERNAL_TYPE >::colHdr (Col col) [inline]`

Definition at line 331 of file SignMatrix.h.

References PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::colHdrs, PC::Col::colid, PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::maxNumCols, and PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::numCols.

**9.123.3.4** `template<class RowHdr , class ColHdr , typename  
INTERNAL_TYPE > ColHdr & PC::SignMatrix< RowHdr, ColHdr,  
INTERNAL_TYPE >::colHdr (unsigned int colNo) [inline]`

Definition at line 325 of file SignMatrix.h.

References PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::colHdrs, PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::getCol(), and PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::numCols.

**9.123.3.5** `template<class RowHdr , class ColHdr , typename  
INTERNAL_TYPE > bool PC::SignMatrix< RowHdr, ColHdr,  
INTERNAL_TYPE >::columnsEqual (Col col1, Col col2)  
[inline]`

Checks whether two specified columns of the matrix are equal (this doesn't compare the headers!).

Definition at line 358 of file SignMatrix.h.

References PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::colHdrs, PC::Col::colid, PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::getRow(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::maxNumCols, PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::numCols, and PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::numRows.

**9.123.3.6** `template<class RowHdr , class ColHdr , typename  
INTERNAL_TYPE > void PC::SignMatrix< RowHdr, ColHdr,  
INTERNAL_TYPE >::deleteCol (Col col) [inline]`

Deletes a column from the matrix.

### 9.123 PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE > Class Template Reference 489

---

Definition at line 493 of file SignMatrix.h.

References PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::colHdrs, PC::Col::colid, PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::firstUnusedCol, PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::maxNumCols, and PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::numCols.

#### 9.123.3.7 **template<class RowHdr , class ColHdr , typename INTERNAL\_TYPE > void PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::deleteRow (Row row) [inline]**

Deletes a row from the matrix.

Definition at line 571 of file SignMatrix.h.

References PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::firstUnusedRow, PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::maxNumRows, PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::numRows, PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::rowHdrs, and PC::Row::rowid.

#### 9.123.3.8 **template<class RowHdr , class ColHdr , typename INTERNAL\_TYPE > void PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::extend (unsigned int rows, unsigned int cols) [inline, private]**

Extends the matrix to have at least the given number of rows and columns.

Definition at line 706 of file SignMatrix.h.

References PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::colHdrs, PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::EXTEND\_CONDITION, PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::EXTEND\_FACTOR, PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::firstUnusedCol, PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::firstUnusedRow, PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::INTERNAL\_SIZE, PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::matrix, PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::maxNumCols, PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::maxNumRows, PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::numCols, and PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::rowHdrs.

Referenced by PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::insertCol(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::insertMatrix(), and PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::insertRow().

**9.123.3.9** `template<class RowHdr , class ColHdr , typename  
INTERNAL_TYPE > void PC::SignMatrix< RowHdr, ColHdr,  
INTERNAL_TYPE >::free () [inline, private]`

Cleans up and deallocates any memory used by this object

Definition at line 194 of file SignMatrix.h.

References PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::colHdrs, PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::matrix, PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::maxNumCols, PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::maxNumRows, PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::numCols, PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::numRows, and PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::rowHdrs.

Referenced by PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::clone(), and PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::~~SignMatrix().

**9.123.3.10** `template<class RowHdr , class ColHdr , typename  
INTERNAL_TYPE > Col PC::SignMatrix< RowHdr, ColHdr,  
INTERNAL_TYPE >::getCol (unsigned int colNo) const  
[inline]`

Definition at line 240 of file SignMatrix.h.

References PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::colHdrs, PC::Col::colid, and PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::numCols.

Referenced by PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::colHdr(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::insertRowCopy(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::insertZeroRow(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::operator()(), and PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::setEntry().

**9.123.3.11** `template<class RowHdr , class ColHdr , typename  
INTERNAL_TYPE > unsigned int PC::SignMatrix< RowHdr,  
ColHdr, INTERNAL_TYPE >::getColNumber (Col col) const  
[inline]`

Definition at line 247 of file SignMatrix.h.

References PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::colHdrs, PC::Col::colid, PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::maxNumCols, and PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::numCols.



**9.123.3.12** `template<class RowHdr , class ColHdr , typename  
INTERNAL_TYPE > unsigned int PC::SignMatrix< RowHdr,  
ColHdr, INTERNAL_TYPE >::getNumCols () const [inline]`

Definition at line 217 of file SignMatrix.h.

References PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::numCols.

**9.123.3.13** `template<class RowHdr , class ColHdr , typename  
INTERNAL_TYPE > unsigned int PC::SignMatrix< RowHdr,  
ColHdr, INTERNAL_TYPE >::getNumRows () const [inline]`

Methods for getting the size of the matrix.

Definition at line 212 of file SignMatrix.h.

References PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::numRows.

**9.123.3.14** `template<class RowHdr , class ColHdr , typename  
INTERNAL_TYPE > Row PC::SignMatrix< RowHdr, ColHdr,  
INTERNAL_TYPE >::getRow (unsigned int rowNo) const  
[inline]`

Methods for getting the index of a row or column or getting the row or column at a specified index.

Definition at line 227 of file SignMatrix.h.

References PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::numRows, PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::rowHdrs, and PC::Row::rowid.

Referenced by PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::columnsEqual(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::operator(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::rowHdr(), and PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::setEntry().

**9.123.3.15** `template<class RowHdr , class ColHdr , typename  
INTERNAL_TYPE > unsigned int PC::SignMatrix< RowHdr,  
ColHdr, INTERNAL_TYPE >::getRowNumber (Row row) const  
[inline]`

Definition at line 234 of file SignMatrix.h.

References PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::maxNumRows, PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE

>::numRows, PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::rowHdrs, and PC::Row::rowid.

**9.123.3.16** `template<class RowHdr , class ColHdr, typename  
INTERNAL_TYPE > Col PC::SignMatrix< RowHdr, ColHdr,  
INTERNAL_TYPE >::insertCol (unsigned int colNo, ColHdr  
colHdr) [inline, private]`

Definition at line 805 of file SignMatrix.h.

References PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::colHdrs, PC::Col::colid, PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::extend(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::firstUnusedCol, PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::maxNumCols, PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::numCols, and PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::numRows.

Referenced by PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::insertColCopy(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::insertColSum(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::insertIntersectionCol(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::insertInvCol(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::insertUnitCol(), and PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::insertZeroCol().

**9.123.3.17** `template<class RowHdr , class ColHdr , typename  
INTERNAL_TYPE > Col PC::SignMatrix< RowHdr, ColHdr,  
INTERNAL_TYPE >::insertColCopy (Col sourceCol, unsigned int  
targetColNo) [inline]`

Copies the values of a specified column (EXcluding the header) to another (or the same) column. If the latter doesn't exist, it is created. Returns the index of the resulting column.

Definition at line 448 of file SignMatrix.h.

References PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::colHdrs, PC::Col::colid, PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::insertCol(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::INTERNAL\_SIZE, PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::matrix, PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::maxNumCols, PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::maxNumRows, and PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::numCols.

**9.123.3.18** `template<class RowHdr , class ColHdr, typename  
INTERNAL_TYPE > Col PC::SignMatrix< RowHdr, ColHdr,  
INTERNAL_TYPE >::insertColSum (Col sourceCol1, Col  
sourceCol2, unsigned int targetColNo, ColHdr targetColHdr =  
ColHdr ()) [inline]`

Adds the values of two given columns and writes it to another column. If the latter doesn't exist, it is created and returned. Assumes that the two columns can be added without leaving {-1,0,+1} otherwise the behavior of this method is undefined.

Definition at line 427 of file SignMatrix.h.

References PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::colHdrs, PC::Col::colid, PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::insertCol(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::INTERNAL\_SIZE, PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::mask1, PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::mask2, PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::matrix, PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::maxNumCols, PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::maxNumRows, and PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::numCols.

**9.123.3.19** `template<class RowHdr , class ColHdr, typename  
INTERNAL_TYPE > Col PC::SignMatrix< RowHdr, ColHdr,  
INTERNAL_TYPE >::insertIntersectionCol (Col sourceCol1, Col  
sourceCol2, unsigned int targetColNo, ColHdr targetColHdr =  
ColHdr ()) [inline]`

Definition at line 458 of file SignMatrix.h.

References PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::colHdrs, PC::Col::colid, PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::insertCol(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::INTERNAL\_SIZE, PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::matrix, PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::maxNumCols, PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::maxNumRows, and PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::numCols.

**9.123.3.20** `template<class RowHdr , class ColHdr, typename  
INTERNAL_TYPE > Col PC::SignMatrix< RowHdr, ColHdr,  
INTERNAL_TYPE >::insertInvCol (Col sourceCol, unsigned int  
targetColNo, ColHdr targetColHdr = ColHdr ()) [inline]`

Inverts a specified column of the matrix and writes the result to another (or the same) column. If the latter doesn't exist, it is created. Returns the changed or newly created column.

Definition at line 405 of file SignMatrix.h.

References PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::colHdrs, PC::Col::colid, PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::insertCol(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::INTERNAL\_SIZE, PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::mask1, PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::mask2, PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::matrix, PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::maxNumCols, PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::maxNumRows, and PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::numCols.

**9.123.3.21** `template<class RowHdr, class ColHdr, typename INTERNAL_TYPE> void PC::SignMatrix< RowHdr, ColHdr, INTERNAL_TYPE >::insertMatrix (const SignMatrix< RowHdr, ColHdr, INTERNAL_TYPE > & mat, unsigned int rowOffset, unsigned int colOffset) [inline]`

Copies a given matrix into this one as a block starting at the given offset.

Definition at line 684 of file SignMatrix.h.

References PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::colHdrs, PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::extend(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::insertZeroCol(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::insertZeroRow(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::maxNumCols, PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::maxNumRows, PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::numCols, PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::numRows, PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::rowHdrs, and PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::setEntry().

**9.123.3.22** `template<class RowHdr, class ColHdr , typename INTERNAL_TYPE > Row PC::SignMatrix< RowHdr, ColHdr, INTERNAL_TYPE >::insertRow (unsigned int rowNo, RowHdr rowHdr) [inline, private]`

Internal methods whose job it is to find or create unused space for a new row or column and create it. It also adjusts the header to make it appear as if the new row or column was inserted at the given position.

Definition at line 773 of file SignMatrix.h.

References PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::extend(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::firstUnusedRow, PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::maxNumRows,

### 9.123 PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE > Class Template Reference 495

PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::numCols,  
PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::numRows,  
PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::rowHdrs, and  
PC::Row::rowid.

Referenced by PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::insertRowCopy(), and PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::insertZeroRow().

#### 9.123.3.23 **template<class RowHdr , class ColHdr , typename INTERNAL\_TYPE > Row** PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::insertRowCopy (Row *sourceRow*, unsigned int *targetRowNo*) [inline]

Inserts a copy of a given row (including its header) at another position in the matrix.

Definition at line 558 of file SignMatrix.h.

References PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::getCol(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::insertRow(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::maxNumRows, PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::numCols, PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::numRows, PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::rowHdrs, PC::Row::rowid, and PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::setEntry().

#### 9.123.3.24 **template<class RowHdr , class ColHdr, typename INTERNAL\_TYPE > Col** PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::insertUnitCol (unsigned int *colNo*, unsigned int *i*, ColHdr *colHdr* = ColHdr()) [inline]

Writes the i-th unit vector to a given column. If the column with the specified index does not yet exist, it is created. Returns the changed or newly created column.

Definition at line 388 of file SignMatrix.h.

References PC::Col::colid, PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::insertCol(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::INTERNAL\_SIZE, PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::matrix, PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::maxNumRows, PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::numRows, PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::rowHdrs, PC::Row::rowid, and PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::setEntry().

**9.123.3.25** `template<class RowHdr , class ColHdr, typename  
INTERNAL_TYPE > Col PC::SignMatrix< RowHdr, ColHdr,  
INTERNAL_TYPE >::insertZeroCol (unsigned int colNo, ColHdr  
colHdr = ColHdr ()) [inline]`

Writes zeros to a given column. If the column with the specified index does not yet exist, it is created. Returns the changed or newly created column.

Definition at line 374 of file SignMatrix.h.

References PC::Col::colid, PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::insertCol(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::INTERNAL\_SIZE, PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::matrix, and PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::maxNumRows.

Referenced by PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::insertMatrix().

**9.123.3.26** `template<class RowHdr, class ColHdr , typename  
INTERNAL_TYPE > Row PC::SignMatrix< RowHdr, ColHdr,  
INTERNAL_TYPE >::insertZeroRow (unsigned int rowNo,  
RowHdr rowHdr = RowHdr ()) [inline]`

Fills a specified row with zeros after creating it if necessary.

Definition at line 545 of file SignMatrix.h.

References PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::getCol(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::insertRow(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::numCols, and PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::setEntry().

Referenced by PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::insertMatrix().

**9.123.3.27** `template<class RowHdr , class ColHdr , typename  
INTERNAL_TYPE > void PC::SignMatrix< RowHdr, ColHdr,  
INTERNAL_TYPE >::moveCol (Col col, unsigned int newColNo)  
[inline]`

Move a column including its header to a new position in the matrix.

Definition at line 514 of file SignMatrix.h.

References PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::colHdrs, PC::Col::colid, PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::maxNumCols, and PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::matrix.

>::numCols.

**9.123.3.28** `template<class RowHdr , class ColHdr , typename  
INTERNAL_TYPE > void PC::SignMatrix< RowHdr, ColHdr,  
INTERNAL_TYPE >::moveRow (Row row, unsigned int  
newRowNo) [inline]`

Move a row including its header to a new position in the matrix.

Definition at line 592 of file SignMatrix.h.

References PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE  
>::maxNumRows, PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE  
>::numRows, PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::rowHdrs,  
and PC::Row::rowid.

**9.123.3.29** `template<class RowHdr , class ColHdr , typename  
INTERNAL_TYPE > Sign PC::SignMatrix< RowHdr, ColHdr,  
INTERNAL_TYPE >::operator() (unsigned int rowNo, Col col)  
const [inline]`

Definition at line 277 of file SignMatrix.h.

References PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::getRow(), and  
PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::operator()().

**9.123.3.30** `template<class RowHdr , class ColHdr , typename  
INTERNAL_TYPE > Sign PC::SignMatrix< RowHdr, ColHdr,  
INTERNAL_TYPE >::operator() (Row row, unsigned int colNo)  
const [inline]`

Definition at line 272 of file SignMatrix.h.

References PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::getCol(), and  
PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::operator()().

**9.123.3.31** `template<class RowHdr , class ColHdr , typename  
INTERNAL_TYPE > Sign PC::SignMatrix< RowHdr, ColHdr,  
INTERNAL_TYPE >::operator() (Row row, Col col) const  
[inline]`

Operators giving access to the entries of the matrix. Index bounds are checked in debug mode only.

Definition at line 258 of file SignMatrix.h.

References PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::colHdrs, PC::Col::colid, PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::INTERNAL\_SIZE, PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::matrix, PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::maxNumCols, PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::maxNumRows, PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::numCols, PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::numRows, PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::rowHdrs, and PC::Row::rowid.

**9.123.3.32** `template<class RowHdr , class ColHdr , typename INTERNAL_TYPE > Sign PC::SignMatrix< RowHdr, ColHdr, INTERNAL_TYPE >::operator() (unsigned int rowNo, unsigned int colNo) const [inline]`

Definition at line 265 of file SignMatrix.h.

References PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::colHdrs, PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::INTERNAL\_SIZE, PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::matrix, PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::maxNumRows, PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::numCols, PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::numRows, and PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::rowHdrs.

Referenced by PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::operator(), and PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::printMatrix().

**9.123.3.33** `template<class RowHdr , class ColHdr , typename INTERNAL_TYPE > void PC::SignMatrix< RowHdr, ColHdr, INTERNAL_TYPE >::printMatrix (std::ostream & os = std::cout) const [inline]`

Prints the matrix in a (semi-)human-readable format to a given ostream.

Definition at line 341 of file SignMatrix.h.

References PC::MINUS, PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::numCols, PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::numRows, PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::operator(), and PC::PLUS.



**9.123.3.34** `template<class RowHdr , class ColHdr , typename  
INTERNAL_TYPE > RowHdr & PC::SignMatrix< RowHdr,  
ColHdr, INTERNAL_TYPE >::rowHdr (Row col) [inline]`

Definition at line 319 of file SignMatrix.h.

References PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::maxNumRows, PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::numRows, PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::rowHdrs, and PC::Row::rowid.

**9.123.3.35** `template<class RowHdr , class ColHdr , typename  
INTERNAL_TYPE > RowHdr & PC::SignMatrix< RowHdr,  
ColHdr, INTERNAL_TYPE >::rowHdr (unsigned int rowNo)  
[inline]`

Operators giving access to the headers of the matrix. Index bounds are checked in debug mode only.

Definition at line 313 of file SignMatrix.h.

References PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::getRow(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::numRows, and PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::rowHdrs.

**9.123.3.36** `template<class RowHdr , class ColHdr , typename  
INTERNAL_TYPE > void PC::SignMatrix< RowHdr, ColHdr,  
INTERNAL_TYPE >::setEntry (unsigned int rowNo, Col col, Sign  
val) [inline]`

Definition at line 303 of file SignMatrix.h.

References PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::getRow(), and PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::setEntry().

**9.123.3.37** `template<class RowHdr , class ColHdr , typename  
INTERNAL_TYPE > void PC::SignMatrix< RowHdr, ColHdr,  
INTERNAL_TYPE >::setEntry (Row row, unsigned int colNo,  
Sign val) [inline]`

Definition at line 298 of file SignMatrix.h.

References PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::getCol(), and PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::setEntry().

**9.123.3.38** `template<class RowHdr , class ColHdr , typename  
INTERNAL_TYPE > void PC::SignMatrix< RowHdr, ColHdr,  
INTERNAL_TYPE >::setEntry (Row row, Col col, Sign val)  
[inline]`

Definition at line 282 of file SignMatrix.h.

References PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::colHdrs, PC::Col::colid, PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::INTERNAL\_SIZE, PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::matrix, PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::maxNumCols, PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::maxNumRows, PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::numCols, PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::numRows, PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::rowHdrs, and PC::Row::rowid.

**9.123.3.39** `template<class RowHdr , class ColHdr , typename  
INTERNAL_TYPE > void PC::SignMatrix< RowHdr, ColHdr,  
INTERNAL_TYPE >::setEntry (unsigned int rowNo, unsigned int  
colNo, Sign val) [inline]`

Definition at line 290 of file SignMatrix.h.

References PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::colHdrs, PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::INTERNAL\_SIZE, PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::matrix, PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::maxNumRows, PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::numCols, PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::numRows, and PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::rowHdrs.

Referenced by PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::insertMatrix(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::insertRowCopy(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::insertUnitCol(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::insertZeroRow(), and PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::setEntry().

## 9.123.4 Member Data Documentation

**9.123.4.1** `template<class RowHdr, class ColHdr, typename  
INTERNAL_TYPE> std::vector<intColHdr> PC::SignMatrix<  
RowHdr, ColHdr, INTERNAL_TYPE >::colHdrs [private]`

Definition at line 42 of file SignMatrix.h.

### 9.123 PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE > Class Template Reference 501

Referenced by PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::addToCol(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::clone(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::colHdr(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::columnsEqual(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::deleteCol(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::extend(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::free(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::getCol(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::getColNumber(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::insertCol(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::insertColCopy(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::insertColSum(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::insertIntersectionCol(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::insertInvCol(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::insertMatrix(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::moveCol(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::operator()(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::setEntry(), and PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::SignMatrix().

**9.123.4.2** `template<class RowHdr, class ColHdr, typename INTERNAL_TYPE> const double PC::SignMatrix< RowHdr, ColHdr, INTERNAL_TYPE >::EXTEND_CONDITION = 1.2 [static, private]`

Definition at line 44 of file SignMatrix.h.

Referenced by PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::extend().

**9.123.4.3** `template<class RowHdr, class ColHdr, typename INTERNAL_TYPE> const double PC::SignMatrix< RowHdr, ColHdr, INTERNAL_TYPE >::EXTEND_FACTOR = 2.0 [static, private]`

Definition at line 45 of file SignMatrix.h.

Referenced by PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::extend().

**9.123.4.4** `template<class RowHdr, class ColHdr, typename INTERNAL_TYPE> int PC::SignMatrix< RowHdr, ColHdr, INTERNAL_TYPE >::firstUnusedCol [private]`

Definition at line 36 of file SignMatrix.h.

Referenced by PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::clone(),

```
PC::SignMatrix< RowHdr, ColHdr, INTERNAL_TYPE >::deleteCol(),
PC::SignMatrix< RowHdr, ColHdr, INTERNAL_TYPE >::extend(),
PC::SignMatrix< RowHdr, ColHdr, INTERNAL_TYPE >::insertCol(), and
PC::SignMatrix< RowHdr, ColHdr, INTERNAL_TYPE >::SignMatrix().
```

#### 9.123.4.5 **template<class RowHdr, class ColHdr, typename INTERNAL\_TYPE> int PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::firstUnusedRow [private]**

Definition at line 35 of file SignMatrix.h.

Referenced by PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::clone(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::deleteRow(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::extend(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::insertRow(), and PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::SignMatrix().

#### 9.123.4.6 **template<class RowHdr, class ColHdr, typename INTERNAL\_TYPE> const unsigned int PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::INTERNAL\_SIZE [private]**

Definition at line 46 of file SignMatrix.h.

Referenced by PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::addToCol(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::clone(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::extend(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::insertColCopy(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::insertColSum(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::insertIntersectionCol(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::insertInvCol(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::insertUnitCol(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::insertZeroCol(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::operator(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::setEntry(), and PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::SignMatrix().

#### 9.123.4.7 **template<class RowHdr, class ColHdr, typename INTERNAL\_TYPE> INTERNAL\_TYPE PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::mask1 [private]**

Definition at line 47 of file SignMatrix.h.

Referenced by PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::addToCol(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::insertColSum(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE

### 9.123 PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE > Class Template Reference 503

---

>::insertInvCol(), and PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::SignMatrix().

**9.123.4.8** `template<class RowHdr, class ColHdr, typename INTERNAL_TYPE> INTERNAL_TYPE PC::SignMatrix< RowHdr, ColHdr, INTERNAL_TYPE >::mask2 [private]`

Definition at line 47 of file SignMatrix.h.

Referenced by PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::addToCol(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::insertColSum(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::insertInvCol(), and PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::SignMatrix().

**9.123.4.9** `template<class RowHdr, class ColHdr, typename INTERNAL_TYPE> INTERNAL_TYPE* PC::SignMatrix< RowHdr, ColHdr, INTERNAL_TYPE >::matrix [private]`

Definition at line 38 of file SignMatrix.h.

Referenced by PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::addToCol(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::clone(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::extend(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::free(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::insertColCopy(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::insertColSum(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::insertIntersectionCol(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::insertInvCol(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::insertUnitCol(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::insertZeroCol(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::operator(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::setEntry(), and PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::SignMatrix().

**9.123.4.10** `template<class RowHdr, class ColHdr, typename INTERNAL_TYPE> unsigned int PC::SignMatrix< RowHdr, ColHdr, INTERNAL_TYPE >::maxNumCols [private]`

Definition at line 34 of file SignMatrix.h.

Referenced by PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::addToCol(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::clone(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::colHdr(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::columnsEqual(),

PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::deleteCol(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::extend(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::free(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::getColNumber(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::insertCol(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::insertColCopy(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::insertColSum(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::insertIntersectionCol(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::insertInvCol(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::insertMatrix(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::moveCol(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::operator(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::setEntry(), and PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::SignMatrix().

#### 9.123.4.11 **template<class RowHdr, class ColHdr, typename INTERNAL\_TYPE> unsigned int PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::maxNumRows [private]**

Definition at line 34 of file SignMatrix.h.

Referenced by PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::addToCol(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::clone(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::deleteRow(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::extend(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::free(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::getRowNumber(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::insertColCopy(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::insertColSum(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::insertIntersectionCol(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::insertInvCol(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::insertMatrix(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::insertRow(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::insertRowCopy(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::insertUnitCol(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::insertZeroCol(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::moveRow(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::operator(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::rowHdr(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::setEntry(), and PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::SignMatrix().

**9.123.4.12   template<class RowHdr, class ColHdr, typename  
INTERNAL\_TYPE> unsigned int PC::SignMatrix< RowHdr,  
ColHdr, INTERNAL\_TYPE >::numCols   [private]**

Definition at line 33 of file SignMatrix.h.

Referenced by PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::addToCol(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::clone(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::colHdr(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::columnsEqual(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::deleteCol(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::extend(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::free(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::getCol(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::getColNumber(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::getNumCols(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::insertCol(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::insertColCopy(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::insertColSum(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::insertIntersectionCol(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::insertInvCol(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::insertMatrix(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::insertRow(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::insertRowCopy(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::insertZeroRow(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::moveCol(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::operator()(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::printMatrix(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::setEntry(), and PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::SignMatrix().

**9.123.4.13   template<class RowHdr, class ColHdr, typename  
INTERNAL\_TYPE> unsigned int PC::SignMatrix< RowHdr,  
ColHdr, INTERNAL\_TYPE >::numRows   [private]**

Definition at line 33 of file SignMatrix.h.

Referenced by PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::clone(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::columnsEqual(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::deleteRow(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::free(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::getNumRows(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::getRow(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::getRowNumber(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::insertCol(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::insertMatrix(), PC::SignMatrix< RowHdr, Col-

Hdr, INTERNAL\_TYPE >::insertRow(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::insertRowCopy(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::insertUnitCol(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::moveRow(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::operator(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::printMatrix(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::rowHdr(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::setEntry(), and PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::SignMatrix().

#### 9.123.4.14 **template<class RowHdr, class ColHdr, typename INTERNAL\_TYPE> std::vector<intRowHdr> PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::rowHdrs [private]**

Definition at line 41 of file SignMatrix.h.

Referenced by PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::clone(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::deleteRow(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::extend(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::free(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::getRow(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::getRowNumber(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::insertMatrix(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::insertRow(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::insertRowCopy(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::insertUnitCol(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::moveRow(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::operator(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::rowHdr(), PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::setEntry(), and PC::SignMatrix< RowHdr, ColHdr, INTERNAL\_TYPE >::SignMatrix().

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

- HigmanGroup/include/**SignMatrix.h**



## 9.124 StraightLineProgramWord Class Reference

class **StraightLineProgramWord** (p. 507).

```
#include <StraightLineProgramWord.h>
```

### Classes

- struct **Assertion**  
*Structure used in function `equal()` (p. 512) only.*
- struct **Production**  
*A production for a rule of a composition system.*

### Public Member Functions

- **StraightLineProgramWord** ()  
*Default constructor.*
- **StraightLineProgramWord** (int init, int gens)  
*Create a composition system producing a 1-generator word 'init'.*
- template<class ConstMapIterator >  
**StraightLineProgramWord** (int init, ConstMapIterator B, ConstMapIterator E)  
*Construct a composition system for a generator init acted on by a sequence of mappings.*
- **StraightLineProgramWord & operator=** (const **StraightLineProgramWord** &SLP)  
*Assignement operator.*
- int **operator[]** (LongInteger i) const  
*Index operator. Obtain a generator at the ith position in the word.*
- **StraightLineProgramWord operator-** () const  
*Compute a composition system which produces the inverse of the current word.*
- **StraightLineProgramWord operator\*** (const **StraightLineProgramWord** &SLP) const  
*Compute an SLP which produces a product of words produced by \*this and SLP (as semigroup elements).*

- **bool equal** (int n1, const **StraightLineProgramWord** &SLP, int n2) const  
*Check if words produced by  $n_1$  in \*this and  $n_2$  in SLP are the same (as elements of semigroup).*
- **int terminalSegment** (int n, **LongInteger** length)  
*Add a vertex  $n'$  into SLP (if required) which represents the word  $w$  which the terminal segment of the word represented by  $n$  of required length.*
- **int initialSegment** (int n, **LongInteger** length)  
*Add a vertex  $n'$  into SLP (if required) which represents the word  $w$  which the initial segment of the word represented by  $n$  of required length.*
- **int truncateVertexLeft** (int n, **LongInteger** length)  
*Add a vertex  $n'$  into SLP (if required) which represents the word  $w_n$  truncated on the left.*
- **int truncateVertexRight** (int n, **LongInteger** length)  
*Add a vertex  $n'$  into SLP (if required) which represents the word  $w_n$  truncated on the left.*
- **LongInteger leftGCDLength** (const **StraightLineProgramWord** &CS) const  
*Compute the length of the longest common initial segment of 2 words given by SLPs (as elements of semigroup).*
- **Word getWord** () const  
*Get a word represented (produced) by the SLP.*
- **Word getWord** (int N) const  
*Get a word represented (produced) by (non-)terminal symbol.*
- **LongInteger length** () const  
*Compute the length of the word produced by the SLP.*
- **void reduce** ()  
*Reduce the SLP.*
- **void simplify** ()  
*Simplify the structure of SLP.*

## Private Member Functions

- void **order\_vertices** (int n, list< int > &order, set< int > &closure) const  
*Order the descendants of n.*
- void **order\_vertices** (int n, list< int > &order) const
- **LongInteger length\_rule** (int n) const  
*Compute the length of the word corresponding to a rule.*
- int **height\_rule** (int n) const  
*Compute the height of the rule.*
- bool **assertionType** (const **Assertion** &A, const **StraightLineProgramWord** &SLP) const  
*Determine the type of the assertion.*
- void **extendTerminals** (int nt)  
*Extend the set of non-terminals. The word produced by the result is the same as the original.*
- void **reduce\_rule** (int n)  
*Reduce a subgraph of the SLP reachable from the vertex n.*
- **LongInteger leftGCDLength** (int n1, const **StraightLineProgramWord** &CS, int n2) const  
*Compute the length of the longest common initial segment of 2 words given by composition systems.*
- set< **Assertion** > **splitAssertion** (const **Assertion** &A, bool firstTerm, const **StraightLineProgramWord** &SLP) const
- void **update\_lengths** ()  
*Update the lengths of all rules in the system.*
- void **update\_length** (int n)  
*Recursively update the length of the rule.*
- void **update\_heights** ()  
*Update the heights of all rules in the system.*
- void **update\_height** (int n)  
*Recursively update the height of the rule.*
- int **getGenerator** (int n, **LongInteger** pos) const

*Get a generator at  $i$ th position of the word  $w_n$ .*

- **LongInteger cancellationLength** (int n1, int n2) const

*Find the amount of cancellation in a product  $w_{n_1} w_{n_2}$ .*

- **StraightLineProgramWord applyMapping** (const Map &M) const

*For a composition system  $C$  and a mapping  $\alpha$  compute a composition system representing  $w_C^\alpha$ .*

- int **max\_rule\_number** () const

*Get the maximal number of the rule in SLP.*

## Static Private Member Functions

- static void **invertProductionPair** (int &A, int &B)

*Function inverts a production pair (A,B).*

- static pair< map< int, **Production** >, vector< int > > **map\_rules** (const Map &M)

*Construct the composition system rules for the map M.*

## Private Attributes

- int **theRoot**

*The number of the root non-terminal. Theroot==0 if the system is empty (default constructor).*

- int **theTerminals**

*The number of terminals in the system.*

- map< int, **Production** > **theRules**

*The production rules of the system.*

## Friends

- ostream & **operator<<** (ostream &os, const **StraightLineProgramWord** &CS)

### 9.124.1 Detailed Description

class **StraightLineProgramWord** (p. 507). A straight line program is basically a program computing a single word. We used "Polynomial-time Word problems" by Saul Schleimer when implementing this class.

In this implementation the class **StraightLineProgramWord** (p. 507) serves 1 purpose - it is a special container class for words. In other words it is just another representation for words, used mainly to efficiently solve the **Word** (p. 642) problem for the automorphism group of a free group.

Notice that we do not reduce SLP when constructing, i.e., initially a word represented by SLP is not reduced and **length()** (p. 514) is not the same as the length of **getWord()** (p. 513). You need to use **reduce()** (p. 516) for that.

Another thing to remember... I do not assume that for all rules  $x_i \rightarrow x_j x_k$   $i > j, k$ . Though it is assumed that  $x_j \neq 1$ . In a situation when  $x_j = 1$ ,  $x_k = 1$   $x_i$  must be removed from the system.

After all is done need to check that all those assumptions are really satisfied.

Definition at line 52 of file StraightLineProgramWord.h.

### 9.124.2 Constructor & Destructor Documentation

#### 9.124.2.1 StraightLineProgramWord::StraightLineProgramWord () [inline]

Default constructor.

Definition at line 140 of file StraightLineProgramWord.h.

#### 9.124.2.2 StraightLineProgramWord::StraightLineProgramWord (int *init*, int *gens*)

Create a composition system producing a 1-generator word 'init'.

#### 9.124.2.3 template<class ConstMapIterator > StraightLineProgramWord::StraightLineProgramWord (int *init*, ConstMapIterator *B*, ConstMapIterator *E*) [inline]

Construct a composition system for a generator *init* acted on by a sequence of mappings. Here if  $M_1, \dots, M_n$  is a sequence of mappings then the action of a sequence is a sequence of actions where  $M_1$  is the first mapping to apply, and so on up to  $M_n$ .

Definition at line 154 of file StraightLineProgramWord.h.

References **applyMapping()**, **theRoot**, **theRules**, and **theTerminals**.

### 9.124.3 Member Function Documentation

#### 9.124.3.1 **StraightLineProgramWord StraightLineProgramWord::applyMapping (const Map & *M*) const** **[private]**

For a composition system  $C$  and a mapping  $\alpha$  compute a composition system representing  $w_C^\alpha$ . The result is computed in a very straightforward way. There is no check that  $\alpha$  is applicable. In case when the domain alphabet of  $\alpha$  is strictly smaller than the number of terminals the output is an empty composition sequence. In case when the domain alphabet of  $\alpha$  is strictly greater than the number of terminals the result is a correct composition sequence.

Referenced by StraightLineProgramWord().

#### 9.124.3.2 **bool StraightLineProgramWord::assertionType (const Assertion & *A*, const StraightLineProgramWord & *SLP*) const** **[private]**

Determine the type of the assertion. The function determines the type of the assertion. The output==true if *A* is an overlap assertion. The output==false if *A* is an subword assertion.

#### 9.124.3.3 **LongInteger StraightLineProgramWord::cancellationLength (int *n1*, int *n2*) const** **[private]**

Find the amount of cancellation in a product  $w_{n_1}w_{n_2}$ .

#### 9.124.3.4 **bool StraightLineProgramWord::equal (int *n1*, const StraightLineProgramWord & *SLP*, int *n2*) const**

Check if words produced by  $n_1$  in \*this and  $n_2$  in *SLP* are the same (as elements of semigroup).

#### 9.124.3.5 **void StraightLineProgramWord::extendTerminals (int *nt*)** **[private]**

Extend the set of non-terminals. The word produced by the result is the same as the original.

#### 9.124.3.6 **int StraightLineProgramWord::getGenerator (int *n*, LongInteger *pos*) const** **[private]**

Get a generator at *ith* position of the word  $w_n$ .

Referenced by operator[]().

#### 9.124.3.7 Word StraightLineProgramWord::getWord (int $N$ ) const

Get a word represented (produced) by (non-)terminal symbol. The current implementation is the most straightforward approach (not efficient). Notice that  $N$  can be negative.

#### 9.124.3.8 Word StraightLineProgramWord::getWord () const [inline]

Get a word represented (produced) by the SLP. Be careful!!! Relatively small SLPs can represent huge words.

Definition at line 244 of file StraightLineProgramWord.h.

References getWord(), and theRoot.

Referenced by getWord().

#### 9.124.3.9 int StraightLineProgramWord::height\_rule (int $n$ ) const [inline, private]

Compute the height of the rule.

Definition at line 303 of file StraightLineProgramWord.h.

References theRules, and theTerminals.

#### 9.124.3.10 int StraightLineProgramWord::initialSegment (int $n$ , LongInteger $length$ ) [inline]

Add a vertex  $n'$  into SLP (if required) which represents the word  $w$  which the initial segment of the word represented by  $n$  of required length.

Definition at line 213 of file StraightLineProgramWord.h.

References length\_rule(), and truncateVertexRight().

#### 9.124.3.11 static void StraightLineProgramWord::invertProductionPair (int & $A$ , int & $B$ ) [static, private]

Function inverts a production pair (A,B).

**9.124.3.12** `LongInteger StraightLineProgramWord::leftGCDLength (int n1,  
const StraightLineProgramWord & CS, int n2) const` [**private**]

Compute the length of the longest common initial segment of 2 words given by composition systems.

**9.124.3.13** `LongInteger StraightLineProgramWord::leftGCDLength (const  
StraightLineProgramWord & CS) const` [**inline**]

Compute the length of the longest common initial segment of 2 words given by SLPs (as elements of semigroup).

Definition at line 235 of file StraightLineProgramWord.h.

References theRoot.

**9.124.3.14** `LongInteger StraightLineProgramWord::length () const`  
[**inline**]

Compute the length of the word produced by the SLP.

Definition at line 256 of file StraightLineProgramWord.h.

References length\_rule(), and theRoot.

**9.124.3.15** `LongInteger StraightLineProgramWord::length_rule (int n) const`  
[**inline, private**]

Compute the length of the word corresponding to a rule.

Definition at line 293 of file StraightLineProgramWord.h.

References theRules, and theTerminals.

Referenced by initialSegment(), length(), and terminalSegment().

**9.124.3.16** `static pair< map< int , Production > , vector< int > >  
StraightLineProgramWord::map_rules (const Map & M)`  
[**static, private**]

Construct the composition system rules for the map *M*. Function prepares a block of rules for the given map. The range terminals have the lowest numbers  $1, \dots, t$ . The domain non-terminals are given in the vector.



**9.124.3.17** `int StraightLineProgramWord::max_rule_number () const`  
`[private]`

Get the maximal number of the rule in SLP.

**9.124.3.18** `StraightLineProgramWord StraightLineProgramWord::operator*`  
`(const StraightLineProgramWord & SLP) const`

Compute an SLP which produces a product of words produced by \*this and SLP (as semigroup elements).

**9.124.3.19** `StraightLineProgramWord StraightLineProgramWord::operator-`  
`const`

Compute a composition system which produces the inverse of the current word.

**9.124.3.20** `StraightLineProgramWord& StraightLineProgram-`  
`Word::operator= (const StraightLineProgramWord & SLP)`  
`[inline]`

Assignment operator.

Definition at line 173 of file StraightLineProgramWord.h.

References theRoot, theRules, and theTerminals.

**9.124.3.21** `int StraightLineProgramWord::operator[] (LongInteger i) const`  
`[inline]`

Index operator. Obtain a generator at the ith position in the word.

Definition at line 181 of file StraightLineProgramWord.h.

References getGenerator(), and theRoot.

**9.124.3.22** `void StraightLineProgramWord::order_vertices (int n, list< int >`  
`& order) const [inline, private]`

Definition at line 286 of file StraightLineProgramWord.h.

References order\_vertices().

**9.124.3.23** `void StraightLineProgramWord::order_vertices (int  $n$ , list< int > & order, set< int > & closure) const` [**private**]

Order the descendants of  $n$ . Operation returns 2 structures: order - contains the ordered set of descendants and closure which contains the same vertices ordered as integers. It is assumed that order and closure are initially empty.

Referenced by order\_vertices().

**9.124.3.24** `void StraightLineProgramWord::reduce ()` [**inline**]

Reduce the SLP.

Definition at line 260 of file StraightLineProgramWord.h.

References reduce\_rule(), and theRoot.

**9.124.3.25** `void StraightLineProgramWord::reduce_rule (int  $n$ )` [**private**]

Reduce a subgraph of the SLP reachable from the vertex  $n$ .

Referenced by reduce().

**9.124.3.26** `void StraightLineProgramWord::simplify ()`

Simplify the structure of SLP. Update vertices of out\_degree 1 and remove all vertices that can not be reached from the root.

**9.124.3.27** `set< Assertion > StraightLineProgramWord::splitAssertion (const Assertion & A, bool firstTerm, const StraightLineProgramWord & SLP) const` [**private**]

**9.124.3.28** `int StraightLineProgramWord::terminalSegment (int  $n$ , LongInteger length)` [**inline**]

Add a vertex  $n'$  into SLP (if required) which represents the word  $w$  which the terminal segment of the word represented by  $n$  of required length.

Definition at line 207 of file StraightLineProgramWord.h.

References length\_rule(), and truncateVertexLeft().

### 9.124.3.29 `int StraightLineProgramWord::truncateVertexLeft (int $n$ , LongInteger $length$ )`

Add a vertex  $n'$  into SLP (if required) which represents the word  $w_n$  truncated on the left. If  $w_n = u \circ v$ ,  $length = |u|$ , and  $t$  is the output of this function then  $w_t = v$ . The vertex  $n$  can be negative. Foolproof.

Referenced by `terminalSegment()`.

### 9.124.3.30 `int StraightLineProgramWord::truncateVertexRight (int $n$ , LongInteger $length$ )`

Add a vertex  $n'$  into SLP (if required) which represents the word  $w_n$  truncated on the right. If  $w_n = u \circ v$ ,  $length = |v|$ , and  $t$  is the output of this function then  $w_t = u$ . The vertex  $n$  can be negative. Foolproof.

Referenced by `initialSegment()`.

### 9.124.3.31 `void StraightLineProgramWord::update_height (int $n$ ) [private]`

Recursively update the height of the rule.

### 9.124.3.32 `void StraightLineProgramWord::update_heights () [private]`

Update the heights of all rules in the system.

### 9.124.3.33 `void StraightLineProgramWord::update_length (int $n$ ) [private]`

Recursively update the length of the rule.

### 9.124.3.34 `void StraightLineProgramWord::update_lengths () [private]`

Update the lengths of all rules in the system.

## 9.124.4 Friends And Related Function Documentation

**9.124.4.1** `ostream& operator<< (ostream & os, const  
StraightLineProgramWord & CS) [friend]`

## 9.124.5 Member Data Documentation

**9.124.5.1** `int StraightLineProgramWord::theRoot [private]`

The number of the root non-terminal. Theroot==0 if the system is empty (default constructor). The root is always non-negative!!!

Definition at line 407 of file StraightLineProgramWord.h.

Referenced by `getWord()`, `leftGCDLength()`, `length()`, `operator=()`, `operator[]()`, `reduce()`, and `StraightLineProgramWord()`.

**9.124.5.2** `map< int , Production > StraightLineProgramWord::theRules  
[private]`

The production rules of the system.

Definition at line 415 of file StraightLineProgramWord.h.

Referenced by `height_rule()`, `length_rule()`, `operator=()`, and `StraightLineProgramWord()`.

**9.124.5.3** `int StraightLineProgramWord::theTerminals [private]`

The number of terminals in the system.

Definition at line 411 of file StraightLineProgramWord.h.

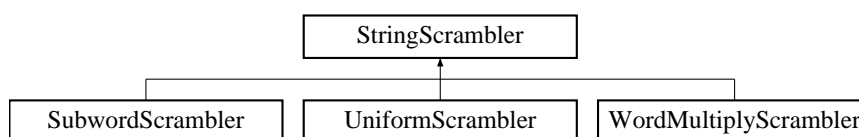
Referenced by `height_rule()`, `length_rule()`, `operator=()`, and `StraightLineProgramWord()`.

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

- `FreeGroup/include/StraightLineProgramWord.h`

## 9.125 StringScrambler Class Reference

#include <StringScramblers.h> Inheritance diagram for StringScrambler::



### Public Member Functions

- virtual **Word** **scramble** (const **Word** &) const =0
- virtual double **fracChanged** () const =0

#### 9.125.1 Detailed Description

Definition at line 26 of file StringScramblers.h.

#### 9.125.2 Member Function Documentation

##### 9.125.2.1 virtual double StringScrambler::fracChanged () const [pure virtual]

Implemented in **UniformScrambler** (p. 615), **SubwordScrambler** (p. 533), and **WordMultiplyScrambler** (p. 666).

##### 9.125.2.2 virtual Word StringScrambler::scramble (const Word &) const [pure virtual]

Implemented in **UniformScrambler** (p. 616), **SubwordScrambler** (p. 534), and **WordMultiplyScrambler** (p. 667).

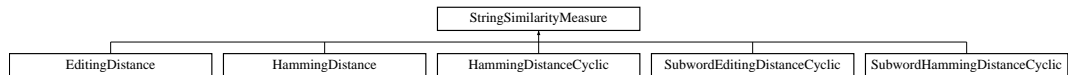
The documentation for this class was generated from the following file:

- StringSimilarity/include/StringScramblers.h

## 9.126 StringSimilarityMeasure Class Reference

Abstract interface for a class implementing string (**Word** (p. 642)) similarity measure.

#include <SimilarityMeasures.h> Inheritance diagram for StringSimilarityMeasure::



### Public Member Functions

- virtual double **measure** (const **Word** &w1, const **Word** &w2) const =0

*Returns a distance (measure) between two words.*

### 9.126.1 Detailed Description

Abstract interface for a class implementing string (**Word** (p. 642)) similarity measure.

Definition at line 87 of file SimilarityMeasures.h.

### 9.126.2 Member Function Documentation

#### 9.126.2.1 virtual double StringSimilarityMeasure::measure (const **Word** & w1, const **Word** & w2) const [pure virtual]

Returns a distance (measure) between two words.

#### Parameters:

- w1** - the first word
- w2** - the second word

#### Returns:

the distance. It is assumed that greater values indicate less similarities.

Implemented in **HammingDistance** (p. 268), **HammingDistanceCyclic** (p. 270), **SubwordHammingDistanceCyclic** (p. 532), **EditingDistance** (p. 193), and **SubwordEditingDistanceCyclic** (p. 530).

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

- `StringSimilarity/include/SimilarityMeasures.h`

## 9.127 SubgroupFG Class Reference

```
#include <SubgroupFG.h>
```

### Public Member Functions

- **SubgroupFG** (int n\_gens=0)
- **SubgroupFG** (int n\_gens, const vector< **Word** > &gens)
- template<class ConstWordIterator >  
**SubgroupFG** (int n\_gens, ConstWordIterator B, ConstWordIterator E)
- bool **operator==** (const **SubgroupFG** &S) const  
*Check the equality of two subgroups.*
- **SubgroupFG operator\*** (const **SubgroupFG** &S) const  
*Compute the intersection of two subgroups.*
- **SubgroupFG & operator^=** (const **Word** &conjugator)  
*Conjugate a subgroup.*
- **SubgroupFG operator^** (const **Word** &conjugator) const  
*Conjugate a subgroup.*
- **SubgroupFG & operator+=** (const **SubgroupFG** &sbgp)  
*Extend subgroup basis.*
- **SubgroupFG operator+** (const **SubgroupFG** &sbgp) const  
*Extend subgroup basis.*
- **SubgroupFG & operator+=** (const **Word** &w)  
*Extend subgroup basis.*
- **SubgroupFG operator+** (const **Word** &w) const  
*Extend subgroup basis.*
- bool **checkIsomorphism** (const **SubgroupFG** &S, int vert1, int vert2) const  
*Check if two subgroup graphs \*this and S are isomorphic as automata, restricted that v1 -> v2.*
- const vector< **Word** > & **getGenerators** () const  
*Get the initial generators of the subgroup.*
- const vector< **Word** > & **getNielsenGenerators** () const



*Get a set of Nielsen generators of the subgroup.*

- **const IntLabeledGraph & getFSA () const**  
*Get an automaton corresponding to a subgroup of a free group.*
- **int getIndex () const**  
*Compute the index of a subgroup of a free group, -1 means infinite.*
- **int getRank () const**  
*Compute the rank of a subgroup of a free group.*
- **bool doesBelong (const Word &w) const**  
*check if a word belongs to a subgroup of a free group*
- **Word express (const Word &w) const**  
*Express a word in a generators of a subgroup of a free group (word must belong to subgroup).*
- **SubgroupFG centralizer () const**  
*Compute the centralizer of a subgroup.*
- **SubgroupFG normalizer () const**  
*Compute the normalizer of a subgroup.*
- **pair< SubgroupFG, Word > trim () const**  
*Trim the subgroup graph. (Cut off the "tail").*
- **pair< bool, Word > areConjugate (const SubgroupFG &sbgrp) const**  
*Check if two subgroups are conjugate.*
- **string graphviz\_format () const**  
*Returns the graphviz graphical description of the subgroup graph (save it to file and use Graphviz to visualize the graph).*

## Protected Member Functions

- **SubgroupFG (int n\_gens, const IntLabeledGraph &fsa)**

## Private Member Functions

- **void computeFSA () const**
- **void computeNielsenGenerators () const**

## Private Attributes

- vector< Word > **theGenerators**
- int **theNumberOfGenerators**
- bool **fsaDone**
- **IntLabeledGraph** **theFSA**
- list< FoldDetails< IntLabeledGraph::vertex\_type, IntLabeledGraph::edge\_type > > **foldDetails**
- bool **nielsDone**
- vector< Word > **theNielsenGenerators**

### 9.127.1 Detailed Description

Definition at line 29 of file SubgroupFG.h.

### 9.127.2 Constructor & Destructor Documentation

**9.127.2.1 SubgroupFG::SubgroupFG (int *n\_gens* = 0)**

**9.127.2.2 SubgroupFG::SubgroupFG (int *n\_gens*, const vector< Word > & *gens*)**

**9.127.2.3 template<class ConstWordIterator > SubgroupFG::SubgroupFG (int *n\_gens*, ConstWordIterator *B*, ConstWordIterator *E*) [inline]**

Definition at line 40 of file SubgroupFG.h.

References `theGenerators`.

**9.127.2.4 SubgroupFG::SubgroupFG (int *n\_gens*, const IntLabeledGraph & *fsa*) [protected]**

### 9.127.3 Member Function Documentation

**9.127.3.1 pair< bool , Word > SubgroupFG::areConjugate (const SubgroupFG & *sbgp*) const**

Check if two subgroups are conjugate.

**9.127.3.2 SubgroupFG SubgroupFG::centralizer () const**

Compute the centralizer of a subgroup.

**9.127.3.3** `bool SubgroupFG::checkIsomorphism (const SubgroupFG & S, int  
vert1, int vert2) const`

Check if two subgroup graphs \*this and S are isomorphic as automata, restricted that  $v1 \rightarrow v2$ .

**9.127.3.4** `void SubgroupFG::computeFSA () const [private]`**9.127.3.5** `void SubgroupFG::computeNielsenGenerators () const [private]`**9.127.3.6** `bool SubgroupFG::doesBelong (const Word & w) const`

check if a word belongs to a subgroup of a free group

**9.127.3.7** `Word SubgroupFG::express (const Word & w) const`

Express a word in a generators of a subgroup of a free group (word must belong to subgroup).

**9.127.3.8** `const IntLabeledGraph& SubgroupFG::getFSA () const`

Get an automaton corresponding to a subgroup of a free group.

**9.127.3.9** `const vector< Word >& SubgroupFG::getGenerators () const  
[inline]`

Get the initial generators of the subgroup.

Definition at line 120 of file SubgroupFG.h.

References theGenerators.

**9.127.3.10** `int SubgroupFG::getIndex () const`

Compute the index of a subgroup of a free group, -1 means infinite.

**9.127.3.11** `const vector< Word >& SubgroupFG::getNielsenGenerators ()  
const`

Get a set of Nielsen generators of the subgroup.

**9.127.3.12 int SubgroupFG::getRank () const**

Compute the rank of a subgroup of a free group.

**9.127.3.13 string SubgroupFG::graphviz\_format () const**

Returns the graphviz graphical description of the subgroup graph (save it to file and use Graphviz to visualize the graph).

**9.127.3.14 SubgroupFG SubgroupFG::normalizer () const**

Compute the normalizer of a subgroup.

**9.127.3.15 SubgroupFG SubgroupFG::operator\* (const SubgroupFG & S)  
const**

Compute the intersection of two subgroups.

**9.127.3.16 SubgroupFG SubgroupFG::operator+ (const Word & w) const  
[inline]**

Extend subgroup basis.

Definition at line 101 of file SubgroupFG.h.

**9.127.3.17 SubgroupFG SubgroupFG::operator+ (const SubgroupFG & sbgp)  
const [inline]**

Extend subgroup basis.

Definition at line 89 of file SubgroupFG.h.

**9.127.3.18 SubgroupFG& SubgroupFG::operator+= (const Word & w)**

Extend subgroup basis.

**9.127.3.19 SubgroupFG& SubgroupFG::operator+= (const SubgroupFG &  
sbgp)**

Extend subgroup basis.

**9.127.3.20** `bool SubgroupFG::operator==(const SubgroupFG & S) const`

Check the equality of two subgroups.

**9.127.3.21** `SubgroupFG SubgroupFG::operator^ (const Word & conjugator)  
const [inline]`

Conjugate a subgroup.

Definition at line 77 of file SubgroupFG.h.

**9.127.3.22** `SubgroupFG& SubgroupFG::operator^= (const Word &  
conjugator)`

Conjugate a subgroup.

**9.127.3.23** `pair< SubgroupFG , Word > SubgroupFG::trim () const`

Trim the subgroup graph. (Cut off the "tail").

**9.127.4 Member Data Documentation****9.127.4.1** `list< FoldDetails< IntLabeledGraph::vertex_type ,  
IntLabeledGraph::edge_type > > SubgroupFG::foldDetails  
[mutable, private]`

Definition at line 193 of file SubgroupFG.h.

**9.127.4.2** `bool SubgroupFG::fsaDone [mutable, private]`

Definition at line 191 of file SubgroupFG.h.

**9.127.4.3** `bool SubgroupFG::nielsDone [mutable, private]`

Definition at line 195 of file SubgroupFG.h.

**9.127.4.4** `IntLabeledGraph SubgroupFG::theFSA [mutable, private]`

Definition at line 192 of file SubgroupFG.h.

**9.127.4.5   vector< Word > SubgroupFG::theGenerators   [private]**

Definition at line 188 of file SubgroupFG.h.

Referenced by `getGenerators()`, and `SubgroupFG()`.

**9.127.4.6   vector< Word > SubgroupFG::theNielsenGenerators   [mutable, private]**

Definition at line 196 of file SubgroupFG.h.

**9.127.4.7   int SubgroupFG::theNumberOfGenerators   [private]**

Definition at line 189 of file SubgroupFG.h.

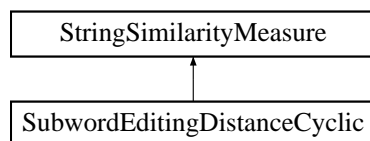
The documentation for this class was generated from the following file:

- SbgpFG/include/**SubgroupFG.h**

## 9.128 SubwordEditingDistanceCyclic Class Reference

Implements the Editing distance between two cyclic words.

#include <SimilarityMeasures.h> Inheritance diagram for SubwordEditingDistanceCyclic::



### Public Member Functions

- **SubwordEditingDistanceCyclic ()**
- double **measure** (const **Word** &w1, const **Word** &w2) const

*Returns the editing distance computed for all cyclic permutations of initial segments of given words and scaled by the length.*

### 9.128.1 Detailed Description

Implements the Editing distance between two cyclic words. No penalty is given if words have different length. Let  $l_m = \min\{|w_1|, |w_2|\}$  be the length of shortest word, then distance is computed between initial segments of length  $l_m$ .

Definition at line 180 of file SimilarityMeasures.h.

### 9.128.2 Constructor & Destructor Documentation

#### 9.128.2.1 SubwordEditingDistanceCyclic::SubwordEditingDistanceCyclic () [inline]

Definition at line 183 of file SimilarityMeasures.h.

### 9.128.3 Member Function Documentation

#### 9.128.3.1 `double SubwordEditingDistanceCyclic::measure (const Word & w1, const Word & w2) const` `[virtual]`

Returns the editing distance computed for all cyclic permutations of initial segments of given words and scaled by the length.

**Parameters:**

*w1* - the first word

*w2* - the second word

**Returns:**

the scaled Editing distance.

Implements **StringSimilarityMeasure** (p. 520).

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

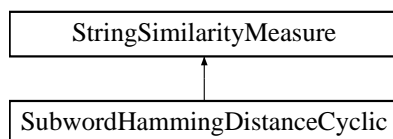
- StringSimilarity/include/**SimilarityMeasures.h**



## 9.129 SubwordHammingDistanceCyclic Class Reference

Implements the Hamming distance between two cyclic words.

#include <SimilarityMeasures.h> Inheritance diagram for SubwordHammingDistanceCyclic::



### Public Member Functions

- **SubwordHammingDistanceCyclic** ()
- double **measure** (const **Word** &w1, const **Word** &w2) const

*Returns the hamming distance computed for all cyclic permutations of initial segments of given words.*

### 9.129.1 Detailed Description

Implements the Hamming distance between two cyclic words. This is similar to **HammingDistanceCyclic** (p. 270), except no penalty is given if words have different length. Let  $l_m = \min\{|w_1|, |w_2|\}$  be the length of shortest word, then distance is computed between initial segments of length  $l_m$ .

Definition at line 141 of file SimilarityMeasures.h.

### 9.129.2 Constructor & Destructor Documentation

#### 9.129.2.1 SubwordHammingDistanceCyclic::SubwordHammingDistanceCyclic () [inline]

Definition at line 144 of file SimilarityMeasures.h.

### 9.129.3 Member Function Documentation

#### 9.129.3.1 `double SubwordHammingDistanceCyclic::measure (const Word & w1, const Word & w2) const [virtual]`

Returns the hamming distance computed for all cyclic permutations of initial segments of given words.

**Parameters:**

`w1` - the first word

`w2` - the second word

**Returns:**

the scaled Hamming distance between initial segments of cyclic words.

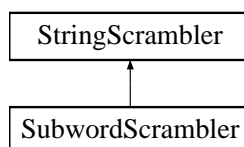
Implements **StringSimilarityMeasure** (p. 520).

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

- StringSimilarity/include/**SimilarityMeasures.h**

## 9.130 SubwordScrambler Class Reference

`#include <StringScramblers.h>`Inheritance diagram for SubwordScrambler::



### Public Member Functions

- **SubwordScrambler** (int *gs*, double *f*)
- **Word scramble** (const **Word** &) const
- double **fracChanged** () const

### Private Attributes

- double **theChangeFrac**
- int **numGens**

### 9.130.1 Detailed Description

Definition at line 59 of file StringScramblers.h.

### 9.130.2 Constructor & Destructor Documentation

#### 9.130.2.1 SubwordScrambler::SubwordScrambler (int *gs*, double *f*) [inline]

Definition at line 62 of file StringScramblers.h.

### 9.130.3 Member Function Documentation

#### 9.130.3.1 double SubwordScrambler::fracChanged () const [inline, virtual]

Implements **StringScrambler** (p. 519).

Definition at line 64 of file StringScramblers.h.

References theChangeFrac.

#### **9.130.3.2 Word SubwordScrambler::scramble (const Word &) const [virtual]**

Implements **StringScrambler** (p. 519).

### **9.130.4 Member Data Documentation**

#### **9.130.4.1 int SubwordScrambler::numGens [private]**

Definition at line 68 of file StringScramblers.h.

#### **9.130.4.2 double SubwordScrambler::theChangeFrac [private]**

Definition at line 67 of file StringScramblers.h.

Referenced by fracChanged().

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

- StringSimilarity/include/**StringScramblers.h**

## 9.131 TheGrigorchukGroupAlgorithms Class Reference

Static class **TheGrigorchukGroupAlgorithms** (p. 535) encapsulates algorithms for the original Grigorchuk group.

```
#include <TheGrigorchukGroupAlgorithms.h>
```

### Public Member Functions

- **TheGrigorchukGroupAlgorithms** ()

*Default constructor is not instantiated to protect from creating the objects of this class.*

### Static Public Member Functions

- static bool **trivial** (const **Word** &w)  
*Solve the Identity Problem for a non-reduced word.*
- static bool **trivial\_reduced** (const **Word** &w)  
*Solve the Identity Problem for a reduced word.*
- static **triple**< int, int, int > **abelianImage** (const **Word** &w)  
*Compute the abelian image of an element.*
- static **Word** **reduce** (const **Word** &w)  
*Reduce a word.*
- static int **findOrder** (const **Word** &w)  
*Find the order of an element.*
- static bool **conjugate** (const **Word** &w1, const **Word** &w2)  
*Determine if 2 words represent conjugate elements of the Grigorchuk group.*
- static set< int > **conjugate\_Kcosets** (const **Word** &w1, const **Word** &w2)  
*Compute the Q-set for a pair of words.*
- static set< **Word** > **findConjugator\_Kcosets** (const **Word** &w1, const **Word** &w2)  
*Determine if 2 words represent conjugate elements of the Grigorchuk group and find the actual conjugators, one for each K-coset.*

- static pair< **Word**, **Word** > **split** (const **Word** &w)  
*Split an element of  $St(1)$ .*
- static void **push\_back** (**Word** &w, int g)  
*Multiply an element by a generator on the right and perform a reduction if possible.*
- static void **push\_front** (**Word** &w, int g)  
*Multiply an element by a generator on the left and perform a reduction if possible.*
- static **Word** **randomWord** (int len)  
*Generate a random reduced word.*
- static pair< **Word**, list< **Word** > > **decompositionBSbgp** (const **Word** &w)  
*Compute the decomposition of a word as an element of a subgroup  $ncl_G(b)$ .*
- static pair< **Word**, **Word** > **liftToSTone** (const **Word** &w1, const **Word** &w2)  
*Compute the preimage of  $(w_1, w_2)$  along  $\psi : St_\Gamma(1) \rightarrow \Gamma \times \Gamma$ .*
- static int **cosetRepresentativeKSbgp** (const **Word** &w)  
*Find a number of a right  $K$ -coset  $Kw$  where  $K = ncl_\Gamma(abab)$ .*
- static **Word** **cosetRepresentativeKSbgp** (int c)  
*Get a word representative of a right  $K$ -coset with number  $c$ .*
- static int **liftPairKcosetsUP** (int x, int y)  
*Lift the pair of right  $K$ -cosets  $(Kx, Ky)$  to  $Kz$  if possible.*
- static set< int > **liftPairsKcosetsUP** (const set< int > &K1, const set< int > &K2)  
*For two sets of  $K$ -cosets find all  $K$ -lifts.*
- static map< pair< int, int >, int > **KCosetLiftTable** ()  
*For each pair of  $K$ -cosets that can be lifted up to  $ST_\Gamma(1)$  it associates such a lift.*

### 9.131.1 Detailed Description

Static class **TheGrigorchukGroupAlgorithms** (p.535) encapsulates algorithms for the original Grigorchuk group. The class **TheGrigorchukGroupAlgorithms** (p.535) is static, i.e., all member functions are static and there is no constructor defined. For

introduction to the original Grigorchuk group see P. de la Harpe "Topics in Geometric Group Theory". For more on algorithmic properties see survey by R. Grigorchuk "Solved and Unsolved Problems Around One Group".

Definition at line 36 of file TheGrigorchukGroupAlgorithms.h.

## 9.131.2 Constructor & Destructor Documentation

### 9.131.2.1 TheGrigorchukGroupAlgorithms::TheGrigorchukGroupAlgorithms()

Default constructor is not instantiated to protect from creating the objects of this class.

## 9.131.3 Member Function Documentation

### 9.131.3.1 static triple< int , int , int > TheGrigorchukGroupAlgorithms::abelianImage (const Word & w) [static]

Compute the abelian image of an element.

### 9.131.3.2 static bool TheGrigorchukGroupAlgorithms::conjugate (const Word & w1, const Word & w2) [inline, static]

Determine if 2 words represent conjugate elements of the Grigorchuk group.

Definition at line 103 of file TheGrigorchukGroupAlgorithms.h.

References conjugate\_Kcosets().

### 9.131.3.3 static set< int > TheGrigorchukGroupAlgorithms::conjugate\_Kcosets (const Word & w1, const Word & w2) [static]

Compute the  $Q$ -set for a pair of words.

Referenced by conjugate().

### 9.131.3.4 static Word TheGrigorchukGroupAlgorithms::cosetRepresentativeKSbgp (int c) [static]

Get a word representative of a right K-coset with number c.

**9.131.3.5** `static int TheGrigorchukGroupAlgorithms::cosetRepresentativeKSbgp (const Word & w)`  
**[static]**

Find a number of a right K-coset  $Kw$  where  $K = ncl_{\Gamma}(abab)$ .

**9.131.3.6** `static pair< Word , list< Word > > TheGrigorchukGroupAlgorithms::decompositionBSbgp (const Word & w)` **[static]**

Compute the decomposition of a word as an element of a subgroup  $ncl_G(b)$ . The function returns a pair  $(c, L)$  where  $c$  is a right coset for  $w$  and  $L$  is a sequence of conjugators for  $b$  comprising the decomposition for  $w$ . The equality  $w = (\prod_{i=1}^n L_i^{-1} b L_i) c$  holds.

**9.131.3.7** `static set< Word > TheGrigorchukGroupAlgorithms::findConjugator_Kcosets (const Word & w1, const Word & w2)`  
**[static]**

Determine if 2 words represent conjugate elements of the Grigorchuk group and find the actual conjugators, one for each K-coset.

**9.131.3.8** `static int TheGrigorchukGroupAlgorithms::findOrder (const Word & w)` **[static]**

Find the order of an element. Compute the smallest positive number  $n$  such that  $w^n$  is trivial in the Grigorchuk group.

**9.131.3.9** `static map< pair< int , int > , int > TheGrigorchukGroupAlgorithms::KCosetLiftTable ()` **[static]**

For each pair of K-cosets that can be lifted up to  $ST_{\Gamma}(1)$  it associates such a lift.

**9.131.3.10** `static int TheGrigorchukGroupAlgorithms::liftPairKcosetsUP (int x, int y)` **[static]**

Lift the pair of right K-cosets  $(Kx, Ky)$  to  $Kz$  if possible. If  $z = (x, y)$  then knowing cosets  $xK$  and  $yK$  it is possible to find a coset  $xK$ . This operation is defined not for all pairs  $x, y$ .



**9.131.3.11** `static set< int > TheGrigorchukGroupAlgorithms::liftPairsKcosetsUP (const set< int > & K1, const set< int > & K2) [static]`

For two sets of K-cosets find all K-lifts. Function uses `liftPairKcosetsUP()` (p. 538) for each pair of cosets.

**9.131.3.12** `static pair< Word , Word > TheGrigorchukGroupAlgorithms::liftToStone (const Word & w1, const Word & w2) [static]`

Compute the preimage of  $(w_1, w_2)$  along  $\psi : St_\Gamma(1) \rightarrow \Gamma \times \Gamma$ .  $\Gamma \times \Gamma \neq \psi(St_\Gamma(1))$ . The function returns a pair  $(L, C)$ , where  $C$  is such that  $(w_1 C^{-1}, w_2) \in \psi(St_\Gamma(1))$  and  $L$  is a preimage of  $(w_1 C^{-1}, w_2)$ .

**9.131.3.13** `static void TheGrigorchukGroupAlgorithms::push_back (Word & w, int g) [static]`

Multiply an element by a generator on the right and perform a reduction if possible.

**9.131.3.14** `static void TheGrigorchukGroupAlgorithms::push_front (Word & w, int g) [static]`

Multiply an element by a generator on the left and perform a reduction if possible.

**9.131.3.15** `static Word TheGrigorchukGroupAlgorithms::randomWord (int len) [static]`

Generate a random reduced word.

**9.131.3.16** `static Word TheGrigorchukGroupAlgorithms::reduce (const Word & w) [static]`

Reduce a word. See VIII.B(12) of de la Harpe.

**9.131.3.17** `static pair< Word , Word > TheGrigorchukGroupAlgorithms::split (const Word & w) [static]`

Split an element of  $St(1)$ . Make sure  $w \in St(1)!$  The output is a pair of words  $(w_0, w_1)$ ,  $w_0$  acts on the left subtree, and  $w_1$  acts on the right subtree.

**9.131.3.18 static bool TheGrigorchukGroupAlgorithms::trivial (const Word & w) [static]**

Solve the Identity Problem for a non-reduced word. Check if a word over the original alphabet represents the identity. See VIII.E(47) of de la Harpe.

**9.131.3.19 static bool TheGrigorchukGroupAlgorithms::trivial\_reduced (const Word & w) [static]**

Solve the Identity Problem for a reduced word. Check if a word over the original alphabet represents the identity. See VIII.E(47) of de la Harpe.

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

- TheGrigorchukGroup/include/TheGrigorchukGroupAlgorithms.h

## 9.132 ThLeftNormalForm Class Reference

Defines a representation of a left Garside normal form.

```
#include <ThLeftNormalForm.h>
```

### Public Member Functions

- **ThLeftNormalForm** (int rank=0)  
*Default constructor (rank specifies the rank of a braid group the form belongs to).*
- **ThLeftNormalForm** (const **ThLeftNormalForm** &rep)
- **ThLeftNormalForm** (int rank, int p, const list< **Permutation** > &d)  
*Constructor (rank specifies the rank of a braid group the form belongs to, p - a power of a half twist, and d - a list of permutations. So the pair (p,d) is a presentation).*
- **ThLeftNormalForm** (const **NF** &pr)
- **ThLeftNormalForm** (const **BraidGroup** &G, const **Word** &w)  
*Constructs the normal form of a braid word w.*
- **ThLeftNormalForm** (const **Permutation** &p)  
*Construct a positive braid from a permutation.*
- **ThLeftNormalForm** & operator= (const **ThLeftNormalForm** &rep)  
*Assignment operator.*
- **ThLeftNormalForm** & operator= (const **NF** &pr)  
*Assignment operator.*
- bool operator== (const **ThLeftNormalForm** &rep) const  
*Compare (check if equal).*
- bool operator!= (const **ThLeftNormalForm** &rep) const  
*Compare (check if not equal).*
- bool operator< (const **ThLeftNormalForm** &rep) const  
*Compare (check if less, performed componentwise).*
- **ThLeftNormalForm** operator- () const
- **ThLeftNormalForm** operator\* (const **ThLeftNormalForm** &pr) const  
*Multiply two normal forms.*
- **ThLeftNormalForm** & operator\*= (const **ThLeftNormalForm** &rep)

*Multiply a normal form by the other on the right.*

- **operator NF () const**  
*Cast operator (returns a representation of a normal form).*
- **operator ThRightNormalForm () const**  
*Cast operator.*
- **int getRank () const**  
*Get the rank of a corresponding braid group.*
- **int getPower () const**  
*Get half-twist power.*
- **const list< Permutation > & getDecomposition () const**  
*Get a list of permutations.*
- **bool isTrivial () const**  
*Check if a normal form is trivial.*
- **ThLeftNormalForm increaseRank (int N) const**  
*Increase the rank of the braid.*
- **ThLeftNormalForm inverse () const**
- **ThLeftNormalForm multiply (const ThLeftNormalForm &rep) const**
- **Word getWord () const**  
*Computes a word represented by a normal form.*
- **void adjust ()**
- **triple< ThLeftNormalForm, ThLeftNormalForm, int > findUSSRepresentative () const**  
*(Low level function) Compute ultra summit set representative.*
- **Permutation getTransport (const ThLeftNormalForm &B, const Permutation &u) const**  
*(Aux, for USS simple) Compute a transport for \*this and  $B = (*this)^u$  where  $p$  is a permutation.*
- **set< Permutation > getTransports (const Permutation &u, int period) const**  
*(Aux, for USS simple) Compute a set of transports for a pair \*this and  $(*this)^u$  where  $p$  is a permutation.*
- **Permutation getPullback (const Permutation &s) const**

(Aux, for USS simple) Compute a pullback for  $*this$  and  $B = (*this)^s$  where  $s$  is a permutation.

- **Permutation** **getMainPullback** (const **Permutation** &s, int period) const  
(Aux, for USS simple) Compute the main pullback.
- int **computePeriod** () const  
Find a period of USS-elements. If applied not to USS-braid routine will get into an infinite loop.
- pair< **Permutation**, bool > **getSimpleUltraConjugator** (int period, const **Permutation** &start) const  
Compute a simple ultra conjugator for starting from "startSymbol".
- set< pair< **Permutation**, bool > > **getSimpleUltraConjugators** (int period) const  
Compute all simple ultra conjugator for a given braid.
- pair< bool, **ThLeftNormalForm** > **areConjugate\_uss** (const **ThLeftNormalForm** &nf, int time\_sec\_bound=999999) const  
Returns true if braids are conjugate.
- void **setPower** (int p)
- void **setDecomposition** (const list< **Permutation** > &d)
- pair< **ThLeftNormalForm**, **ThLeftNormalForm** > **cycle** () const  
Cycle a normal form.
- pair< **ThLeftNormalForm**, **ThLeftNormalForm** > **decycle** () const  
Decycle a normal form.
- pair< **ThLeftNormalForm**, **ThLeftNormalForm** > **findSSSRepresentative** () const
- **Permutation** **getSimpleConjugator** (const **Permutation** &start) const  
Compute the minimal simple conjugator (permutation) starting from "start".
- set< **Permutation** > **getSimpleConjugators** () const  
Find a list of simple conjugators (permutations) conjugation by which does not decrease the infimum of the element.
- **Permutation** **getSimpleSummitConjugator** (const **Permutation** &start) const  
Compute a minimal simple summit conjugator for starting from "start".

- `set< Permutation > getSimpleSummitConjugators ()` const

*Find a list of simple conjugators (permutations) conjugation by which does not change infimum and supremum of the given element.*

- `pair< bool, ThLeftNormalForm > areConjugate (const ThLeftNormalForm &rep)` const

*Check whether two braids are conjugate.*

## Static Public Member Functions

- static void `adjustDecomposition` (int rank, int &power, list< **Permutation** > &decomp)

## Private Types

- enum `transformationResult` { `TWO_MULTIPLIERS`, `ONE_MULTIPLIER`, `NO_CHANGE` }
- typedef `triple< int, int, list< Permutation > >` **NF**

*Presentation of a normal form.*

## Private Member Functions

- `pair< bool, ThLeftNormalForm > ussConstructionIteration` (map< **ThLeftNormalForm**, pair< **ThLeftNormalForm**, int > > &uss\_new1, map< **ThLeftNormalForm**, pair< **ThLeftNormalForm**, int > > &uss\_checked1, const map< **ThLeftNormalForm**, pair< **ThLeftNormalForm**, int > > &uss\_new2, const map< **ThLeftNormalForm**, pair< **ThLeftNormalForm**, int > > &uss\_checked2) const

*(Aux, USS)*

- `pair< bool, ThLeftNormalForm > ussAddTrajectory` (const **ThLeftNormalForm** &new\_elt, const **ThLeftNormalForm** &new\_conjugator, map< **ThLeftNormalForm**, pair< **ThLeftNormalForm**, int > > &uss\_new1, map< **ThLeftNormalForm**, pair< **ThLeftNormalForm**, int > > &uss\_checked1, const map< **ThLeftNormalForm**, pair< **ThLeftNormalForm**, int > > &uss\_new2, const map< **ThLeftNormalForm**, pair< **ThLeftNormalForm**, int > > &uss\_checked2) const

*(Aux, USS)*

## Static Private Member Functions

- static **transformationResult transform** (int theIndex, **Permutation** &p1, **Permutation** &p2)
- static void **reverse** (NF &pr)

## Private Attributes

- int **theRank**  
*The rank of the braid group (number of strands).*
- int **theOmegaPower**  
*Power of omega.*
- list< **Permutation** > **theDecomposition**  
*Sequence of permutations.*

### 9.132.1 Detailed Description

Defines a representation of a left Garside normal form. Left Garside normal form has the following presentation  $\Delta^\omega \xi_1 \dots \xi_{t-1} \xi_t$  where  $\Delta$  is a half-twist permutation and  $\xi_1 \dots \xi_{t-1} \xi_t$  is a left-weighted sequence of permutation braids. A sequence of permutation braids is called right-weighted if for any pair  $\xi_i \xi_{i+1}$   $F(\xi_i) \supseteq S(\xi_{i+1})$ . Here  $S(\xi) = \{i \mid \xi = x_i \xi' \text{ for some permutation } \xi'\}$  and  $F(\xi) = \{i \mid \xi = \xi' x_i \text{ for some permutation } \xi'\}$  are starting and finishing sets.

Permutations  $\xi$  are translated to braids "from right to left". So, you construct a minimal positive braid where points on the right will go to positions defined by  $\xi$  on the left.

Definition at line 45 of file ThLeftNormalForm.h.

### 9.132.2 Member Typedef Documentation

#### 9.132.2.1 typedef triple< int , int , list< Permutation > > ThLeftNormalForm::NF [private]

Presentation of a normal form. The first component specifies the rank of a braid group. The second component specifies the power of the half twist. The third component specifies the list of braid permutations.

Definition at line 54 of file ThLeftNormalForm.h.

### 9.132.3 Member Enumeration Documentation

#### 9.132.3.1 enum ThLeftNormalForm::transformationResult [private]

Enumerator:

*TWO\_MULTIPLIERS*

*ONE\_MULTIPLIER*

*NO\_CHANGE*

Definition at line 440 of file ThLeftNormalForm.h.

### 9.132.4 Constructor & Destructor Documentation

#### 9.132.4.1 ThLeftNormalForm::ThLeftNormalForm (int *rank* = 0) [inline]

Default constructor (rank specifies the rank of a braid group the form belongs to). Default rank value is for map<...> only make sure to provide the argument for this constructor

Definition at line 70 of file ThLeftNormalForm.h.

#### 9.132.4.2 ThLeftNormalForm::ThLeftNormalForm (const ThLeftNormalForm & *rep*) [inline]

Definition at line 76 of file ThLeftNormalForm.h.

#### 9.132.4.3 ThLeftNormalForm::ThLeftNormalForm (int *rank*, int *p*, const list< Permutation > & *d*) [inline]

Constructor (rank specifies the rank of a braid group the form belongs to, p - a power of a half twist, and d - a list of permutations. So the pair (p,d) is a presentation). There is no check that the pair (p,d) defines a correct normal form representation. If you are not sure if (p,d) is correct apply static function adjustDecomposition first.

Definition at line 87 of file ThLeftNormalForm.h.

#### 9.132.4.4 ThLeftNormalForm::ThLeftNormalForm (const NF & *pr*) [inline]

Definition at line 92 of file ThLeftNormalForm.h.



#### 9.132.4.5 ThLeftNormalForm::ThLeftNormalForm (const BraidGroup & *G*, const Word & *w*)

Constructs the normal form of a braid word *w*.

#### 9.132.4.6 ThLeftNormalForm::ThLeftNormalForm (const Permutation & *p*) [inline]

Construct a positive braid from a permutation.

Definition at line 102 of file ThLeftNormalForm.h.

References Permutation::getHalfTwistPermutation(), Permutation::isTrivial(), Permutation::size(), theDecomposition, theOmegaPower, and theRank.

### 9.132.5 Member Function Documentation

#### 9.132.5.1 void ThLeftNormalForm::adjust () [inline]

Definition at line 234 of file ThLeftNormalForm.h.

References adjustDecomposition(), theDecomposition, theOmegaPower, and theRank.

#### 9.132.5.2 static void ThLeftNormalForm::adjustDecomposition (int *rank*, int & *power*, list< Permutation > & *decomp*) [static]

Referenced by adjust().

#### 9.132.5.3 pair< bool , ThLeftNormalForm > ThLeftNormalForm::areConjugate (const ThLeftNormalForm & *rep*) const

Check whether two braids are conjugate. Current implementation of this function transforms the left form into the right form and invokes **ThRightNormalForm::areConjugate** (p.567)( ... ). Very slow function. Checks if super summit sets of two braids contain the same element (i.e., coincide). ... and the super summit sets are typically huge even for decent parameters.

#### Returns:

- a pair (R,C), where R is a boolean value (true if braids are conjugate, false otherwise), and C is a conjugator (if braids conjugate)

**9.132.5.4** `pair< bool , ThLeftNormalForm >`  
`ThLeftNormalForm::areConjugate_uss (const ThLeftNormalForm &`  
`nf, int time_sec_bound = 999999) const`

Returns true if braids are conjugate. Procedure constructs USS for 2 braids until finds intersections.

**9.132.5.5** `int ThLeftNormalForm::computePeriod () const`

Find a period of USS-elements. If applied not to USS-braid routine will get into an infinite loop.

**9.132.5.6** `pair< ThLeftNormalForm , ThLeftNormalForm >`  
`ThLeftNormalForm::cycle () const`

Cycle a normal form. Operation:  $\xi_1 \dots \xi_{t-1} \xi_t \Delta^s \mapsto \xi_t \Delta^s \xi_1 \dots \xi_{t-1} \Delta^s$ . Main purpose of the function is to increase the infimum (half twist power) of a normal form. If  $n$  cyclings do not increase infimum then the infimum is already maximal.

**Returns:**

a pair (R,C), where R is the result of a cycling and C is a conjugator (if A is an input then  $R = C^{-1}AC$ )

**9.132.5.7** `pair< ThLeftNormalForm , ThLeftNormalForm >`  
`ThLeftNormalForm::decycle () const`

Decycle a normal form. Operation:  $\xi_1 \xi_2 \dots \xi_t \Delta^s \mapsto \Delta^s \xi_2 \dots \xi_t \Delta^s \xi_1$ . Main purpose of the function is to decrease the supremum (size of the decomposition) of a normal form. If  $n$  decyclings do not decrease supremum then the supremum is already minimal.

**Returns:**

a pair (R,C), where R is the result of a cycling and C is a conjugator (if A is an input then  $R = C^{-1}AC$ )

**9.132.5.8** `pair< ThLeftNormalForm , ThLeftNormalForm >`  
`ThLeftNormalForm::findSSSRepresentative () const`

**9.132.5.9** `triple< ThLeftNormalForm , ThLeftNormalForm , int >`  
`ThLeftNormalForm::findUSSRepresentative () const`

(Low level function) Compute ultra summit set representative.

**Returns:**

a **triple** (p. 577) (R,C,P), where R is the result, C is a conjugator (if A is an input then  $R = C^{-1}AC$ ), and P is the period of the trajectory.

**9.132.5.10** `const list< Permutation >& ThLeftNormalForm::getDecomposition  
() const [inline]`

Get a list of permutations.

Definition at line 203 of file ThLeftNormalForm.h.

References theDecomposition.

Referenced by TTPAttack::printStats().

**9.132.5.11** `Permutation ThLeftNormalForm::getMainPullback (const  
Permutation & s, int period) const`

(Aux, for USS simple) Compute the main pullback.

**9.132.5.12** `int ThLeftNormalForm::getPower () const [inline]`

Get half-twist power.

Definition at line 201 of file ThLeftNormalForm.h.

References theOmegaPower.

Referenced by TTPAttack::printStats().

**9.132.5.13** `Permutation ThLeftNormalForm::getPullback (const Permutation  
& s) const`

(Aux, for USS simple) Compute a pullback for \*this and  $B = (*this)^s$  where s is a permutation. See Definition 4.6 in Gebhardt, "A New Approach to the Conjugacy Problem in Garside Groups". It is important that braids (\*this) and (\*this)<sup>u</sup> belong to their SSS. Also, it is important that (\*this) is not a power of  $\Delta$  (USS is "trivial" for such elements).

**9.132.5.14** `int ThLeftNormalForm::getRank () const [inline]`

Get the rank of a corresponding braid group.

Definition at line 199 of file ThLeftNormalForm.h.

References theRank.

#### 9.132.5.15 **Permutation ThLeftNormalForm::getSimpleConjugator (const Permutation & start) const**

Compute the minimal simple conjugator (permutation) starting from "start". Algorithm 1 from Franco, Gonzalez-Menese, "Conjugacy problem for braid and Garside groups". Conjugating by a simple element does not decrease the infimum of the element.

#### 9.132.5.16 **set< Permutation > ThLeftNormalForm::getSimpleConjugators () const**

Find a list of simple conjugators (permutations) conjugation by which does not decrease the infimum of the element. The current version is very simple. For each permutation cycle  $(i, i + 1)$  it computes the minimal permutation  $p$  which starts with  $(i, i + 1)$  such that conjugation by  $p$  does not decrease the infimum of the element and outputs all of such permutations. The output is not guaranteed to be minimal, i.e., some of the permutations can be prefixes of the other.

#### 9.132.5.17 **Permutation ThLeftNormalForm::getSimpleSummitConjugator (const Permutation & start) const**

Compute a minimal simple summit conjugator for starting from "start". Algorithm 4 from Franco, Gonzalez-Menese, "Conjugacy problem for braid and Garside groups". Conjugating by a simple summit element does not decrease the infimum of the element and does not increase the canonical length.

#### 9.132.5.18 **set< Permutation > ThLeftNormalForm::getSimpleSummitConjugators () const**

Find a list of simple conjugators (permutations) conjugation by which does not change infimum and supremum of the given element. The current version is very simple. For each permutation cycle  $(i, i + 1)$  it computes the minimal permutation  $p$  which starts with  $(i, i + 1)$  such that conjugation by  $p$  does not decrease the infimum of the element and does not increase the canonical length, and outputs all of such permutations. The output is not guaranteed to be minimal, i.e., some of the permutations can be prefixes of the other.

#### 9.132.5.19 **pair< Permutation , bool > ThLeftNormalForm::getSimpleUltraConjugator (int period, const Permutation & start) const**

Compute a simple ultra conjugator for starting from "startSymbol". Algorithm 4.9 from Gebhardt, "A New Approach to the Conjugacy Problem in Garside Groups". Conjugating by a simple ultra element does not decrease the infimum of the element and does

not increase the canonical length. Also, the result of the conjugation belongs to the USS.

**Returns:**

a pair (R,M), where R is the resulting permutation, M is true if R is minimal among all simple ultra conjugators.

**9.132.5.20** `set< pair< Permutation , bool > >  
ThLeftNormalForm::getSimpleUltraConjugators (int period) const`

Compute all simple ultra conjugator for a given braid.

**9.132.5.21** `Permutation ThLeftNormalForm::getTransport (const  
ThLeftNormalForm & B, const Permutation & u) const`

(Aux, for USS simple) Compute a transport for \*this and  $B = (*this)^u$  where p is a permutation. See Proposition 2.1 and Definition 2.4 of Gebhardt "A New Approach to the Conjugacy Problem in Garside Groups". It is important that braids (\*this) and  $B = (*this)^u$  belong to their SSS. Also, it is important that (\*this) is not a power of  $\Delta$  (USS is "trivial" for such elements).

**9.132.5.22** `set< Permutation > ThLeftNormalForm::getTransports (const  
Permutation & u, int period) const`

(Aux, for USS simple) Compute a set of transports for a pair \*this and  $(*this)^u$  where p is a permutation. F-set, see Definition 4.2 in Gebhardt, "A New Approach to the Conjugacy Problem in Garside Groups". It is important that braids (\*this) and  $(*this)^u$  belong to their SSS. Also, it is important that (\*this) is not a power of  $\Delta$  (USS is "trivial" for such elements).

**9.132.5.23** `Word ThLeftNormalForm::getWord () const`

Computes a word represented by a normal form. For each permutation-braid computes a braid-word it represents and, finally, concatenate all the results.

**9.132.5.24** `ThLeftNormalForm ThLeftNormalForm::increaseRank (int N)  
const`

Increase the rank of the braid. The current braid does not change as an element of  $F_\infty$ . We simply increase the rank and recompute the normal form presentation. If  $N < \text{theRank}$  then output \*this.

**9.132.5.25 ThLeftNormalForm ThLeftNormalForm::inverse () const**

Referenced by operator-().

**9.132.5.26 bool ThLeftNormalForm::isTrivial () const [inline]**

Check if a normal form is trivial.

Definition at line 207 of file ThLeftNormalForm.h.

References theDecomposition, and theOmegaPower.

**9.132.5.27 ThLeftNormalForm ThLeftNormalForm::multiply (const ThLeftNormalForm & rep) const**

Referenced by operator\*(), and operator\*=( ).

**9.132.5.28 ThLeftNormalForm::operator NF () const [inline]**

Cast operator (returns a representation of a normal form).

Definition at line 183 of file ThLeftNormalForm.h.

References theDecomposition, theOmegaPower, and theRank.

**9.132.5.29 ThLeftNormalForm::operator ThRightNormalForm () const**

Cast operator.

**9.132.5.30 bool ThLeftNormalForm::operator!= (const ThLeftNormalForm & rep) const [inline]**

Compare (check if not equal).

Definition at line 145 of file ThLeftNormalForm.h.

References theDecomposition, theOmegaPower, and theRank.

**9.132.5.31 ThLeftNormalForm ThLeftNormalForm::operator\* (const ThLeftNormalForm & pr) const [inline]**

Multiply two normal forms.

Definition at line 172 of file ThLeftNormalForm.h.

References multiply().

#### 9.132.5.32 ThLeftNormalForm& ThLeftNormalForm::operator\*= (const ThLeftNormalForm & rep) [inline]

Multiply a normal form by the other on the right.

Definition at line 176 of file ThLeftNormalForm.h.

References multiply().

#### 9.132.5.33 ThLeftNormalForm ThLeftNormalForm::operator- () const [inline]

Definition at line 168 of file ThLeftNormalForm.h.

References inverse().

#### 9.132.5.34 bool ThLeftNormalForm::operator< (const ThLeftNormalForm & rep) const [inline]

Compare (check if less, performed componentwise).

Definition at line 154 of file ThLeftNormalForm.h.

References theDecomposition, theOmegaPower, and theRank.

#### 9.132.5.35 ThLeftNormalForm& ThLeftNormalForm::operator= (const NF & pr) [inline]

Assignment operator.

Definition at line 133 of file ThLeftNormalForm.h.

References triple< T1, T2, T3 >::first, triple< T1, T2, T3 >::second, theDecomposition, theOmegaPower, theRank, and triple< T1, T2, T3 >::third.

#### 9.132.5.36 ThLeftNormalForm& ThLeftNormalForm::operator= (const ThLeftNormalForm & rep) [inline]

Assignment operator.

Definition at line 125 of file ThLeftNormalForm.h.

References theDecomposition, theOmegaPower, and theRank.

**9.132.5.37** `bool ThLeftNormalForm::operator==(const ThLeftNormalForm & rep) const`

Compare (check if equal).

**9.132.5.38** `static void ThLeftNormalForm::reverse(NF & pr) [static, private]`

**9.132.5.39** `void ThLeftNormalForm::setDecomposition(const list< Permutation > & d) [inline]`

Definition at line 348 of file ThLeftNormalForm.h.

References theDecomposition.

**9.132.5.40** `void ThLeftNormalForm::setPower(int p) [inline]`

Definition at line 346 of file ThLeftNormalForm.h.

References theOmegaPower.

**9.132.5.41** `static transformationResult ThLeftNormalForm::transform(int theIndex, Permutation & p1, Permutation & p2) [static, private]`

**9.132.5.42** `pair< bool , ThLeftNormalForm > ThLeftNormalForm::ussAddTrajectory(const ThLeftNormalForm & new_elt, const ThLeftNormalForm & new_conjugator, map< ThLeftNormalForm, pair< ThLeftNormalForm, int > > & uss_new1, map< ThLeftNormalForm, pair< ThLeftNormalForm, int > > & uss_checked1, const map< ThLeftNormalForm, pair< ThLeftNormalForm, int > > & uss_new2, const map< ThLeftNormalForm, pair< ThLeftNormalForm, int > > & uss_checked2) const [private]`

(Aux, USS)



```

9.132.5.43 pair< bool , ThLeftNormalForm >
    ThLeftNormalForm::ussConstructionIteration (map<
    ThLeftNormalForm, pair< ThLeftNormalForm, int > > &
    uss_new1, map< ThLeftNormalForm, pair< ThLeftNormalForm,
    int > > & uss_checked1, const map< ThLeftNormalForm,
    pair< ThLeftNormalForm, int > > & uss_new2, const map<
    ThLeftNormalForm, pair< ThLeftNormalForm, int > > &
    uss_checked2) const [private]

```

(Aux, USS)

## 9.132.6 Member Data Documentation

```

9.132.6.1 list< Permutation > ThLeftNormalForm::theDecomposition
    [private]

```

Sequence of permutations.

Definition at line 460 of file ThLeftNormalForm.h.

Referenced by `adjust()`, `getDecomposition()`, `isTrivial()`, `operator NF()`, `operator!=()`, `operator<()`, `operator=()`, `setDecomposition()`, and `ThLeftNormalForm()`.

```

9.132.6.2 int ThLeftNormalForm::theOmegaPower [private]

```

Power of omega.

Definition at line 457 of file ThLeftNormalForm.h.

Referenced by `adjust()`, `getPower()`, `isTrivial()`, `operator NF()`, `operator!=()`, `operator<()`, `operator=()`, `setPower()`, and `ThLeftNormalForm()`.

```

9.132.6.3 int ThLeftNormalForm::theRank [private]

```

The rank of the braid group (number of strands).

Definition at line 454 of file ThLeftNormalForm.h.

Referenced by `adjust()`, `getRank()`, `operator NF()`, `operator!=()`, `operator<()`, `operator=()`, and `ThLeftNormalForm()`.

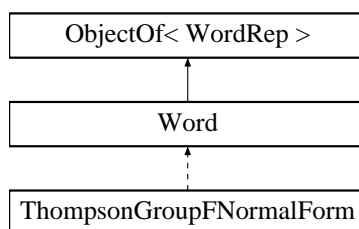
The documentation for this class was generated from the following file:

- BraidGroup/include/ThLeftNormalForm.h

## 9.133 ThompsonGroupFNormalForm Class Reference

Class **ThompsonGroupFNormalForm** (p. 556) (defines a representation of a normal form of an element of the Thompson's group F (infinitely generated))//.

#include <ThompsonGroupFNormalForm.h> Inheritance diagram for ThompsonGroupFNormalForm::



### Public Member Functions

- **ThompsonGroupFNormalForm** ()
- **ThompsonGroupFNormalForm** (const **Word** &w)
- bool **operator==** (const **ThompsonGroupFNormalForm** &nf) const  
*Comparison operator.*
- **ThompsonGroupFNormalForm** & **operator\*=** (const **ThompsonGroupFNormalForm** &nf)  
*Multiply the word on the right by another word (the result is reduced).*
- **ThompsonGroupFNormalForm** **operator\*** (const **ThompsonGroupFNormalForm** &nf) const  
*Multiply two normal forms (the result is a normal form).*
- **ThompsonGroupFNormalForm** **operator-** () const  
*Multiply two normal forms (the result is a normal form).*

### Static Private Member Functions

- static **Word** **semiNormalFormFor** (const **Word** &w)  
*Function returns a "sorted peak-like word" which can contain bad pairs of generators.*
- static list< int > **mergeUnits** (list< int > &unit1, list< int > &unit2)  
*Function merges 2 seminormal forms.*

- static int **nextOperation** (int n1, int n2, int n3, int n4)  
*Auxiliary function used in "mergeUnits".*
- static **Word removeBadPairs** (const **Word** &w)  
*Remove pairs of inverse generators in a seminormal form.*

## Friends

- ostream & **operator<<** (ostream &os, const **ThompsonGroupFNormalForm** &nf)

### 9.133.1 Detailed Description

Class **ThompsonGroupFNormalForm** (p. 556) (defines a representation of a normal form of an element of the Thompson's group F (infinitely generated))//. A normal form of an element of the Thompson's group F is a word  $w = n \cdot p$  To be finished.

Definition at line 29 of file ThompsonGroupFNormalForm.h.

### 9.133.2 Constructor & Destructor Documentation

#### 9.133.2.1 ThompsonGroupFNormalForm::ThompsonGroupFNormalForm ()

Referenced by operator-().

#### 9.133.2.2 ThompsonGroupFNormalForm::ThompsonGroupFNormalForm (const **Word** & w)

### 9.133.3 Member Function Documentation

#### 9.133.3.1 static list< int > ThompsonGroupFNormalForm::mergeUnits (list< int > & unit1, list< int > & unit2) [static, private]

Function merges 2 seminormal forms.

Referenced by operator\*(), and operator\*=(.).

#### 9.133.3.2 static int ThompsonGroupFNormalForm::nextOperation (int n1, int n2, int n3, int n4) [static, private]

Auxiliary function used in "mergeUnits".

### 9.133.3.3 **ThompsonGroupFNormalForm** **ThompsonGroupFNormalForm::operator\* (const** **ThompsonGroupFNormalForm & *nf*) const [inline]**

Multiply two normal forms (the result is a normal form).

Reimplemented from **Word** (p. 653).

Definition at line 68 of file ThompsonGroupFNormalForm.h.

References **Word::getList()**, **mergeUnits()**, and **removeBadPairs()**.

### 9.133.3.4 **ThompsonGroupFNormalForm&** **ThompsonGroupFNormalForm::operator\*= (const** **ThompsonGroupFNormalForm & *nf*) [inline]**

Multiply the word on the right by another word (the result is reduced).

Reimplemented from **Word** (p. 653).

Definition at line 58 of file ThompsonGroupFNormalForm.h.

References **Word::getList()**, **mergeUnits()**, and **removeBadPairs()**.

### 9.133.3.5 **ThompsonGroupFNormalForm** **ThompsonGroupFNormalForm::operator- () const** **[inline]**

Multiply two normal forms (the result is a normal form).

Reimplemented from **Word** (p. 653).

Definition at line 76 of file ThompsonGroupFNormalForm.h.

References **ThompsonGroupFNormalForm()**.

### 9.133.3.6 **bool ThompsonGroupFNormalForm::operator== (const** **ThompsonGroupFNormalForm & *wr*) const [inline]**

Comparison operator.

Reimplemented from **Word** (p. 654).

Definition at line 52 of file ThompsonGroupFNormalForm.h.

### 9.133.3.7 **static Word ThompsonGroupFNormalForm::removeBadPairs (const** **Word & *w*) [static, private]**

Remove pairs of inverse generators in a seminormal form.

Referenced by `operator*()`, and `operator*=(())`.

**9.133.3.8** `static Word ThompsonGroupFNormalForm::semiNormalFormFor  
(const Word & w) [static, private]`

Function returns a "sorted peak-like word" which can contain bad pairs of generators.

### 9.133.4 Friends And Related Function Documentation

**9.133.4.1** `ostream& operator<< (ostream & os, const  
ThompsonGroupFNormalForm & nf) [friend]`

Reimplemented from **Word** (p. 657).

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

- ThompsonGroup/include/ThompsonGroupFNormalForm.h

## 9.134 ThRightNormalForm Class Reference

Defines a representation of a right Garside normal form.

```
#include <ThRightNormalForm.h>
```

### Public Types

- typedef **triple**< int, int, list< **Permutation** > > **NF**

*Presentation of a normal form.*

### Public Member Functions

- **ThRightNormalForm** (int rank=0)  
*Default constructor (rank specifies the rank of a braid group the form belongs to).*
- **ThRightNormalForm** (int rank, int p, const list< **Permutation** > &d)  
*Constructor (rank specifies the rank of a braid group the form belongs to, p - a power of a half twist, and d - a list of permutations. So the pair (p,d) is a presentation).*
- **ThRightNormalForm** (const **NF** &tr)  
*Constructor (tr specifies a presentation of the normal form).*
- **ThRightNormalForm** (const **BraidGroup** &G, const **Word** &w)  
*Constructs the normal form of a braid word w.*
- **ThRightNormalForm** (const **Permutation** &p)  
*Construct a positive braid from a permutation.*
- **ThRightNormalForm** & **operator=** (const **NF** &tr)  
*Assignment operator.*
- **operator NF** () const  
*Cast operator (returns a representation of a normal form).*
- bool **operator==** (const **ThRightNormalForm** &rep) const  
*Compare (check if equal).*
- bool **operator!=** (const **ThRightNormalForm** &rep) const  
*Compare (check if not equal).*

- **bool operator<** (const **ThRightNormalForm** &rep) const  
*Compare (check if less, performed componentwise).*
- **ThRightNormalForm operator-** () const
- **ThRightNormalForm operator\*** (const **ThRightNormalForm** &rep) const  
*Multiply two normal forms.*
- **ThRightNormalForm & operator\*=** (const **ThRightNormalForm** &rep)  
*Multiply a normal form by the other on the right.*
- **operator ThLeftNormalForm** () const  
*Cast operator.*
- **int getRank** () const  
*Get the rank of a corresponding braid group.*
- **int getPower** () const  
*Get half-twist power.*
- **const list< Permutation > & getDecomposition** () const  
*Get a list of permutations.*
- **bool isTrivial** () const  
*Check if a normal form is trivial.*
- **Word getWord** () const  
*Computes a word represented by a normal form.*
- **Word getShortWord** () const  
*Computes a "shorter" word represented by a normal form.*
- **ThRightNormalForm increaseRank** (int N) const  
*Increase the rank of the braid.*
- **pair< bool, ThRightNormalForm > areConjugate** (const **ThRightNormalForm** &rep) const  
*Check whether two braids are conjugate.*
- **void adjust** ()  
*Adjust a normal form.*
- **set< ThRightNormalForm > computeCentralizer** () const

*Compute generators of the centralizer of an element.*

- **pair< ThRightNormalForm, ThRightNormalForm > findSSSRepresentative () const**  
*(Low level function) Compute super summit set representative.*
- **pair< ThRightNormalForm, ThRightNormalForm > cycle () const**  
*Cycle a normal form.*
- **pair< ThRightNormalForm, ThRightNormalForm > decycle () const**  
*Decycle a normal form.*
- **Permutation getSimpleConjugator (const Permutation &start) const**  
*Compute the minimal simple conjugator (permutation) starting from "start".*
- **set< Permutation > getSimpleConjugators () const**  
*Find a list of simple conjugators (permutations) conjugation by which does not decrease the infimum of the element.*
- **Permutation getSimpleSummitConjugator (const Permutation &start) const**  
*Compute a minimal simple summit conjugator for starting from "start".*
- **set< Permutation > getSimpleSummitConjugators () const**  
*Find a list of simple conjugators (permutations) conjugation by which does not change infimum and supremum of the given element.*
- **triple< ThRightNormalForm, ThRightNormalForm, int > findUSSRepresentative () const**  
*(Low level function) Compute ultra summit set representative.*
- **triple< Permutation, bool, int > getSimpleUltraConjugator (int period, const Permutation &start)**  
*Compute a simple ultra conjugator for starting from "startSymbol".*
- **set< pair< Permutation, int > > getSimpleUltraConjugators (int period)**  
*Compute all minimal simple ultra conjugators for "rep".*
- **set< Permutation > getTransports (const Permutation &u, int period) const**  
*(Aux, for USS simple) Compute a set of transports for a pair \*this and (\*this)^u where p is a permutation.*
- **Permutation getTransport (const ThRightNormalForm &B, const Permutation &u) const**



(Aux, for USS simple) Compute a transport for *\*this* and  $B = (*this)^u$  where  $p$  is a permutation.

- **Permutation getPullback** (const **Permutation** &s) const  
(Aux, for USS simple) Compute a pullback for *\*this* and  $B = (*this)^s$  where  $s$  is a permutation.
- **Permutation getPullback** (const **Permutation** &u, int period) const  
(Aux, for USS simple) Compute the main pullback for *\*this* and a permutation  $u$
- void **setPower** (int p)  
Set a power of a half-twist.
- void **setDecomposition** (const list< **Permutation** > &d)  
Set a decomposition of a normal form (without any check of "greedy conditions").

## Static Public Member Functions

- static **ThRightNormalForm randomPositive** (int rank, int decomp\_length)  
Generate random positive normal form.
- static void **adjustDecomposition** (int rank, int &power, list< **Permutation** > &decomp)  
Adjust a normal form.

## Private Types

- enum **transformationResult** { **TWO\_MULTIPLIERS**, **ONE\_MULTIPLIER**, **NO\_CHANGE** }  
The main function which "canonify" a product of two permutations.

## Private Member Functions

- **NF inverse** () const  
(Low level function) Invert a normal form.
- **NF multiply** (const **ThRightNormalForm** &rep) const  
(Low level function) Multiply normal forms.

- `pair< bool, ThRightNormalForm > sssConstructionIteration` (`map< ThRightNormalForm, ThRightNormalForm > &sss_new1, map< ThRightNormalForm, ThRightNormalForm > &sss_checked1, const map< ThRightNormalForm, ThRightNormalForm > &sss_new2, const map< ThRightNormalForm, ThRightNormalForm > &sss_checked2`) const

*I think function processes a vertex in a Super Summit Set graph.*

## Static Private Member Functions

- static `transformationResult transform` (int theIndex, `Permutation &p1, Permutation &p2`)

## Private Attributes

- int `theRank`

*The rank of the braid group (number of strands).*

- int `theOmegaPower`

*Power of omega.*

- list< `Permutation` > `theDecomposition`

*Sequence of permutations.*

### 9.134.1 Detailed Description

Defines a representation of a right Garside normal form. Right Garside normal form has the following presentation  $\xi_1 \dots \xi_{t-1} \xi_t \Delta^\omega$  where  $\Delta$  is a half-twist permutation and  $\xi_1 \dots \xi_{t-1} \xi_t$  is a right-weighted sequence of permutation braids. A sequence of permutation braids is called right-weighted if for any pair  $\xi_i \xi_{i+1}$   $F(\xi_i) \subseteq S(\xi_{i+1})$ . Here  $S(\xi) = \{i \mid \xi = x_i \xi' \text{ for some permutation } \xi'\}$  and  $F(\xi) = \{i \mid \xi = \xi' x_i \text{ for some permutation } \xi'\}$  are starting and finishing sets.

Permutations  $\xi$  are translated to braids "from right to left". So, you construct a minimal positive braid where points on the right will go to positions defined by  $\xi$  on the left.

Definition at line 42 of file ThRightNormalForm.h.

## 9.134.2 Member Typedef Documentation

### 9.134.2.1 typedef triple< int , int , list< Permutation > > ThRightNormalForm::NF

Presentation of a normal form. The first component specifies the rank of a braid group. The second component specifies the power of the half twist. The third component specifies the list of braid permutations.

Definition at line 51 of file ThRightNormalForm.h.

## 9.134.3 Member Enumeration Documentation

### 9.134.3.1 enum ThRightNormalForm::transformationResult [private]

The main function which "canonify" a product of two permutations.

Enumerator:

*TWO\_MULTIPLIERS*  
*ONE\_MULTIPLIER*  
*NO\_CHANGE*

Definition at line 444 of file ThRightNormalForm.h.

## 9.134.4 Constructor & Destructor Documentation

### 9.134.4.1 ThRightNormalForm::ThRightNormalForm (int *rank* = 0) [inline]

Default constructor (rank specifies the rank of a braid group the form belongs to). Default rank value is for map<...> only make sure to provide the argument for this constructor

Definition at line 66 of file ThRightNormalForm.h.

### 9.134.4.2 ThRightNormalForm::ThRightNormalForm (int *rank*, int *p*, const list< Permutation > & *d*) [inline]

Constructor (rank specifies the rank of a braid group the form belongs to, p - a power of a half twist, and d - a list of permutations. So the pair (p,d) is a presentation). There is no check that the pair (p,d) defines a correct normal form representation. If you are not sure if (p,d) is correct apply static function adjustDecomposition first.

Definition at line 77 of file ThRightNormalForm.h.

#### 9.134.4.3 **ThRightNormalForm::ThRightNormalForm (const NF & *tr*)** [inline]

Constructor (*tr* specifies a presentation of the normal form).

Definition at line 84 of file ThRightNormalForm.h.

#### 9.134.4.4 **ThRightNormalForm::ThRightNormalForm (const BraidGroup & *G*, const Word & *w*)**

Constructs the normal form of a braid word *w*.

#### 9.134.4.5 **ThRightNormalForm::ThRightNormalForm (const Permutation & *p*)** [inline]

Construct a positive braid from a permutation.

Definition at line 95 of file ThRightNormalForm.h.

References Permutation::getHalfTwistPermutation(), Permutation::isTrivial(), Permutation::size(), theDecomposition, theOmegaPower, and theRank.

### 9.134.5 Member Function Documentation

#### 9.134.5.1 **void ThRightNormalForm::adjust ()** [inline]

Adjust a normal form. Use this function if there is a possibility that the sequence of permutations does not satisfy the "greedy conditions". Right now this might happen only after use of a constructor **ThRightNormalForm** (p. 560) (const NF &tr) or, more likely, in function setDecomposition(...). This function uses adjustDecomposition( ... ).

Definition at line 260 of file ThRightNormalForm.h.

References adjustDecomposition(), theDecomposition, theOmegaPower, and theRank.

#### 9.134.5.2 **static void ThRightNormalForm::adjustDecomposition (int *rank*, int & *power*, list< Permutation > & *decomp*)** [static]

Adjust a normal form. Use this function if the **triple** (p. 577) of arguments does not satisfy "greedy conditions".

Referenced by adjust().

**9.134.5.3** `pair< bool , ThRightNormalForm >`  
`ThRightNormalForm::areConjugate (const ThRightNormalForm &`  
`rep) const`

Check whether two braids are conjugate. Very slow function. Checks if super summit sets of two braids contain the same element (i.e., coincide). ... and the super summit sets are typically huge even for decent parameters.

**Returns:**

- a pair (R,C), where R is a boolean value (true if braids are conjugate, false otherwise), and C is a conjugator (if braids conjugate)

**9.134.5.4** `set<ThRightNormalForm> ThRightNormal-`  
`Form::computeCentralizer () const`

Compute generators of the centralizer of an element. Very slow function. Function computes the graph with the super summit set of the element as the vertex set. Even for a small rank of a braid group (say 10) and a short braid word (say 100) the super summit set is typically huge.

**9.134.5.5** `pair< ThRightNormalForm , ThRightNormalForm >`  
`ThRightNormalForm::cycle () const`

Cycle a normal form. Operation:  $\xi_1 \dots \xi_{t-1} \xi_t \Delta^s \mapsto \xi_t \Delta^s \xi_1 \dots \xi_{t-1} \Delta^s$ . Main purpose of the function is to increase the infimum (half twist power) of a normal form. If  $n$  cyclings do not increase infimum then the infimum is already maximal.

**Returns:**

a pair (R,C), where R is the result of a cycling and C is a conjugator (if A is an input then  $R = C^{-1}AC$ )

**9.134.5.6** `pair<ThRightNormalForm,ThRightNormalForm>`  
`ThRightNormalForm::decycle () const`

Decycle a normal form. Operation:  $\xi_1 \xi_2 \dots \xi_t \Delta^s \mapsto \Delta^s \xi_2 \dots \xi_t \Delta^s \xi_1$ . Main purpose of the function is to decrease the supremum (size of the decomposition) of a normal form. If  $n$  decyclings do not decrease supremum then the supremum is already minimal.

**Returns:**

a pair (R,C), where R is the result of a cycling and C is a conjugator (if A is an input then  $R = C^{-1}AC$ )

**9.134.5.7** `pair< ThRightNormalForm , ThRightNormalForm >  
ThRightNormalForm::findSSSRepresentative () const`

(Low level function) Compute super summit set representative.

**9.134.5.8** `triple< ThRightNormalForm , ThRightNormalForm , int >  
ThRightNormalForm::findUSSRepresentative () const`

(Low level function) Compute ultra summit set representative.

**Returns:**

a **triple** (p.577) (R,C,P), where R is the result, C is a conjugator (if A is an input then  $R = C^{-1}AC$ ), and P is the period of the trajectory.

**9.134.5.9** `const list< Permutation >& ThRightNormalForm::getDecomposition  
() const [inline]`

Get a list of permutations.

Definition at line 195 of file ThRightNormalForm.h.

References theDecomposition.

**9.134.5.10** `int ThRightNormalForm::getPower () const [inline]`

Get half-twist power.

Definition at line 193 of file ThRightNormalForm.h.

References theOmegaPower.

**9.134.5.11** `Permutation ThRightNormalForm::getPullback (const Permutation  
& u, int period) const`

(Aux, for USS simple) Compute the main pullback for \*this and a permutation u

**9.134.5.12** `Permutation ThRightNormalForm::getPullback (const Permutation  
& s) const`

(Aux, for USS simple) Compute a pullback for \*this and  $B = (*this)^s$  where s is a permutation. See Definition 4.6 in Gebhardt, "A New Approach to the Conjugacy Problem in Garside Groups". It is important that braids (\*this) and (\*this)<sup>u</sup> belong to their SSS. Also, it is important that (\*this) is not a power of  $\Delta$  (USS is "trivial" for such elements).

**9.134.5.13 int ThRightNormalForm::getRank () const [inline]**

Get the rank of a corresponding braid group.

Definition at line 191 of file ThRightNormalForm.h.

References `theRank`.

**9.134.5.14 Word ThRightNormalForm::getShortWord () const**

Computes a "shorter" word represented by a normal form. Try to use the fact that the square of a half twist generates a center. If the power of half-twist is negative then it cancel them with the other permutation braids. After that, for newly obtained permutations computes braid words and concatenates them.

**9.134.5.15 Permutation ThRightNormalForm::getSimpleConjugator (const Permutation & start) const**

Compute the minimal simple conjugator (permutation) starting from "start". Algorithm 1 from Franco, Gonzalez-Menese, "Conjugacy problem for braid and Garside groups". Conjugating by a simple element does not decrease the infimum of the element.

**9.134.5.16 set< Permutation > ThRightNormalForm::getSimpleConjugators () const**

Find a list of simple conjugators (permutations) conjugation by which does not decrease the infimum of the element. The current version is very simple. For each permutation cycle  $(i, i + 1)$  it computes the minimal permutation  $p$  which starts with  $(i, i + 1)$  such that conjugation by  $p$  does not decrease the infimum of the element and outputs all of such permutations. The output is not guaranteed to be minimal, i.e., some of the permutations can be prefixes of the other.

**9.134.5.17 Permutation ThRightNormalForm::getSimpleSummitConjugator (const Permutation & start) const**

Compute a minimal simple summit conjugator for starting from "start". Algorithm 4 from Franco, Gonzalez-Menese, "Conjugacy problem for braid and Garside groups". Conjugating by a simple summit element does not decrease the infimum of the element and does not increase the canonical length.

#### 9.134.5.18 `set< Permutation > ThRightNormalForm::getSimpleSummitConjugators () const`

Find a list of simple conjugators (permutations) conjugation by which does not change infimum and supremum of the given element. The current version is very simple. For each permutation cycle  $(i, i + 1)$  it computes the minimal permutation  $p$  which starts with  $(i, i + 1)$  such that conjugation by  $p$  does not decrease the infimum of the element and does not increase the canonical length, and outputs all of such permutations. The output is not guaranteed to be minimal, i.e., some of the permutations can be prefixes of the other.

#### 9.134.5.19 `triple< Permutation , bool , int > ThRightNormalForm::getSimpleUltraConjugator (int period, const Permutation & start)`

Compute a simple ultra conjugator for starting from "startSymbol". Algorithm 4.9 from Gebhardt, "A New Approach to the Conjugacy Problem in Garside Groups". Conjugating by a simple ultra element does not decrease the infimum of the element and does not increase the canonical length. Also, the result of the conjugation belongs to the USS.

#### Returns:

a **triple** (p, 577) (R,M,P), where R is the resulting permutation, M is true if R is minimal, and P is a period of a conjugated braid.

#### 9.134.5.20 `set< pair< Permutation , int > > ThRightNormalForm::getSimpleUltraConjugators (int period)`

Compute all minimal simple ultra conjugators for "rep".

#### 9.134.5.21 `Permutation ThRightNormalForm::getTransport (const ThRightNormalForm & B, const Permutation & u) const`

(Aux, for USS simple) Compute a transport for  $*this$  and  $B = (*this)^u$  where  $p$  is a permutation. See Proposition 2.1 and Definition 2.4 of Gebhardt "A New Approach to the Conjugacy Problem in Garside Groups". It is important that braids  $*this$  and  $B = (*this)^u$  belong to their SSS. Also, it is important that  $*this$  is not a power of  $\Delta$  (USS is "trivial" for such elements).



#### 9.134.5.22 `set< Permutation > ThRightNormalForm::getTransports (const Permutation & u, int period) const`

(Aux, for USS simple) Compute a set of transports for a pair `*this` and `(*this)^u` where `p` is a permutation. F-set, see Definition 4.2 in Gebhardt, "A New Approach to the Conjugacy Problem in Garside Groups". It is important that braids `(*this)` and `(*this)^u` belong to their SSS. Also, it is important that `(*this)` is not a power of  $\Delta$  (USS is "trivial" for such elements).

#### 9.134.5.23 `Word ThRightNormalForm::getWord () const`

Computes a word represented by a normal form. For each permutation-braid computes a braid-word it represents and, finally, concatenate all the results.

#### 9.134.5.24 `ThRightNormalForm ThRightNormalForm::increaseRank (int N) const`

Increase the rank of the braid. The current braid does not change as an element of  $F_\infty$ . We simply increase the rank and recompute the normal form presentation. If  $N < \text{theRank}$  then output `*this`.

#### 9.134.5.25 `NF ThRightNormalForm::inverse () const [private]`

(Low level function) Invert a normal form.

Referenced by operator-().

#### 9.134.5.26 `bool ThRightNormalForm::isTrivial () const [inline]`

Check if a normal form is trivial.

Definition at line 199 of file ThRightNormalForm.h.

References theDecomposition, and theOmegaPower.

#### 9.134.5.27 `NF ThRightNormalForm::multiply (const ThRightNormalForm & rep) const [private]`

(Low level function) Multiply normal forms.

Referenced by operator\*(), and operator\*=(.).

**9.134.5.28 ThRightNormalForm::operator NF () const [inline]**

Cast operator (returns a representation of a normal form).

Definition at line 127 of file ThRightNormalForm.h.

References theDecomposition, theOmegaPower, and theRank.

**9.134.5.29 ThRightNormalForm::operator ThLeftNormalForm () const**

Cast operator.

**9.134.5.30 bool ThRightNormalForm::operator!= (const ThRightNormalForm & rep) const [inline]**

Compare (check if not equal).

Definition at line 142 of file ThRightNormalForm.h.

References theDecomposition, theOmegaPower, and theRank.

**9.134.5.31 ThRightNormalForm ThRightNormalForm::operator\* (const ThRightNormalForm & rep) const [inline]**

Multiply two normal forms.

Definition at line 169 of file ThRightNormalForm.h.

References multiply().

**9.134.5.32 ThRightNormalForm& ThRightNormalForm::operator\*= (const ThRightNormalForm & rep) [inline]**

Multiply a normal form by the other on the right.

Definition at line 174 of file ThRightNormalForm.h.

References multiply().

**9.134.5.33 ThRightNormalForm ThRightNormalForm::operator- () const [inline]**

Definition at line 164 of file ThRightNormalForm.h.

References inverse().

#### 9.134.5.34 **bool ThRightNormalForm::operator< (const ThRightNormalForm & rep) const [inline]**

Compare (check if less, performed componentwise).

Definition at line 150 of file ThRightNormalForm.h.

References theDecomposition, theOmegaPower, and theRank.

#### 9.134.5.35 **ThRightNormalForm& ThRightNormalForm::operator= (const NF & tr) [inline]**

Assignment operator.

Definition at line 110 of file ThRightNormalForm.h.

References triple< T1, T2, T3 >::first, triple< T1, T2, T3 >::second, theDecomposition, theOmegaPower, theRank, and triple< T1, T2, T3 >::third.

#### 9.134.5.36 **bool ThRightNormalForm::operator== (const ThRightNormalForm & rep) const [inline]**

Compare (check if equal).

Definition at line 133 of file ThRightNormalForm.h.

References theDecomposition, theOmegaPower, and theRank.

#### 9.134.5.37 **static ThRightNormalForm ThRightNormalForm::randomPositive (int rank, int decomp\_length) [static]**

Generate random positive normal form. Function returns a right normal form with canonical length decomp\_length (it can be smaller due to possible cancellations).

#### 9.134.5.38 **void ThRightNormalForm::setDecomposition (const list< Permutation > & d) [inline]**

Set a decomposition of a normal form (without any check of "greedy conditions").

Definition at line 425 of file ThRightNormalForm.h.

References theDecomposition.

#### 9.134.5.39 **void ThRightNormalForm::setPower (int p) [inline]**

Set a power of a half-twist.

Definition at line 423 of file ThRightNormalForm.h.

References theOmegaPower.

```
9.134.5.40 pair< bool , ThRightNormalForm >
ThRightNormalForm::sssConstructionIteration (map<
ThRightNormalForm, ThRightNormalForm > & sss_new1, map<
ThRightNormalForm, ThRightNormalForm > & sss_checked1,
const map< ThRightNormalForm, ThRightNormalForm > &
sss_new2, const map< ThRightNormalForm, ThRightNormalForm
> & sss_checked2) const [private]
```

I think function processes a vertex in a Super Summit Set graph.

```
9.134.5.41 static transformationResult ThRightNormalForm::transform (int
theIndex, Permutation & p1, Permutation & p2) [static,
private]
```

## 9.134.6 Member Data Documentation

```
9.134.6.1 list< Permutation > ThRightNormalForm::theDecomposition
[private]
```

Sequence of permutations.

Definition at line 469 of file ThRightNormalForm.h.

Referenced by adjust(), getDecomposition(), isTrivial(), operator NF(), operator!=(), operator<(), operator=(), operator==(), setDecomposition(), and ThRightNormalForm().

```
9.134.6.2 int ThRightNormalForm::theOmegaPower [private]
```

Power of omega.

Definition at line 466 of file ThRightNormalForm.h.

Referenced by adjust(), getPower(), isTrivial(), operator NF(), operator!=(), operator<(), operator=(), operator==(), setPower(), and ThRightNormalForm().

```
9.134.6.3 int ThRightNormalForm::theRank [private]
```

The rank of the braid group (number of strands).

Definition at line 463 of file ThRightNormalForm.h.

Referenced by `adjust()`, `getRank()`, `operator NF()`, `operator!=()`, `operator<()`, `operator=()`, `operator==()`, and `ThRightNormalForm()`.

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

- `BraidGroup/include/ThRightNormalForm.h`

## 9.135 Permutation::triple Struct Reference

### Public Member Functions

- **triple** (int p1=0, int p2=0, int p3=0)

### Public Attributes

- int **c1**
- int **c2**
- int **c3**

#### 9.135.1 Detailed Description

Definition at line 288 of file Permutation.h.

#### 9.135.2 Constructor & Destructor Documentation

##### 9.135.2.1 Permutation::triple::triple (int *p1* = 0, int *p2* = 0, int *p3* = 0) [inline]

Definition at line 289 of file Permutation.h.

#### 9.135.3 Member Data Documentation

##### 9.135.3.1 int Permutation::triple::c1

Definition at line 290 of file Permutation.h.

##### 9.135.3.2 int Permutation::triple::c2

Definition at line 290 of file Permutation.h.

##### 9.135.3.3 int Permutation::triple::c3

Definition at line 290 of file Permutation.h.

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

- general/include/**Permutation.h**

## 9.136 `triple< T1, T2, T3 >` Class Template Reference

```
#include <tuples.h>
```

### Public Member Functions

- **`triple`** (const T1 &t1=T1(), const T2 &t2=T2(), const T3 &t3=T3())
- bool **`operator==`** (const **`triple`** &t) const
- bool **`operator!=`** (const **`triple`** &t) const
- bool **`operator>`** (const **`triple`** &t) const
- bool **`operator<`** (const **`triple`** &t) const

### Public Attributes

- T1 **`first`**
- T2 **`second`**
- T3 **`third`**

### Friends

- ostream & **`operator<<`** (ostream &os, const **`triple`** &t)

#### 9.136.1 Detailed Description

```
template<class T1, class T2, class T3> class triple< T1, T2, T3 >
```

Definition at line 17 of file tuples.h.

#### 9.136.2 Constructor & Destructor Documentation

**9.136.2.1** `template<class T1, class T2, class T3> triple< T1, T2, T3 >::triple`  
(const T1 &*t1* = T1 (), const T2 &*t2* = T2 (), const T3 &*t3* = T3 ())  
[**`inline`**]

Definition at line 28 of file tuples.h.

### 9.136.3 Member Function Documentation

**9.136.3.1** `template<class T1, class T2, class T3> bool triple< T1, T2, T3 >::operator!=(const triple< T1, T2, T3 > & t) const [inline]`

Definition at line 43 of file tuples.h.

**9.136.3.2** `template<class T1, class T2, class T3> bool triple< T1, T2, T3 >::operator<(const triple< T1, T2, T3 > & t) const [inline]`

Definition at line 63 of file tuples.h.

**9.136.3.3** `template<class T1, class T2, class T3> bool triple< T1, T2, T3 >::operator==(const triple< T1, T2, T3 > & t) const [inline]`

Definition at line 39 of file tuples.h.

**9.136.3.4** `template<class T1, class T2, class T3> bool triple< T1, T2, T3 >::operator>(const triple< T1, T2, T3 > & t) const [inline]`

Definition at line 47 of file tuples.h.

### 9.136.4 Friends And Related Function Documentation

**9.136.4.1** `template<class T1, class T2, class T3> ostream& operator<<(ostream & os, const triple< T1, T2, T3 > & t) [friend]`

Definition at line 87 of file tuples.h.

### 9.136.5 Member Data Documentation

**9.136.5.1** `template<class T1, class T2, class T3> T1 triple< T1, T2, T3 >::first`

Definition at line 100 of file tuples.h.

Referenced by `triple< Word, Word, Word >::operator!=()`, `triple< Word, Word, Word >::operator<()`, `ThRightNormalForm::operator=()`, `ThLeftNormalForm::operator=()`, `BKLRightNormalForm::operator=()`, `BKLLeftNormalForm::operator=()`, `triple< Word, Word, Word >::operator==(())`, and `triple< Word, Word, Word >::operator>()`.



**9.136.5.2** `template<class T1, class T2, class T3> T2 triple< T1, T2, T3 >::second`

Definition at line 101 of file `tuples.h`.

Referenced by `triple< Word, Word, Word >::operator!=()`, `triple< Word, Word, Word >::operator<()`, `ThRightNormalForm::operator=()`, `ThLeftNormalForm::operator=()`, `BKLRightNormalForm::operator=()`, `BKLLeftNormalForm::operator=()`, `triple< Word, Word, Word >::operator==()`, and `triple< Word, Word, Word >::operator>()`.

**9.136.5.3** `template<class T1, class T2, class T3> T3 triple< T1, T2, T3 >::third`

Definition at line 102 of file `tuples.h`.

Referenced by `triple< Word, Word, Word >::operator!=()`, `triple< Word, Word, Word >::operator<()`, `ThRightNormalForm::operator=()`, `ThLeftNormalForm::operator=()`, `BKLRightNormalForm::operator=()`, `BKLLeftNormalForm::operator=()`, `triple< Word, Word, Word >::operator==()`, and `triple< Word, Word, Word >::operator>()`.

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

- `general/include/tuples.h`

## 9.137 TripleDecompositionProtocolInstance Class Reference

Definition of the class **TripleDecompositionProtocolInstance** (p. 580).

```
#include <TripleDecompositionKeyGeneration.h>
```

### Public Member Functions

- **TripleDecompositionProtocolInstance** (int braid\_rank, **quadruple**< **Word**, **Word**, **Word**, **Word** > conjugators, **quintuple**< **Word**, **Word**, **Word**, **Word**, **Word** > privateKeyA, **quintuple**< **Word**, **Word**, **Word**, **Word**, **Word** > privateKeyB)  
*Create an instance of the protocol for a particular choice of certain public/private information.*
- int **getBraidRank** () const  
*(accessor function) Get the rank of the braid group.*
- **quintuple**< **Word**, **Word**, **Word**, **Word**, **Word** > **getPrivateKeyA** () const  
*(accessor function) Get the private key of the first party (Alice).*
- **quintuple**< **Word**, **Word**, **Word**, **Word**, **Word** > **getPrivateKeyB** () const  
*(accessor function) Get the private key of the second party (Bob).*
- **triple**< **Word**, **Word**, **Word** > **getPublicKeyA** () const  
*(accessor function) Get the public key of the first party (Alice).*
- **triple**< **Word**, **Word**, **Word** > **getPublicKeyB** () const  
*(accessor function) Get the public key of the second party (Bob).*
- **ThRightNormalForm** **getSharedKey** () const  
*(accessor function) Get the shared key of two parties.*

### Static Public Member Functions

- static **TripleDecompositionProtocolInstance** **random** (int braid\_rank, int baseLenth, int keyLength)  
*Generate a random instance of the protocol.*

## Static Private Member Functions

- static **Word randomWord** (int lowerIndex, int upperIndex, int len)  
*Generate a random reduced word over the alphabet  $x_l, \dots, x_u$  of length len.*

## Private Attributes

- int **theRank**  
*the rank of the braid group*
- quintuple< **Word, Word, Word, Word, Word** > **thePrivateKeyA**  
*The private key of the first party (Alice).*
- quintuple< **Word, Word, Word, Word, Word** > **thePrivateKeyB**  
*The private key of the second party (Bob).*
- triple< **Word, Word, Word** > **thePublicKeyA**  
*The public key of the first party (Alice).*
- triple< **Word, Word, Word** > **thePublicKeyB**  
*The public key of the second party (Bob).*
- quadruple< **Word, Word, Word, Word** > **theConjugators**  
*The conjugators for the elementary commuting subgroups (public information).*
- **ThRightNormalForm theSharedKey**  
*The shared key  $a_1b_1a_2b_2a_3b_3$ .*

### 9.137.1 Detailed Description

Definition of the class **TripleDecompositionProtocolInstance** (p.580). Objects of this class contain public and private information of two parties as used in Kurt' key exchange protocol. Also it contains a routine for random generation of keys as proposed in Y. Kurt, J. Koh, "A New Key Exchange Primitive Based on the Triple Decomposition Problem".

Definition at line 34 of file TripleDecompositionKeyGeneration.h.

## 9.137.2 Constructor & Destructor Documentation

### 9.137.2.1 TripleDecompositionProtocolInstance::TripleDecompositionProtocolInstance (int *braid\_rank*, quadruple< Word, Word, Word, Word > *conjugators*, quintuple< Word, Word, Word, Word, Word > *privateKeyA*, quintuple< Word, Word, Word, Word, Word > *privateKeyB*)

Create an instance of the protocol for a particular choice of certain public/private information.

## 9.137.3 Member Function Documentation

### 9.137.3.1 int TripleDecompositionProtocolInstance::getBraidRank () const [inline]

(accessor function) Get the rank of the braid group.

Definition at line 69 of file TripleDecompositionKeyGeneration.h.

References theRank.

### 9.137.3.2 quintuple< Word , Word , Word , Word , Word > TripleDecompositionProtocolInstance::getPrivateKeyA () const [inline]

(accessor function) Get the private key of the first party (Alice).

Definition at line 73 of file TripleDecompositionKeyGeneration.h.

References thePrivateKeyA.

### 9.137.3.3 quintuple< Word , Word , Word , Word , Word > TripleDecompositionProtocolInstance::getPrivateKeyB () const [inline]

(accessor function) Get the private key of the second party (Bob).

Definition at line 75 of file TripleDecompositionKeyGeneration.h.

References thePrivateKeyB.

#### 9.137.3.4 `triple< Word , Word , Word > TripleDecompositionProtocolInstance::getPublicKeyA () const [inline]`

(accessor function) Get the public key of the first party (Alice).

Definition at line 79 of file TripleDecompositionKeyGeneration.h.

References thePublicKeyA.

#### 9.137.3.5 `triple< Word , Word , Word > TripleDecompositionProtocolInstance::getPublicKeyB () const [inline]`

(accessor function) Get the public key of the second party (Bob).

Definition at line 81 of file TripleDecompositionKeyGeneration.h.

References thePublicKeyB.

#### 9.137.3.6 `ThRightNormalForm TripleDecompositionProtocolInstance::getSharedKey () const [inline]`

(accessor function) Get the shared key of two parties.

Definition at line 85 of file TripleDecompositionKeyGeneration.h.

References theSharedKey.

#### 9.137.3.7 `static TripleDecompositionProtocolInstance TripleDecompositionProtocolInstance::random (int braid_rank, int baseLenth, int keyLength) [static]`

Generate a random instance of the protocol.

##### Parameters:

*braid\_rank* - rank of the braid group;

*baseLenth* - the length of thePublicKey.first

*keyLength* - the length of thePrivateKey

#### 9.137.3.8 `static Word TripleDecompositionProtocolInstance::randomWord (int lowerIndex, int upperIndex, int len) [static, private]`

Generate a random reduced word over the alphabet  $x_l, \dots, x_u$  of length len.

## 9.137.4 Member Data Documentation

### 9.137.4.1 `quadruple< Word , Word , Word , Word >` **`TripleDecompositionProtocolInstance::theConjugators`** **[private]**

The conjugators for the elementary commuting subgroups (public information). The conjugators  $s_1, s_2, s_3, s_4$ . See notation of Y. Kurt, J. Koh, "A New Key Exchange Primitive Based on the Triple Decomposition Problem".

Definition at line 147 of file `TripleDecompositionKeyGeneration.h`.

### 9.137.4.2 `quintuple< Word , Word , Word , Word , Word >` **`TripleDecompositionProtocolInstance::thePrivateKeyA`** **[private]**

The private key of the first party (Alice). The private key  $Private_A = (a_1, a_2, a_3, x_1, x_2)$ . See notation of Y. Kurt, J. Koh, "A New Key Exchange Primitive Based on the Triple Decomposition Problem".

Definition at line 116 of file `TripleDecompositionKeyGeneration.h`.

Referenced by `getPrivateKeyA()`.

### 9.137.4.3 `quintuple< Word , Word , Word , Word , Word >` **`TripleDecompositionProtocolInstance::thePrivateKeyB`** **[private]**

The private key of the second party (Bob). The private key  $Private_B = (b_1, b_2, b_3, y_1, y_2)$ . See notation of Y. Kurt, J. Koh, "A New Key Exchange Primitive Based on the Triple Decomposition Problem".

Definition at line 124 of file `TripleDecompositionKeyGeneration.h`.

Referenced by `getPrivateKeyB()`.

### 9.137.4.4 `triple< Word , Word , Word >` **`TripleDecompositionProtocolInstance::thePublicKeyA`** **[private]**

The public key of the first party (Alice). The public key  $Public_A = (u, v, w)$ . See notation of Y. Kurt, J. Koh, "A New Key Exchange Primitive Based on the Triple Decomposition Problem".

Definition at line 132 of file `TripleDecompositionKeyGeneration.h`.

Referenced by `getPublicKeyA()`.

**9.137.4.5 triple< Word , Word , Word > TripleDecompositionProtocolInstance::thePublicKeyB [private]**

The public key of the second party (Bob). The public key  $Public_B = (p, q, r)$ . See notation of Y. Kurt, J. Koh, "A New Key Exchange Primitive Based on the Triple Decomposition Problem".

Definition at line 139 of file TripleDecompositionKeyGeneration.h.

Referenced by getPublicKeyB().

**9.137.4.6 int TripleDecompositionProtocolInstance::theRank [private]**

the rank of the braid group

Definition at line 108 of file TripleDecompositionKeyGeneration.h.

Referenced by getBraidRank().

**9.137.4.7 ThRightNormalForm TripleDecompositionProtocolInstance::theSharedKey [private]**

The shared key  $a_1 b_1 a_2 b_2 a_3 b_3$ .

Definition at line 151 of file TripleDecompositionKeyGeneration.h.

Referenced by getSharedKey().

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

- CryptoTripleDecomposition/include/TripleDecompositionKeyGeneration.h

## 9.138 TTP\_Conf Struct Reference

Set of parameters required to construct the protocol instance.

```
#include <AEProtocol.h>
```

### Public Attributes

- int **nBL**
- int **nBR**
- int **N**
- int **nGamma**
- int **len\_z**
- int **len\_w**

### Friends

- ostream & **operator**<< (ostream &out, const **TTP\_Conf** &ttp\_conf)

### 9.138.1 Detailed Description

Set of parameters required to construct the protocol instance.

Definition at line 41 of file AEProtocol.h.

### 9.138.2 Friends And Related Function Documentation

#### 9.138.2.1 ostream& operator<< (ostream & out, const TTP\_Conf & ttp\_conf) [friend]

Definition at line 51 of file AEProtocol.h.

### 9.138.3 Member Data Documentation

#### 9.138.3.1 int TTP\_Conf::len\_w

Definition at line 49 of file AEProtocol.h.

#### 9.138.3.2 int TTP\_Conf::len\_z

Definition at line 48 of file AEProtocol.h.



**9.138.3.3 int TTP\_Conf::N**

Definition at line 45 of file AEProtocol.h.

**9.138.3.4 int TTP\_Conf::nBL**

Definition at line 43 of file AEProtocol.h.

**9.138.3.5 int TTP\_Conf::nBR**

Definition at line 44 of file AEProtocol.h.

**9.138.3.6 int TTP\_Conf::nGamma**

Definition at line 46 of file AEProtocol.h.

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

- CryptoAE/include/AEProtocol.h

## 9.139 TTPAttack Class Reference

This is an implementation of an attack on TTP algorithm for generating public sets of generators of the Algebraic Eraser protocol.

```
#include <TTPAttack.h>
```

### Public Types

- enum **TTP\_Result** { **TTP\_FAILED**, **TTP\_NOTSURE**, **TTP\_SUCCESSFULL** }

### Public Member Functions

- **TTPAttack** (int n, BSets bs)  
*Constructor.*
- bool **run** (const **TTPTuple** &d)  
*Excutes te attack.*

### Private Member Functions

- void **printStats** (const **ThLeftNormalForm** &nf, ostream &out)
- **ThLeftNormalForm** **cycleDecycle** (const **ThLeftNormalForm** &nf)
- bool **simpleLBA** (int NWL, int NWR, const vector< **ThLeftNormalForm** > &theTuple, const **Word** &z)
- bool **LBA** (int NWL, int NWR, const vector< **ThLeftNormalForm** > &theTuple, const **Word** &z)
- bool **oneOfSSSReps** (int N1, int N2, const vector< **ThLeftNormalForm** > &theTuple)
- void **reduceDeltaLBA** (vector< **ThLeftNormalForm** > &theTuple)

### Private Attributes

- int **N**
- BSets **BS**

### 9.139.1 Detailed Description

This is an implementation of an attack on TTP algorithm for generating public sets of generators of the Algebraic Eraser protocol.

Definition at line 59 of file TTPAttack.h.

### 9.139.2 Member Enumeration Documentation

#### 9.139.2.1 enum TTPAttack::TTP\_Result

Enumerator:

*TTP\_FAILED*

*TTP\_NOTSURE*

*TTP\_SUCCESSFULL*

Definition at line 68 of file TTPAttack.h.

### 9.139.3 Constructor & Destructor Documentation

#### 9.139.3.1 TTPAttack::TTPAttack (int *n*, BSets *bs*) [inline]

Constructor.

Parameters:

*n* - group rank (number of braids

*bs* -the initial commuting sets of subgroup generators

Definition at line 66 of file TTPAttack.h.

### 9.139.4 Member Function Documentation

**9.139.4.1** `ThLeftNormalForm TTPAttack::cycleDecycle (const ThLeftNormalForm & nf) [private]`

**9.139.4.2** `bool TTPAttack::LBA (int NWL, int NWR, const vector< ThLeftNormalForm > & theTuple, const Word & z) [private]`

**9.139.4.3** `bool TTPAttack::oneOfSSSReps (int N1, int N2, const vector< ThLeftNormalForm > & theTuple) [private]`

**9.139.4.4** `void TTPAttack::printStats (const ThLeftNormalForm & nf, ostream & out) [inline, private]`

Definition at line 87 of file TTPAttack.h.

References `ThLeftNormalForm::getDecomposition()`, and `ThLeftNormalForm::getPower()`.

**9.139.4.5** `void TTPAttack::reduceDeltaLBA (vector< ThLeftNormalForm > & theTuple) [private]`

**9.139.4.6** `bool TTPAttack::run (const TTPTuple & d)`

Excutes te attack.

#### Parameters:

*d* - the output of TTP algorithm

**9.139.4.7** `bool TTPAttack::simpleLBA (int NWL, int NWR, const vector< ThLeftNormalForm > & theTuple, const Word & z) [private]`

### 9.139.5 Member Data Documentation

**9.139.5.1** `BSets TTPAttack::BS [private]`

Definition at line 101 of file TTPAttack.h.

**9.139.5.2** `int TTPAttack::N [private]`

Definition at line 100 of file TTPAttack.h.

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

- `CryptoAE/include/TTPAttack.h`

## 9.140 TTPLBA Class Reference

```
#include <TTPAttack.h>
```

### Public Member Functions

- **TTPLBA** ()
- bool **reduce** (int N, const **BSets** &bs, const **TPTuple** &theTuple, const vector< **Word** > &gens, int sec, ostream &out, **TPTuple** &red\_T, const **Word** &z)
- bool **simpleLBA** (int N, const **BSets** &bs, const **TPTuple** &theTuple, const **Word** &z, **TPTuple** \*ret\_T=NULL)

### Private Member Functions

- void **addNewElt** (const **TPTuple** &T, const set< **NODE** > &checkedElements, set< **NODE** > &uncheckedElements)
- void **tryNode** (int N, **NODE** cur, const vector< **Word** > &gens, const set< **NODE** > &checkedElements, set< **NODE** > &uncheckedElements, int &min\_weight)

### Private Attributes

- **TPTuple** savTuple

#### 9.140.1 Detailed Description

Definition at line 35 of file TTPAttack.h.

#### 9.140.2 Constructor & Destructor Documentation

##### 9.140.2.1 TTPLBA::TTPLBA () [inline]

Definition at line 38 of file TTPAttack.h.

### 9.140.3 Member Function Documentation

- 9.140.3.1** void TTPLBA::addNewElt (const TTPTuple & *T*, const set< NODE > & *checkedElements*, set< NODE > & *uncheckedElements*)  
[private]
- 9.140.3.2** bool TTPLBA::reduce (int *N*, const BSets & *bs*, const TTPTuple & *theTuple*, const vector< Word > & *gens*, int *sec*, ostream & *out*, TTPTuple & *red\_T*, const Word & *z*)
- 9.140.3.3** bool TTPLBA::simpleLBA (int *N*, const BSets & *bs*, const TTPTuple & *theTuple*, const Word & *z*, TTPTuple \* *ret\_T* = NULL)
- 9.140.3.4** void TTPLBA::tryNode (int *N*, NODE *cur*, const vector< Word > & *gens*, const set< NODE > & *checkedElements*, set< NODE > & *uncheckedElements*, int & *min\_weight*) [private]

### 9.140.4 Member Data Documentation

- 9.140.4.1** TTPTuple TTPLBA::savTuple [private]

Definition at line 48 of file TTPAttack.h.

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

- CryptoAE/include/TTPAttack.h

## 9.141 TPTuple Class Reference

Implements tuples corresponding to the putput of TTP algorithm.

```
#include <AEProtocol.h>
```

### Public Member Functions

- **TPTuple** ()
- **TPTuple** (const vector< **Word** > &L, const vector< **Word** > &R)
- **TPTuple** (const vector< **Word** > &L, const vector< **Word** > &R, const **Word** &conj)
- int **length** () const  
*Returns the total length of the tuples.*
- void **shorten** (int N)  
*Shorten words in each tuple.*
- bool **testTuples** (int N, bool details) const  
*Test tuples for being "seprated", i.e. two nonintersecting sets of commuting generators.*
- bool **shortAndTestTuples** (int N, bool details=false)  
*Performs shorten and then test on being "separated".*

### Public Attributes

- vector< **Word** > **WL**
- vector< **Word** > **WR**
- **Word** **z**

### Private Attributes

- vector< **Word** > **origWL**
- vector< **Word** > **origWR**

### Friends

- bool **operator**< (const **TPTuple** &t1, const **TPTuple** &t2)  
*Tuple ordering operator.*



### 9.141.1 Detailed Description

Implements tuples corresponding to the putput of TTP algorithm.

Definition at line 108 of file AEProtocol.h.

### 9.141.2 Constructor & Destructor Documentation

#### 9.141.2.1 TPTuple::TPTuple () [inline]

Definition at line 111 of file AEProtocol.h.

#### 9.141.2.2 TPTuple::TPTuple (const vector< Word > & L, const vector< Word > & R) [inline]

Definition at line 112 of file AEProtocol.h.

#### 9.141.2.3 TPTuple::TPTuple (const vector< Word > & L, const vector< Word > & R, const Word & conj) [inline]

Definition at line 116 of file AEProtocol.h.

### 9.141.3 Member Function Documentation

#### 9.141.3.1 int TPTuple::length () const

Returns the total length of the tuples.

#### 9.141.3.2 bool TPTuple::shortAndTestTuples (int N, bool details = false) [inline]

Performs shorten and then test on being "separated".

Definition at line 137 of file AEProtocol.h.

References shorten(), and testTuples().

#### 9.141.3.3 void TPTuple::shorten (int N)

Shorten words in each tuple.

Referenced by shortAndTestTuples().

#### 9.141.3.4 `bool TTPTuple::testTuples (int N, bool details) const`

Test tuples for being "seprated", i.e. two nonintersecting sets of commuting generators.

##### Parameters:

*N* - braid group rank

*details* - if true will print verbose information

Referenced by `shortAndTestTuples()`.

### 9.141.4 Friends And Related Function Documentation

#### 9.141.4.1 `bool operator< (const TTPTuple & t1, const TTPTuple & t2)` `[friend]`

Tuple ordering operator.

Definition at line 143 of file `AEProtocol.h`.

### 9.141.5 Member Data Documentation

#### 9.141.5.1 `vector<Word> TTPTuple::origWL` `[private]`

Definition at line 150 of file `AEProtocol.h`.

#### 9.141.5.2 `vector<Word> TTPTuple::origWR` `[private]`

Definition at line 151 of file `AEProtocol.h`.

#### 9.141.5.3 `vector<Word> TTPTuple::WL`

Definition at line 121 of file `AEProtocol.h`.

Referenced by `AEKeyExchange::alicePublicKey()`.

#### 9.141.5.4 `vector<Word> TTPTuple::WR`

Definition at line 122 of file `AEProtocol.h`.

Referenced by `AEKeyExchange::bobPublicKey()`.

#### 9.141.5.5 Word TPTuple::z

Definition at line 148 of file AEProtocol.h.

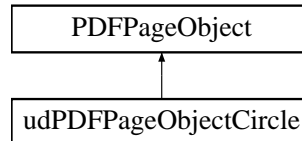
The documentation for this class was generated from the following file:

- CryptoAE/include/AEProtocol.h

## 9.142 udPDFPageObjectCircle Class Reference

Class implements a square pdf object.

#include <PDFgraphing.h> Inheritance diagram for udPDFPageObjectCircle::



### Public Member Functions

- **udPDFPageObjectCircle** (double *cx*, double *cy*, double *r*, double *bc*=0, double *f*=-1)  
*Constructor for a circle pdf object.*
- void **write** (ostream &os)

### Private Attributes

- double **theX**
- double **theY**
- double **theRad**
- double **theFill**
- double **border\_color**

#### 9.142.1 Detailed Description

Class implements a square pdf object.

Definition at line 327 of file PDFgraphing.h.

#### 9.142.2 Constructor & Destructor Documentation

**9.142.2.1 udPDFPageObjectCircle::udPDFPageObjectCircle** (double *cx*, double *cy*, double *r*, double *bc* = 0, double *f* = -1) [**inline**]

Constructor for a circle pdf object.

**Parameters:**

*cx,cy* - coordinates of the center of a circle

*r* - radius of a circle

*bc* - border color from 0 to 1

*f* - fill color. If  $f < 0$  then no filling

Definition at line 344 of file PDFgraphing.h.

**9.142.3 Member Function Documentation****9.142.3.1 void udPDFPageObjectCircle::write (ostream & os) [virtual]**

Reimplemented from **PDFPageObject** (p. 365).

**9.142.4 Member Data Documentation****9.142.4.1 double udPDFPageObjectCircle::border\_color [private]**

Definition at line 367 of file PDFgraphing.h.

**9.142.4.2 double udPDFPageObjectCircle::theFill [private]**

Definition at line 366 of file PDFgraphing.h.

**9.142.4.3 double udPDFPageObjectCircle::theRad [private]**

Definition at line 365 of file PDFgraphing.h.

**9.142.4.4 double udPDFPageObjectCircle::theX [private]**

Definition at line 364 of file PDFgraphing.h.

**9.142.4.5 double udPDFPageObjectCircle::theY [private]**

Definition at line 364 of file PDFgraphing.h.

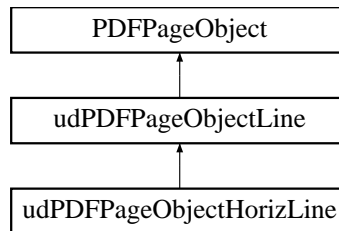
The documentation for this class was generated from the following file:

- Graphics/include/PDFgraphing.h

## 9.143 udPDFPageObjectHorizLine Class Reference

Class implements a horizontal line pdf object.

#include <PDFgraphing.h> Inheritance diagram for udPDFPageObjectHorizLine::



### Public Member Functions

- **udPDFPageObjectHorizLine** (double x, double y, double s, double gc=0, bool d=false)

*Constructor for horizontal line pdf object.*

### 9.143.1 Detailed Description

Class implements a horizontal line pdf object.

Definition at line 223 of file PDFgraphing.h.

### 9.143.2 Constructor & Destructor Documentation

#### 9.143.2.1 udPDFPageObjectHorizLine::udPDFPageObjectHorizLine (double x, double y, double s, double gc = 0, bool d = false) [inline]

Constructor for horizontal line pdf object.

#### Parameters:

- x,y** - coordinates of the left end point
- s** - length
- gc** - gray scale intensity (0 - black, 1 - white)
- d** - if true than a dashed line is drawn

Definition at line 240 of file PDFgraphing.h.

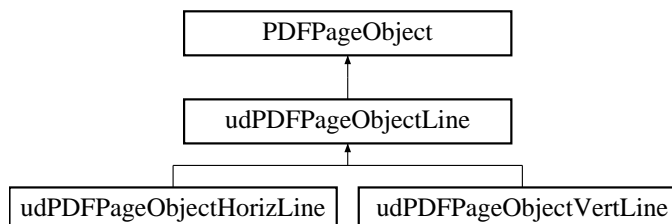
The documentation for this class was generated from the following file:

- Graphics/include/**PDFgraphing.h**

## 9.144 udPDFPageObjectLine Class Reference

Class implements pdf line object.

#include <PDFgraphing.h> Inheritance diagram for udPDFPageObjectLine::



### Public Member Functions

- **udPDFPageObjectLine** (double x1, double y1, double x2, double y2, double gc=0, bool d=false)

*Constructor for pdf line object.*

- void **write** (ostream &os)

### Private Attributes

- double **theX1**
- double **theY1**
- double **theX2**
- double **theY2**
- double **gray\_color**
- bool **dashed**

#### 9.144.1 Detailed Description

Class implements pdf line object.

Definition at line 148 of file PDFgraphing.h.



## 9.144.2 Constructor & Destructor Documentation

**9.144.2.1** `udPDFPageObjectLine::udPDFPageObjectLine (double x1, double y1, double x2, double y2, double gc = 0, bool d = false) [inline]`

Constructor for pdf line object.

### Parameters:

*x1,y1,x2,y2* - coordinates of the two endpoints of a line  
*gc* - gray scale intensity (0 - black, 1 - white)  
*d* - if true than a dashed line is drawn

Definition at line 163 of file PDFgraphing.h.

## 9.144.3 Member Function Documentation

**9.144.3.1** `void udPDFPageObjectLine::write (ostream & os) [virtual]`

Reimplemented from `PDFPageObject` (p. 365).

## 9.144.4 Member Data Documentation

**9.144.4.1** `bool udPDFPageObjectLine::dashed [private]`

Definition at line 186 of file PDFgraphing.h.

**9.144.4.2** `double udPDFPageObjectLine::gray_color [private]`

Definition at line 185 of file PDFgraphing.h.

**9.144.4.3** `double udPDFPageObjectLine::theX1 [private]`

Definition at line 183 of file PDFgraphing.h.

**9.144.4.4** `double udPDFPageObjectLine::theX2 [private]`

Definition at line 184 of file PDFgraphing.h.

**9.144.4.5 double udPDFPageObjectLine::theY1 [private]**

Definition at line 183 of file PDFgraphing.h.

**9.144.4.6 double udPDFPageObjectLine::theY2 [private]**

Definition at line 184 of file PDFgraphing.h.

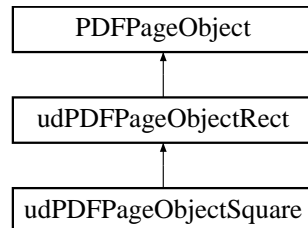
The documentation for this class was generated from the following file:

- Graphics/include/PDFgraphing.h

## 9.145 udPDFPageObjectRect Class Reference

Class implements a rectangle pdf object.

`#include <PDFgraphing.h>`Inheritance diagram for udPDFPageObjectRect::



### Public Member Functions

- **udPDFPageObjectRect** (double x, double y, double w, double h, double bc=0, double f=-1)

*Constructor for a rectangle pdf object.*

- void **write** (ostream &os)

### Private Attributes

- double **theX**
- double **theY**
- double **theW**
- double **theH**
- double **theFill**
- double **border\_color**

#### 9.145.1 Detailed Description

Class implements a rectangle pdf object.

Definition at line 250 of file PDFgraphing.h.

## 9.145.2 Constructor & Destructor Documentation

### 9.145.2.1 `udPDFPageObjectRect::udPDFPageObjectRect (double x, double y, double w, double h, double bc = 0, double f = -1) [inline]`

Constructor for a rectangle pdf object.

#### Parameters:

- x,y* - coordinates of the left end point
- w,h* - width and height of the rectangle
- bc* - border color from 0 to 1
- f* - fill color. If  $f < 0$  then no filling

Definition at line 267 of file PDFgraphing.h.

## 9.145.3 Member Function Documentation

### 9.145.3.1 `void udPDFPageObjectRect::write (ostream & os) [virtual]`

Reimplemented from `PDFPageObject` (p. 365).

## 9.145.4 Member Data Documentation

### 9.145.4.1 `double udPDFPageObjectRect::border_color [private]`

Definition at line 290 of file PDFgraphing.h.

### 9.145.4.2 `double udPDFPageObjectRect::theFill [private]`

Definition at line 289 of file PDFgraphing.h.

### 9.145.4.3 `double udPDFPageObjectRect::theH [private]`

Definition at line 288 of file PDFgraphing.h.

### 9.145.4.4 `double udPDFPageObjectRect::theW [private]`

Definition at line 288 of file PDFgraphing.h.

**9.145.4.5 double udPDFPageObjectRect::theX [private]**

Definition at line 287 of file PDFgraphing.h.

**9.145.4.6 double udPDFPageObjectRect::theY [private]**

Definition at line 287 of file PDFgraphing.h.

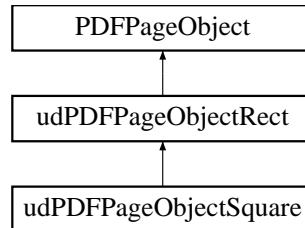
The documentation for this class was generated from the following file:

- Graphics/include/PDFgraphing.h

## 9.146 udPDFPageObjectSquare Class Reference

Class implements a square pdf object.

#include <PDFgraphing.h> Inheritance diagram for udPDFPageObjectSquare::



### Public Member Functions

- **udPDFPageObjectSquare** (double *x*, double *y*, double *s*, double *bc*=0, double *f*=-1)

*Constructor for a square pdf object.*

### 9.146.1 Detailed Description

Class implements a square pdf object.

Definition at line 299 of file PDFgraphing.h.

### 9.146.2 Constructor & Destructor Documentation

#### 9.146.2.1 udPDFPageObjectSquare::udPDFPageObjectSquare (double *x*, double *y*, double *s*, double *bc* = 0, double *f* = -1) [inline]

Constructor for a square pdf object.

#### Parameters:

- x,y* - coordinates of the left end point
- s* - length of the sides
- bc* - border color from 0 to 1
- f* - fill color. If  $f < 0$  then no filling

Definition at line 316 of file PDFgraphing.h.

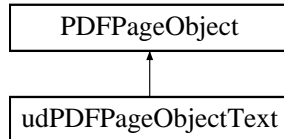
The documentation for this class was generated from the following file:

- Graphics/include/**PDFgraphing.h**

## 9.147 udPDFPageObjectText Class Reference

Class for pdf text object.

#include <PDFgraphing.h> Inheritance diagram for udPDFPageObjectText::



### Public Member Functions

- **udPDFPageObjectText** (double x, double y, const string &t, int s=12)  
*Constructor of text pdf object.*
- void **write** (ostream &os)

### Private Attributes

- double **theX**
- double **theY**
- string **text**
- int **tSize**

#### 9.147.1 Detailed Description

Class for pdf text object.

Definition at line 56 of file PDFgraphing.h.

#### 9.147.2 Constructor & Destructor Documentation

##### 9.147.2.1 udPDFPageObjectText::udPDFPageObjectText (double x, double y, const string &t, int s = 12) [inline]

Constructor of text pdf object.

#### Parameters:

**x,y** - coordinates where text will be placed



*t* - text to be output  
*s* - text size (default 12)

Definition at line 72 of file PDFgraphing.h.

### 9.147.3 Member Function Documentation

#### 9.147.3.1 void udPDFPageObjectText::write (ostream & *os*) [virtual]

Reimplemented from **PDFPageObject** (p. 365).

### 9.147.4 Member Data Documentation

#### 9.147.4.1 string udPDFPageObjectText::text [private]

Definition at line 93 of file PDFgraphing.h.

#### 9.147.4.2 double udPDFPageObjectText::theX [private]

Definition at line 92 of file PDFgraphing.h.

#### 9.147.4.3 double udPDFPageObjectText::theY [private]

Definition at line 92 of file PDFgraphing.h.

#### 9.147.4.4 int udPDFPageObjectText::tSize [private]

Definition at line 94 of file PDFgraphing.h.

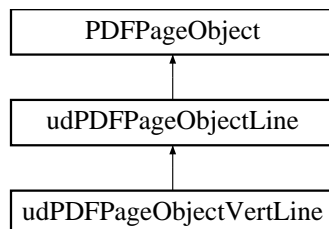
The documentation for this class was generated from the following file:

- Graphics/include/PDFgraphing.h

## 9.148 udPDFPageObjectVertLine Class Reference

Class implements a vertical line pdf object.

`#include <PDFgraphing.h>` Inheritance diagram for udPDFPageObjectVertLine::



### Public Member Functions

- **udPDFPageObjectVertLine** (double *x*, double *y*, double *s*, double *gc*=0, bool *d*=false)

*Constructor for vertical line pdf object.*

#### 9.148.1 Detailed Description

Class implements a vertical line pdf object.

Definition at line 195 of file PDFgraphing.h.

#### 9.148.2 Constructor & Destructor Documentation

##### 9.148.2.1 udPDFPageObjectVertLine::udPDFPageObjectVertLine (double *x*, double *y*, double *s*, double *gc* = 0, bool *d* = false) [inline]

Constructor for vertical line pdf object.

##### Parameters:

*x,y* - coordinates of the top end point

*s* - length

*gc* - gray scale intensity (0 - black, 1 - white)

*d* - if true than a dashed line is drawn

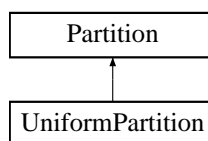
Definition at line 212 of file PDFgraphing.h.

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

- Graphics/include/**PDFgraphing.h**

## 9.149 UniformPartition Class Reference

`#include <DCBraidReduction.h>`Inheritance diagram for UniformPartition::



### Public Member Functions

- **UniformPartition ()**
- `list< int > getPartition (const Word &w)`

#### 9.149.1 Detailed Description

Definition at line 31 of file DCBraidReduction.h.

#### 9.149.2 Constructor & Destructor Documentation

##### 9.149.2.1 UniformPartition::UniformPartition () [inline]

Definition at line 34 of file DCBraidReduction.h.

#### 9.149.3 Member Function Documentation

##### 9.149.3.1 list<int> UniformPartition::getPartition (const Word & w) [inline, virtual]

Implements **Partition** (p. 357).

Definition at line 35 of file DCBraidReduction.h.

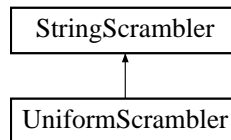
References Word::length().

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

- Experiments/include/DCBraidReduction.h

## 9.150 UniformScrambler Class Reference

#include <StringScramblers.h> Inheritance diagram for UniformScrambler::



### Public Member Functions

- **UniformScrambler** (int *gs*, double *prob*)
- **Word scramble** (const **Word** &) const
- double **fracChanged** () const

### Private Attributes

- double **theChangeProb**
- double **theFracChanged**
- int **numGens**

### 9.150.1 Detailed Description

Definition at line 40 of file StringScramblers.h.

### 9.150.2 Constructor & Destructor Documentation

#### 9.150.2.1 UniformScrambler::UniformScrambler (int *gs*, double *prob*) [inline]

Definition at line 43 of file StringScramblers.h.

### 9.150.3 Member Function Documentation

#### 9.150.3.1 double UniformScrambler::fracChanged () const [inline, virtual]

Implements **StringScrambler** (p. 519).

Definition at line 45 of file StringScramblers.h.

References theFracChanged.

#### **9.150.3.2 Word UniformScrambler::scramble (const Word &) const [virtual]**

Implements **StringScrambler** (p. 519).

### **9.150.4 Member Data Documentation**

#### **9.150.4.1 int UniformScrambler::numGens [private]**

Definition at line 50 of file StringScramblers.h.

#### **9.150.4.2 double UniformScrambler::theChangeProb [private]**

Definition at line 48 of file StringScramblers.h.

#### **9.150.4.3 double UniformScrambler::theFracChanged [private]**

Definition at line 49 of file StringScramblers.h.

Referenced by fracChanged().

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

- StringSimilarity/include/**StringScramblers.h**

## 9.151 Value Class Reference

Implements data types of parameters obtained from a configuration file.

```
#include <ConfigFile.h>
```

### Public Member Functions

- **Value** ()  
*Default constructor.*
- **Value** (const char \*v)  
*Constructor.*
- **Value** (int n)  
*Constructor.*
- **operator int** () const  
*Converts value into an integer.*
- **operator double** () const  
*Converts value into a double precision number.*
- **operator string** () const  
*Converts value into a string.*

### Private Attributes

- string **theValue**

### Friends

- istream & **operator**>> (istream &in, **Value** &v)  
*Input operator.*

#### 9.151.1 Detailed Description

Implements data types of parameters obtained from a configuration file. Allows universal handling of parameters with different data types. Usually an instance of this class is returned by **ConfigFile::getValue()** (p. 169) function.

Definition at line 38 of file ConfigFile.h.

## 9.151.2 Constructor & Destructor Documentation

### 9.151.2.1 Value::Value () [inline]

Default constructor.

Definition at line 42 of file ConfigFile.h.

### 9.151.2.2 Value::Value (const char \* *v*) [inline]

Constructor.

#### Parameters:

*v* - a string value.

Definition at line 47 of file ConfigFile.h.

### 9.151.2.3 Value::Value (int *n*) [inline]

Constructor.

#### Parameters:

*n* - an integer value.

Definition at line 52 of file ConfigFile.h.

References theValue.

## 9.151.3 Member Function Documentation

### 9.151.3.1 Value::operator double () const [inline]

Converts value into a double precision number.

#### Returns:

a double value. 0 if an error.

Definition at line 67 of file ConfigFile.h.

References theValue.



### 9.151.3.2 Value::operator int () const [inline]

Converts value into an integer.

**Returns:**

an integer value. 0 if an error.

Definition at line 62 of file ConfigFile.h.

References theValue.

### 9.151.3.3 Value::operator string () const [inline]

Converts value into a string.

**Returns:**

a string value.

Definition at line 72 of file ConfigFile.h.

References theValue.

## 9.151.4 Friends And Related Function Documentation

### 9.151.4.1 istream& operator>> (istream & in, Value & v) [friend]

Input operator.

Definition at line 74 of file ConfigFile.h.

## 9.151.5 Member Data Documentation

### 9.151.5.1 string Value::theValue [private]

Definition at line 80 of file ConfigFile.h.

Referenced by operator double(), operator int(), operator string(), and Value().

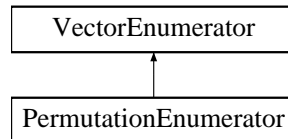
The documentation for this class was generated from the following file:

- general/include/ConfigFile.h

## 9.152 VectorEnumerator Class Reference

Class **VectorEnumerator** (p. 620).

#include <VectorEnumerator.h> Inheritance diagram for VectorEnumerator::



### Public Member Functions

- **VectorEnumerator ()**  
*Default constructor.*
- **VectorEnumerator & operator++ ()**  
*Compute the next element.*
- **vector< int > operator\* ()**  
*Get the current element.*
- **bool end ()**  
*Returns true if there are no more elements to enumerate.*
- **ostream & printSeq (ostream &os) const**

### Protected Member Functions

- **virtual bool seqOK () const =0**  
*Check if the currently constructed sequence is legitimate.*
- **virtual bool seqLimit () const =0**  
*Returns true if the length-limit on the sequence is passed.*
- **virtual bool seqComplete () const =0**  
*Returns true if the current sequence is complete.*
- **virtual int start () const =0**  
*Returns the initial value for the current element of the sequence (usually 0).*

- virtual int **next** (int cur) const =0

*Returns the next value for the current element of the sequence (usually cur+1).*

- virtual bool **finish** (int cur) const =0

- int **getLength** () const

*Get the length of the current sequence.*

- const vector< int > & **getSeq** () const

*Get the current sequence.*

- virtual void **stepTo** ()

*Function is invoked when the current value of the current element of the sequence is accepted and we go for the next element.*

- virtual void **stepBack** (int cur)

*Function is invoked when we go backward from the current element to the previous element.*

## Private Attributes

- int **curLength**
- vector< int > **curVector**

### 9.152.1 Detailed Description

Class **VectorEnumerator** (p. 620). **VectorEnumerator** (p. 620) provides the ground-level interface for vector-enumerators (for instance it can be used for enumeration of permutations, finite sequences over finite alphabet, etc.).

Definition at line 31 of file VectorEnumerator.h.

### 9.152.2 Constructor & Destructor Documentation

#### 9.152.2.1 VectorEnumerator::VectorEnumerator () [inline]

Default constructor.

Definition at line 43 of file VectorEnumerator.h.

### 9.152.3 Member Function Documentation

#### 9.152.3.1 `bool VectorEnumerator::end () [inline]`

Returns true if there are no more elements to enumerate.

Definition at line 67 of file VectorEnumerator.h.

References `curLength`, and `seqComplete()`.

#### 9.152.3.2 `virtual bool VectorEnumerator::finish (int cur) const [protected, pure virtual]`

Implemented in `PermutationEnumerator` (p. 383).

#### 9.152.3.3 `int VectorEnumerator::getLength () const [inline, protected]`

Get the length of the current sequence.

Definition at line 111 of file VectorEnumerator.h.

References `curLength`.

Referenced by `PermutationEnumerator::seqComplete()`, `PermutationEnumerator::seqLimit()`, `PermutationEnumerator::seqOK()`, `PermutationEnumerator::stepBack()`, and `PermutationEnumerator::stepTo()`.

#### 9.152.3.4 `const vector< int >& VectorEnumerator::getSeq () const [inline, protected]`

Get the current sequence.

Definition at line 113 of file VectorEnumerator.h.

References `curVector`.

Referenced by `PermutationEnumerator::getPermutation()`, `PermutationEnumerator::seqOK()`, `PermutationEnumerator::stepBack()`, and `PermutationEnumerator::stepTo()`.

#### 9.152.3.5 `virtual int VectorEnumerator::next (int cur) const [protected, pure virtual]`

Returns the next value for the current element of the sequence (usually `cur+1`).

Implemented in `PermutationEnumerator` (p. 384).

**9.152.3.6** `vector< int > VectorEnumerator::operator* ()`

Get the current element.

**9.152.3.7** `VectorEnumerator& VectorEnumerator::operator++ ()`

Compute the next element.

**9.152.3.8** `ostream& VectorEnumerator::printSeq (ostream & os) const`**9.152.3.9** `virtual bool VectorEnumerator::seqComplete () const`  
`[protected, pure virtual]`

Returns true if the current sequence is complete.

Implemented in **PermutationEnumerator** (p. 384).

Referenced by end().

**9.152.3.10** `virtual bool VectorEnumerator::seqLimit () const` `[protected, pure virtual]`

Returns true if the length-limit on the sequence is passed.

Implemented in **PermutationEnumerator** (p. 384).

**9.152.3.11** `virtual bool VectorEnumerator::seqOK () const` `[protected, pure virtual]`

Check if the currently constructed sequence is legitimate.

Implemented in **PermutationEnumerator** (p. 384).

**9.152.3.12** `virtual int VectorEnumerator::start () const` `[protected, pure virtual]`

Returns the initial value for the current element of the sequence (usually 0).

Implemented in **PermutationEnumerator** (p. 385).

**9.152.3.13 virtual void VectorEnumerator::stepBack (int *cur*) [inline, protected, virtual]**

Function is invoked when we go backward from the current element to the previous element. The value of `curElement` is already decreased, so you are at the previous element

Reimplemented in **PermutationEnumerator** (p. 385).

Definition at line 124 of file `VectorEnumerator.h`.

**9.152.3.14 virtual void VectorEnumerator::stepTo () [inline, protected, virtual]**

Function is invoked when the current value of the current element of the sequence is accepted and we go for the next element. The value of `curElement` is already increased, so you are at the next element.

Reimplemented in **PermutationEnumerator** (p. 385).

Definition at line 119 of file `VectorEnumerator.h`.

**9.152.4 Member Data Documentation****9.152.4.1 int VectorEnumerator::curLength [private]**

Definition at line 134 of file `VectorEnumerator.h`.

Referenced by `end()`, and `getLength()`.

**9.152.4.2 vector< int > VectorEnumerator::curVector [private]**

Definition at line 135 of file `VectorEnumerator.h`.

Referenced by `getSeq()`.

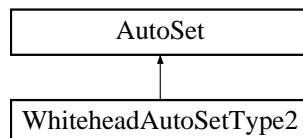
The documentation for this class was generated from the following file:

- `general/include/VectorEnumerator.h`

## 9.153 WhiteheadAutoSetType2 Class Reference

Implements the set of Whitehead automorphisms of type II.

`#include <WhiteheadAutoSet.h>` Inheritance diagram for WhiteheadAutoSetType2::



### Public Member Functions

- **WhiteheadAutoSetType2** (int n)  
*Constructor. Generates the set of Whitehead automorphisms of type II.*
- **~WhiteheadAutoSetType2** ()
- const **SetOfMaps** & **getSet** () const  
*Returns a random Whitehead automorphism.*
- const **Map** & **getRandomAuto** () const  
*Returns the set of Whitehead automorphisms.*

### Private Member Functions

- **Map** **getMap** (int n, const vector< int > &tCounts, **Word** a)
- void **computeSet** (int n)

### Private Attributes

- **SetOfMaps** theSet
- int nGens

### Static Private Attributes

- static const int nElemAutos = 4

### 9.153.1 Detailed Description

Implements the set of Whitehead automorphisms of type II. Whitehead automorphisms of the type II are the Whitehead automorphisms which may alter the length of an input word. Basically it is all the Whitehead automorphisms excluding permutations of the generators.

Definition at line 169 of file WhiteheadAutoSet.h.

### 9.153.2 Constructor & Destructor Documentation

#### 9.153.2.1 WhiteheadAutoSetType2::WhiteheadAutoSetType2 (int *n*)

Constructor. Generates the set of Whitehead automorphisms of type II. The set of Whitehead automorphisms for a free group of rank *n* is generated.

Note, the whole set is generated by enumeration. ( $2^n$ ) automorphisms. This is a brute force approach and inapplicable for groups with large ranks.

#### Parameters:

*n* - rank of a free group.

#### 9.153.2.2 WhiteheadAutoSetType2::~~WhiteheadAutoSetType2 ()

### 9.153.3 Member Function Documentation

#### 9.153.3.1 void WhiteheadAutoSetType2::computeSet (int *n*) [private]

#### 9.153.3.2 Map WhiteheadAutoSetType2::getMap (int *n*, const vector< int > & *tCounts*, Word *a*) [private]

#### 9.153.3.3 const Map& WhiteheadAutoSetType2::getRandomAuto () const

Returns the set of Whitehead automorphisms.

#### Returns:

the set of Whitehead automorphisms.

#### 9.153.3.4 const SetOfMaps& WhiteheadAutoSetType2::getSet () const [inline, virtual]

Returns a random Whitehead automorphism.



**Returns:**

Whitehead automorphisms selected uniformly randomly form the set.

Implements **AutoSet** (p. 110).

Definition at line 204 of file WhiteheadAutoSet.h.

References theSet.

**9.153.4 Member Data Documentation****9.153.4.1** `const int WhiteheadAutoSetType2::nElemAutos = 4` `[static, private]`

Definition at line 222 of file WhiteheadAutoSet.h.

**9.153.4.2** `int WhiteheadAutoSetType2::nGens` `[private]`

Definition at line 225 of file WhiteheadAutoSet.h.

**9.153.4.3** `SetOfMaps WhiteheadAutoSetType2::theSet` `[private]`

Definition at line 224 of file WhiteheadAutoSet.h.

Referenced by getSet().

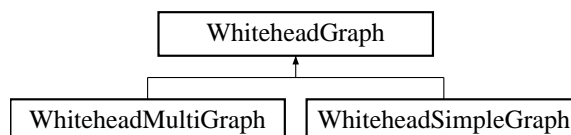
The documentation for this class was generated from the following file:

- Maps/include/WhiteheadAutoSet.h

## 9.154 WhiteheadGraph Class Reference

Interface class for Whitehead graphs.

`#include <WhiteheadGraph.h>` Inheritance diagram for WhiteheadGraph::



### Public Member Functions

- **WhiteheadGraph** (const **Word** &w, int num\_of\_gens)  
*Constructor.*
- const **Word** & **getWord** () const  
*Get the input word.*
- **WhiteheadGraph** (int n\_gens, const **Word** &w)
- template<class ConstIterator >  
void **assign** (int n\_gens, ConstIterator B, ConstIterator E)
- const **Graph** & **getGraph** ()

### Protected Attributes

- **Word** theWord  
*The input word.*
- int nOfGenerators  
*The number of generators in the group.*

### Private Attributes

- **Graph** theGraph

#### 9.154.1 Detailed Description

Interface class for Whitehead graphs.

Definition at line 121 of file WhiteheadGraph.h.

## 9.154.2 Constructor & Destructor Documentation

### 9.154.2.1 WhiteheadGraph::WhiteheadGraph (const Word & *w*, int *num\_of\_gens*) [inline]

Constructor.

**Parameters:**

*w* - the input word.

*num\_of\_gens* - the number of generators in the corresponding group (i.e.  $w \in F_{num\_of\_gens}$ ).

Definition at line 130 of file WhiteheadGraph.h.

### 9.154.2.2 WhiteheadGraph::WhiteheadGraph (int *n\_gens*, const Word & *w*) [inline]

Definition at line 35 of file WhiteheadGraph.h.

References `assign()`, `Word::begin()`, and `Word::end()`.

## 9.154.3 Member Function Documentation

### 9.154.3.1 template<class ConstIterator > void WhiteheadGraph::assign (int *n\_gens*, ConstIterator *B*, ConstIterator *E*) [inline]

Definition at line 44 of file WhiteheadGraph.h.

References `Graph::clear()`, `Graph::newEdge()`, and `theGraph`.

Referenced by `WhiteheadGraph()`.

### 9.154.3.2 const Graph& WhiteheadGraph::getGraph () [inline]

Definition at line 65 of file WhiteheadGraph.h.

References `theGraph`.

### 9.154.3.3 const Word& WhiteheadGraph::getWord () const [inline]

Get the input word.

Definition at line 138 of file WhiteheadGraph.h.

References `theWord`.

## 9.154.4 Member Data Documentation

### 9.154.4.1 `int WhiteheadGraph::nOfGenerators` `[protected]`

The number of generators in the group.

Definition at line 143 of file `WhiteheadGraph.h`.

Referenced by `WhiteheadSimpleGraph::genToIndex()`, and `WhiteheadSimpleGraph::indToGenerator()`.

### 9.154.4.2 `Graph WhiteheadGraph::theGraph` `[private]`

Definition at line 76 of file `WhiteheadGraph.h`.

Referenced by `assign()`, and `getGraph()`.

### 9.154.4.3 `Word WhiteheadGraph::theWord` `[protected]`

The input word.

Definition at line 141 of file `WhiteheadGraph.h`.

Referenced by `getWord()`.

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

- `FreeGroup/include/WhiteheadGraph.h`
- `SbgpFG/include/WhiteheadGraph.h`

## 9.155 WhiteheadMinimization Class Reference

Implements a greedy procedure of reducing a word to its minimal length.

```
#include <WhiteheadAutoSet.h>
```

### Public Member Functions

- **WhiteheadMinimization** (int *n*)  
*Constructor.*
- bool **isMinimal** (const **Word** &*w*) const  
*Test if the word is minimal.*
- **Word** **findMinimal** (const **Word** &*w*, ostream \**out*=NULL) const  
*Find a word of the minimal length.*
- const **WhiteheadAutoSetType2** & **getSet** () const  
*Get the Whitehead set of type II.*

### Private Attributes

- **WhiteheadAutoSetType2** *wSet*

### 9.155.1 Detailed Description

Implements a greedy procedure of reducing a word to its minimal length. Greedy procedure applies automorphisms from a Whitehead set of type II to a word until its length is reduced. If a shorter image is found, procedure is applied to the shorten word. If none of the automorphisms can reduce the length, procedure stops and the shortest word is returned.

Definition at line 238 of file WhiteheadAutoSet.h.

### 9.155.2 Constructor & Destructor Documentation

#### 9.155.2.1 WhiteheadMinimization::WhiteheadMinimization (int *n*) [inline]

Constructor. A set of Whitehead automorphisms is created in the constructor. May not be applicable when the rank is large.

**Parameters:**

$n$  - the rank of a free group.

Definition at line 246 of file WhiteheadAutoSet.h.

### 9.155.3 Member Function Documentation

#### 9.155.3.1 Word WhiteheadMinimization::findMinimal (const Word & $w$ , ostream \* $out$ = NULL) const

Find a word of the minimal length. Applies automorphisms from the Whitehead set to find a word of the minimal length in the automorphic orbit of a given word.

**Parameters:**

$w$  - initial word.

**Returns:**

a word of the minimal length in the orbit of  $w$ .

#### 9.155.3.2 const WhiteheadAutoSetType2& WhiteheadMinimization::getSet () const [inline]

Get the Whitehead set of type II.

**Returns:**

the Whitehead set of type II.

Definition at line 271 of file WhiteheadAutoSet.h.

References wSet.

#### 9.155.3.3 bool WhiteheadMinimization::isMinimal (const Word & $w$ ) const

Test if the word is minimal. Applies automorphisms from the Whitehead set to test if a given word has minimal length.

**Parameters:**

$w$  - word to be tested.

**Returns:**

true if  $w$  has the minimal length, false otherwise.

## 9.155.4 Member Data Documentation

### 9.155.4.1 WhiteheadAutoSetType2 WhiteheadMinimization::wSet [private]

Definition at line 274 of file WhiteheadAutoSet.h.

Referenced by `getSet()`.

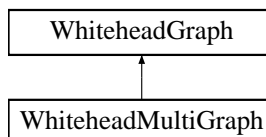
The documentation for this class was generated from the following file:

- `Maps/include/WhiteheadAutoSet.h`

## 9.156 WhiteheadMultiGraph Class Reference

Whitehead Multi-Graph (Not implemented yet).

`#include <WhiteheadGraph.h>`Inheritance diagram for WhiteheadMultiGraph::



### Public Member Functions

- **WhiteheadMultiGraph** (const **Word** &*w*, int *n*)

#### 9.156.1 Detailed Description

Whitehead Multi-Graph (Not implemented yet).

Definition at line 286 of file WhiteheadGraph.h.

#### 9.156.2 Constructor & Destructor Documentation

##### 9.156.2.1 WhiteheadMultiGraph::WhiteheadMultiGraph (const Word & *w*, int *n*) [**inline**]

Definition at line 289 of file WhiteheadGraph.h.

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

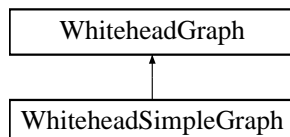
- FreeGroup/include/**WhiteheadGraph.h**



## 9.157 WhiteheadSimpleGraph Class Reference

Implements Whitehead Simple graph (i.e. no multiple edges or loops allowed).

`#include <WhiteheadGraph.h>` Inheritance diagram for WhiteheadSimpleGraph::



### Public Member Functions

- **WhiteheadSimpleGraph** (const **Word** &w, int num\_of\_gens)  
*Constructor.*

- **~WhiteheadSimpleGraph** ()

- int **getCount** (int i, int j) const  
*Get the label of the edge (i,j).*

- int **getSize** () const  
*Get the number of vertices of the graph.*

- vector< double > **getWeightVector** () const  
*Get a vector containing weights of the edges of the graph.*

- vector< **Word** > **getWeightNames** () const  
*Get a vector of words, corresponding to edges of the graph.*

- bool **isUndirected** () const  
*Check if undirected graph.*

- void **makeUndirected** ()  
*Convert to undirected Whitehead graph.*

- int **numberOfComponents** () const  
*Get the number of connected components.*

- int **numberOfCutVertices** () const  
*Get the number of cut vertices (articulation points).*

- `vector< int > cutVertices () const`  
*Get the list of cut vertices (articulation points).*
- `int numberOfCutVerticesBruteForce () const`
- `vector< int > cutVerticesBruteForce () const`

## Private Member Functions

- `void printOn (ostream &out) const`  
*Outputs the graph into a stream.*
- `int genToIndex (const Generator &g) const`  
*Transfers a letter into a graph vertex.*
- `Generator indToGenerator (int i) const`  
*Transfers a vertex into the corresponding letter.*
- `int nOfComponents () const`  
*computes the number of connected components*

## Private Attributes

- `int theSize`  
*The number of vertices.*
- `int ** theAdjMatrix`  
*The adjacensy matrix of the graph.*
- `bool undirected`  
*True if undirected.*

## Friends

- `ostream & operator<< (ostream &out, const WhiteheadSimpleGraph &g)`  
*Output operator.*

### 9.157.1 Detailed Description

Implements Whitehead Simple graph (i.e. no multiple edges or loops allowed).

Definition at line 155 of file WhiteheadGraph.h.

### 9.157.2 Constructor & Destructor Documentation

#### 9.157.2.1 WhiteheadSimpleGraph::WhiteheadSimpleGraph (const Word & *w*, int *num\_of\_gens*)

Constructor.

**Parameters:**

*w* - input word.

*num\_of\_gen* - the number of generators in the corresponding group

#### 9.157.2.2 WhiteheadSimpleGraph::~~WhiteheadSimpleGraph ()

### 9.157.3 Member Function Documentation

#### 9.157.3.1 vector<int> WhiteheadSimpleGraph::cutVertices () const

Get the list of cut vertices (articulation points).

**Returns:**

The list of cut vertices.

#### 9.157.3.2 vector<int> WhiteheadSimpleGraph::cutVerticesBruteForce () const

#### 9.157.3.3 int WhiteheadSimpleGraph::genToIndex (const Generator & *g*) const [inline, private]

Transfers a letter into a graph vertex.

Definition at line 246 of file WhiteheadGraph.h.

References WhiteheadGraph::nOfGenerators.

#### 9.157.3.4 int WhiteheadSimpleGraph::getCount (int *i*, int *j*) const [inline]

Get the label of the edge (i,j). The label is the number of times (counts) the subword  $ij^{-1}$  occurred in *w*.

**Parameters:**

*i* - first vertex (letter)  
*j* - second vertex (letter)

**Returns:**

Label of the edge (i,j)

Definition at line 180 of file WhiteheadGraph.h.

References `msgs::error()`, `theAdjMatrix`, and `theSize`.

**9.157.3.5 int WhiteheadSimpleGraph::getSize () const [inline]**

Get the number of vertices of the graph.

**Returns:**

The number of vertices

Definition at line 190 of file WhiteheadGraph.h.

References `theSize`.

**9.157.3.6 vector<Word> WhiteheadSimpleGraph::getWeightNames () const**

Get a vector of words, corresponding to edges of the graph.

**Returns:**

The vector of words, corresponding to the edges of the Whitehead graph.

**9.157.3.7 vector<double> WhiteheadSimpleGraph::getWeightVector () const**

Get a vector containing weights of the edges of the graph.

**Returns:**

Vector of the weights of the edges of the graph. NOTE: the graph is treated as undirected Whitehead graph when weights are computed. See `makeUndirected()` (p. 639) for more information.

### 9.157.3.8 Generator WhiteheadSimpleGraph::indToGenerator (int *i*) const [inline, private]

Transfers a vertex into the corresponding letter.

Definition at line 255 of file WhiteheadGraph.h.

References WhiteheadGraph::nOfGenerators.

### 9.157.3.9 bool WhiteheadSimpleGraph::isUndirected () const [inline]

Check if undirected graph. By default the graph is undirected Whitehead graph. Run **makeUndirected()** (p. 639) to convert to undirected graph.

#### Returns:

`true` if the graph is undirected Whitehead graph, `false` otherwise.

Definition at line 210 of file WhiteheadGraph.h.

References undirected.

### 9.157.3.10 void WhiteheadSimpleGraph::makeUndirected ()

Convert to undirected Whitehead graph. Initially a Directed Whitehead graph is constructed. Use this function to convert to undirected Whitehead graph.

Edges are added, and the new edge weights  $\omega^*(i, j) = \omega^*(j, i) = \omega(i, j) + \omega(j, i)$ , where  $\omega(m, n)$  are the weights in the directed graph.

### 9.157.3.11 int WhiteheadSimpleGraph::nOfComponents () const [private]

computes the number of connected components

### 9.157.3.12 int WhiteheadSimpleGraph::numberOfComponents () const

Get the number of connected components.

#### Returns:

The number of connected components.

### 9.157.3.13 int WhiteheadSimpleGraph::numberOfCutVertices () const

Get the number of cut vertices (articulation points).

**Returns:**

The number of cut vertices.

**9.157.3.14** `int WhiteheadSimpleGraph::numberOfCutVerticesBruteForce () const`

**9.157.3.15** `void WhiteheadSimpleGraph::printOn (ostream & out) const [private]`

Outputs the graph into a stream.

## **9.157.4 Friends And Related Function Documentation**

**9.157.4.1** `ostream& operator<< (ostream & out, const WhiteheadSimpleGraph & g) [friend]`

Output operator.

Definition at line 167 of file WhiteheadGraph.h.

## **9.157.5 Member Data Documentation**

**9.157.5.1** `int** WhiteheadSimpleGraph::theAdjMatrix [private]`

The adjacency matrix of the graph.

Definition at line 272 of file WhiteheadGraph.h.

Referenced by getCount().

**9.157.5.2** `int WhiteheadSimpleGraph::theSize [private]`

The number of vertices.

Definition at line 269 of file WhiteheadGraph.h.

Referenced by getCount(), and getSize().

**9.157.5.3** `bool WhiteheadSimpleGraph::undirected [private]`

True if undirected.

Definition at line 275 of file WhiteheadGraph.h.

Referenced by `isUndirected()`.

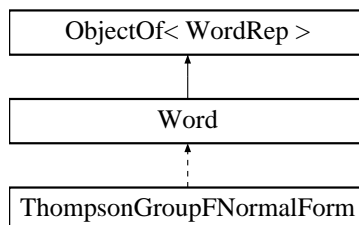
The documentation for this class was generated from the following file:

- `FreeGroup/include/WhiteheadGraph.h`

## 9.158 Word Class Reference

Class **Word** (p. 642) (defines a representation of a **Word** (p. 642) over a group alphabet)//.

#include <Word.h> Inheritance diagram for Word::



### Public Types

- typedef **ConstWordIterator** **const\_iterator**
- typedef **WordIterator** **iterator**
- typedef pair< int, int > **PII**

### Public Member Functions

- **Word** ()  
*Default constructor (creates the empty word  $w = \varepsilon$ ).*
- **Word** (const vector< int > &gens)  
*Cast constructor. Constructs a word by its presentation (if the word defined in gens is not reduced then it reduces it).*
- **Word** (const list< int > &gens)  
*Cast constructor. Constructs a word by its presentation (if the word defined in gens is not reduced then it reduces it).*
- **Word** (int g)  
*Cast constructor. Constructs a one letter word.*
- template<class IntIterator >  
  **Word** (const IntIterator &B, const IntIterator &E)
- bool **operator**< (const **Word** &wr) const  
*Comparison operator.*



- **bool operator>** (const **Word** &wr) const  
*Comparison operator.*
- **bool operator==** (const **Word** &wr) const  
*Comparison operator.*
- **bool operator!=** (const **Word** &wr) const  
*Comparison operator.*
- **Word & operator\*=** (const **Word** &w)  
*Multiply the word on the right by another word (the result is reduced).*
- **Word operator\*** (const **Word** &w) const  
*Multiply two words (the result is reduced).*
- **Word & operator^=** (const **Word** &conjugator)  
*Conjugate a word by another word (the result is reduced).*
- **Word operator^** (const **Word** &conjugator) const
- **Word & operator^=** (int power)  
*Conjugate a word by another word (the result is reduced).*
- **Word operator^** (int power) const
- **Word operator-** () const  
*Invert a word (works the same as inverse).*
- **const\_iterator begin** () const
- **const\_iterator end** () const
- **iterator begin** ()
- **iterator end** ()
- **Word freelyReduce** ()  
*Freely reduce a word.*
- **Word freelyReduce (iterator B, iterator E)**  
*Freely reduce a segment of a word defined by [B,E).*
- **const list< int > & getList** () const  
*Get a constant representation the word.*
- **list< int > & getList** ()  
*Get a representation the word. This allows direct manipulation with the representation (which requires caution).*

- **int length ()** const  
*Get the length of the word.*
- **Word & push\_back (int gen)**  
*Multiply the word by a one-letter word defined by gen on the right. The result is being reduced.*
- **Word & push\_front (int gen)**  
*Multiply the word by a one-letter word defined by gen on the left. The result is being reduced.*
- **Word & push\_back (const Word &w)**  
*Multiply the word by a word on the right. The result is being reduced.*
- **Word & push\_front (const Word &w)**  
*Multiply the word by a word on the left. The result is being reduced.*
- **void pop\_back ()**  
*Remove the last symbol.*
- **void pop\_front ()**  
*Remove the first symbol.*
- **int getPower (Word &base)** const  
*Determines the power of the word (as an element of a free monoid, not as an element of a free group).*
- **bool doesContain (const int &gen)** const  
*Checks if the word contains a generator given by gen.*
- **void cyclicLeftShift ()**  
*Shifts the word one position to the left, i.e.,  $\text{cyclicLeftShift}(x_1x_2 \dots x_{k-1}x_k) = x_2 \dots x_{k-1}x_kx_1$ .*
- **void cyclicRightShift ()**  
*Shifts the word one position to the right, i.e.,  $\text{cyclicRightShift}(x_1x_2 \dots x_{k-1}x_k) = x_kx_1x_2 \dots x_{k-1}$ .*
- **Word cyclicallyReduce ()** const  
*Returns the cyclically reduced word.*
- **void cyclicallyReduceWord ()**  
*Cyclically reduces the **Word** (p. 642).*

- **Word cyclicallyReduce** (**Word** &conjugator) const  
*Returns the cyclically reduced word and the corresponding conjugator.*
- void **cyclicallyReduceWord** (**Word** &conjugator)  
*Cyclically reduces the **Word** (p. 642) and returns the corresponding conjugator.*
- **Word inverse** () const  
*Invert the word.*
- **Word cyclicallyPermute** (int n) const  
*Cyclically permute the word and return the result.*
- **Word & \_cyclicallyPermute** (int n)  
*Cyclically permute the word.*
- **Word initialSegment** (int len) const  
*Get an initial segment of the word of length len.*
- **Word terminalSegment** (int len) const  
*Get a terminal segment of the word of length len.*
- **Word segment** (int from, int to) const
- int **exponentSum** (const int &gen) const
- int **isIn** (const int &gen) const
- **Word power** (int t) const
- template<class ConstIntIterator >  
void **insert** (int pos, ConstIntIterator B, ConstIntIterator E)  
*Insert a sequence of generators [B,E) into a word at a position pos.*
- void **insert** (int pos, int g)  
*Insert a generator g into a word at a position pos.*
- template<class ConstIntIterator >  
void **insert** (**WordIterator** it, ConstIntIterator B, ConstIntIterator E)  
*Insert a sequence of generators [B,E) into a word before a position it.*
- void **insert** (**WordIterator** it, int g)  
*Insert a generator g into a word before a position it.*
- void **replace** (**WordIterator** it, const **Generator** &g)  
*Replace a generator at a position it by g.*

- void **replace** (int pos, const **Generator** &g)  
*Replace a generator at a position pos by g.*
- template<class ConstIntIterator >  
void **replace** (**WordIterator** it, ConstIntIterator B, ConstIntIterator E)  
*Replace a subword of a word starting at a position it by a word [B,E). The length of the word does not increase if [B,E) is longer than the terminal segment of the word [it,end() (p. 650)). In that case terminal symbols of [B,E) are ignored.*
- **Word replaceGenerators** (const vector< **Word** > &images) const  
*Returns a word in which generators are replaced by words.*
- **Word minimalEquivalentForm** (const set< int > &permutableGenerators, bool inverses, bool cyclicPermutations) const  
*Compute the minimal equivalent word.*

## Static Public Member Functions

- static **Word randomWord** (int n, int wLen)  
*Generates a pseudo randomly reduced word of the length wLen over the alphabet  $X = \{x_1, \dots, x_n\}$ .*
- static **Word randomWord** (int n, int wLenMin, int wLenMax)  
*Generates a pseudo randomly reduced word of a length in [wLenMin,wLenMax] and over the alphabet  $X = \{x_1, \dots, x_n\}$ .*

## Private Member Functions

- ostream & **printOn** (ostream &os) const

## Friends

- ostream & **operator**<< (ostream &os, const **Word** &w)
- istream & **operator**>> (istream &is, **Word** &w)

### 9.158.1 Detailed Description

Class **Word** (p. 642) (defines a representation of a **Word** (p. 642) over a group alphabet)//. A reduced word is the one which does not involve subwords of the type  $xx^{-1}$  or  $x^{-1}x$ . We represent a reduced word over a group alphabet  $X$  as a list of non-trivial

integers `list< int >`. Each generator  $x \in X$  is represented by the unique number  $n_x$ . For each  $x \in X$  it is assumed that  $n_{x^{-1}} = -n_x$ .

Definition at line 38 of file Word.h.

## 9.158.2 Member Typedef Documentation

### 9.158.2.1 `typedef ConstWordIterator Word::const_iterator`

Definition at line 42 of file Word.h.

### 9.158.2.2 `typedef WordIterator Word::iterator`

Definition at line 43 of file Word.h.

### 9.158.2.3 `typedef pair< int , int > Word::PII`

Definition at line 53 of file Word.h.

## 9.158.3 Constructor & Destructor Documentation

### 9.158.3.1 `Word::Word () [inline]`

Default constructor (creates the empty word  $w = \varepsilon$ ).

Definition at line 56 of file Word.h.

### 9.158.3.2 `Word::Word (const vector< int > & gens) [inline]`

Cast constructor. Constructs a word by its presentation (if the word defined in gens is not reduced then it reduces it).

Definition at line 58 of file Word.h.

### 9.158.3.3 `Word::Word (const list< int > & gens) [inline]`

Cast constructor. Constructs a word by its presentation (if the word defined in gens is not reduced then it reduces it).

Definition at line 60 of file Word.h.

**9.158.3.4 Word::Word (int *g*) [inline]**

Cast constructor. Constructs a one letter word.

Definition at line 62 of file Word.h.

**9.158.3.5 template<class IntIterator > Word::Word (const IntIterator & *B*, const IntIterator & *E*) [inline]**

Definition at line 65 of file Word.h.

**9.158.4 Member Function Documentation****9.158.4.1 Word& Word::\_cyclicallyPermute (int *n*) [inline]**

Cyclically permute the word.

Definition at line 233 of file Word.h.

References ObjectOf< WordRep >::change(), and cyclicallyPermute().

**9.158.4.2 iterator Word::begin () [inline]**

Definition at line 145 of file Word.h.

**9.158.4.3 const\_iterator Word::begin () const [inline]**

Definition at line 143 of file Word.h.

Referenced by BraidDrawPDF::drawCompressedBraid(), WordDraw::drawCompressedBraid(), MaxCommutePartition::findCommutParts(), freelyReduce(), DCBraidReduction::shorten(), and WhiteheadGraph::WhiteheadGraph().

**9.158.4.4 Word Word::cyclicallyPermute (int *n*) const [inline]**

Cyclically permute the word and return the result.  $n > 0 \Rightarrow$  left-shift,  $n < 0 \Rightarrow$  right-shift permute.

Definition at line 227 of file Word.h.

References ObjectOf< Rep >::change().

Referenced by \_cyclicallyPermute().

#### 9.158.4.5 Word Word::cyclicallyReduce (Word & *conjugator*) const [inline]

Returns the cyclically reduced word and the corresponding conjugator.

Definition at line 208 of file Word.h.

References ObjectOf< Rep >::change(), and cyclicallyReduce().

#### 9.158.4.6 Word Word::cyclicallyReduce () const [inline]

Returns the cyclically reduced word.

Definition at line 198 of file Word.h.

References ObjectOf< Rep >::change().

Referenced by cyclicallyReduce(), and cyclicallyReduceWord().

#### 9.158.4.7 void Word::cyclicallyReduceWord (Word & *conjugator*) [inline]

Cyclically reduces the **Word** (p. 642) and returns the corresponding conjugator.

Definition at line 214 of file Word.h.

References ObjectOf< Rep >::change(), ObjectOf< WordRep >::change(), and cyclicallyReduce().

#### 9.158.4.8 void Word::cyclicallyReduceWord () [inline]

Cyclically reduces the **Word** (p. 642).

Definition at line 205 of file Word.h.

References ObjectOf< WordRep >::change(), and cyclicallyReduce().

#### 9.158.4.9 void Word::cyclicLeftShift () [inline]

Shifts the word one position to the left, i.e.,  $\text{cyclicLeftShift}(x_1x_2 \dots x_{k-1}x_k) = x_2 \dots x_{k-1}x_kx_1$ .

Definition at line 193 of file Word.h.

References ObjectOf< WordRep >::change(), and WordRep::cyclicLeftShift().

**9.158.4.10 void Word::cyclicRightShift () [inline]**

Shifts the word one position to the right, i.e.,  $cyclicRightShift(x_1x_2 \dots x_{k-1}x_k) = x_kx_1x_2 \dots x_{k-1}$ .

Definition at line 195 of file Word.h.

References ObjectOf< WordRep >::change(), and WordRep::cyclicRightShift().

**9.158.4.11 bool Word::doesContain (const int & gen) const [inline]**

Checks if the word contains a generator given by gen.

Definition at line 190 of file Word.h.

References WordRep::doesContain(), and ObjectOf< WordRep >::look().

**9.158.4.12 iterator Word::end () [inline]**

Definition at line 146 of file Word.h.

**9.158.4.13 const\_iterator Word::end () const [inline]**

Definition at line 144 of file Word.h.

Referenced by BraidDrawPDF::drawCompressedBraid(), WordDraw::drawCompressedBraid(), MaxCommutePartition::findCommutParts(), freelyReduce(), DCBraidReduction::shorten(), and WhiteheadGraph::WhiteheadGraph().

**9.158.4.14 int Word::exponentSum (const int & gen) const [inline]**

Definition at line 259 of file Word.h.

References WordRep::exponentSum(), and ObjectOf< WordRep >::look().

**9.158.4.15 Word Word::freelyReduce (iterator B, iterator E)**

Freely reduce a segment of a word defined by [B,E).

**9.158.4.16 Word Word::freelyReduce () [inline]**

Freely reduce a word.

Definition at line 149 of file Word.h.



References `begin()`, `end()`, and `freelyReduce()`.

Referenced by `freelyReduce()`.

#### 9.158.4.17 `list< int >& Word::getList () [inline]`

Get a representation the word. This allows direct manipulation with the representation (which requires caution).

Definition at line 166 of file `Word.h`.

References `ObjectOf< WordRep >::change()`, and `WordRep::getList()`.

#### 9.158.4.18 `const list< int >& Word::getList () const [inline]`

Get a constant representation the word.

Definition at line 164 of file `Word.h`.

References `WordRep::getList()`, and `ObjectOf< WordRep >::look()`.

Referenced by `ThompsonGroupFNormalForm::operator*()`, and `ThompsonGroupFNormalForm::operator*=( )`.

#### 9.158.4.19 `int Word::getPower (Word & base) const [inline]`

Determines the power of the word (as an element of a free monoid, not as an element of a free group).

Definition at line 187 of file `Word.h`.

References `ObjectOf< Rep >::change()`, `WordRep::getPower()`, and `ObjectOf< WordRep >::look()`.

#### 9.158.4.20 `Word Word::initialSegment (int len) const [inline]`

Get an initial segment of the word of length `len`.

Definition at line 239 of file `Word.h`.

References `ObjectOf< Rep >::change()`.

#### 9.158.4.21 `void Word::insert (WordIterator it, int g)`

Insert a generator `g` into a word before a position `it`.

**9.158.4.22** `template<class ConstIntIterator > void Word::insert (WordIterator it, ConstIntIterator B, ConstIntIterator E) [inline]`

Insert a sequence of generators [B,E) into a word before a position *it*.

**9.158.4.23** `void Word::insert (int pos, int g)`

Insert a generator *g* into a word at a position *pos*.

**9.158.4.24** `template<class ConstIntIterator > void Word::insert (int pos, ConstIntIterator B, ConstIntIterator E) [inline]`

Insert a sequence of generators [B,E) into a word at a position *pos*.

**9.158.4.25** `Word Word::inverse () const [inline]`

Invert the word.

Definition at line 219 of file Word.h.

References `ObjectOf< Rep >::change()`, `WordRep::inverse()`, and `ObjectOf< WordRep >::look()`.

**9.158.4.26** `int Word::isIn (const int & gen) const [inline]`

Definition at line 263 of file Word.h.

References `WordRep::isIn()`, and `ObjectOf< WordRep >::look()`.

**9.158.4.27** `int Word::length () const [inline]`

Get the length of the word.

Definition at line 169 of file Word.h.

References `WordRep::length()`, and `ObjectOf< WordRep >::look()`.

Referenced by `UniformPartition::getPartition()`, and `WordDraw::WordDraw()`.

**9.158.4.28** `Word Word::minimalEquivalentForm (const set< int > & permutableGenerators, bool inverses, bool cyclicPermutations) const`

Compute the minimal equivalent word. *permutableGenerators* must be positive.

**9.158.4.29 bool Word::operator!= (const Word & wr) const [inline]**

Comparison operator.

Definition at line 88 of file Word.h.

References ObjectOf< Rep >::look(), and ObjectOf< WordRep >::look().

**9.158.4.30 Word Word::operator\* (const Word & w) const [inline]**

Multiply two words (the result is reduced).

Reimplemented in **ThompsonGroupFNormalForm** (p. 558).

Definition at line 96 of file Word.h.

**9.158.4.31 Word& Word::operator\*= (const Word & w) [inline]**

Multiply the word on the right by another word (the result is reduced).

Reimplemented in **ThompsonGroupFNormalForm** (p. 558).

Definition at line 91 of file Word.h.

References ObjectOf< WordRep >::change(), and ObjectOf< Rep >::look().

**9.158.4.32 Word Word::operator- () const [inline]**

Invert a word (works the same as inverse).

Reimplemented in **ThompsonGroupFNormalForm** (p. 558).

Definition at line 128 of file Word.h.

References ObjectOf< Rep >::change(), WordRep::inverse(), and ObjectOf< WordRep >::look().

**9.158.4.33 bool Word::operator< (const Word & wr) const [inline]**

Comparison operator. The result of comparison of  $u = u_1 \dots u_k$  and  $v = v_1 \dots v_m$  is defined by  $u < v \Leftrightarrow k < m \vee (k = m \wedge u_i < v_i \text{ where } i \text{ is the smallest index such that } u_i \neq v_i)$

Definition at line 82 of file Word.h.

References ObjectOf< Rep >::look(), and ObjectOf< WordRep >::look().

**9.158.4.34** `bool Word::operator==(const Word & wr) const [inline]`

Comparison operator.

Reimplemented in **ThompsonGroupFNormalForm** (p. 558).

Definition at line 86 of file Word.h.

References `ObjectOf< Rep >::look()`, and `ObjectOf< WordRep >::look()`.

**9.158.4.35** `bool Word::operator>(const Word & wr) const [inline]`

Comparison operator.

Definition at line 84 of file Word.h.

References `ObjectOf< Rep >::look()`, and `ObjectOf< WordRep >::look()`.

**9.158.4.36** `Word Word::operator^(int power) const [inline]`

Definition at line 120 of file Word.h.

**9.158.4.37** `Word Word::operator^(const Word & conjugator) const [inline]`

Definition at line 108 of file Word.h.

**9.158.4.38** `Word& Word::operator^=(int power) [inline]`

Conjugate a word by another word (the result is reduced).

Definition at line 116 of file Word.h.

References `ObjectOf< WordRep >::change()`.

**9.158.4.39** `Word& Word::operator^=(const Word & conjugator) [inline]`

Conjugate a word by another word (the result is reduced).

Definition at line 104 of file Word.h.

References `ObjectOf< WordRep >::change()`, and `ObjectOf< Rep >::look()`.

**9.158.4.40** `void Word::pop_back() [inline]`

Remove the last symbol.

Definition at line 181 of file Word.h.

References `ObjectOf< WordRep >::change()`, and `WordRep::pop_back()`.

#### **9.158.4.41 void Word::pop\_front () [inline]**

Remove the first symbol.

Definition at line 183 of file Word.h.

References `ObjectOf< WordRep >::change()`, and `WordRep::pop_front()`.

#### **9.158.4.42 Word Word::power (int *t*) const**

#### **9.158.4.43 ostream& Word::printOn (ostream & *os*) const [inline, private]**

Definition at line 334 of file Word.h.

References `InfiniteAlphabet::defaultAlphabet`, and `Alphabet::printWord()`.

#### **9.158.4.44 Word& Word::push\_back (const Word & *w*)**

Multiply the word by a word on the right. The result is being reduced.

#### **9.158.4.45 Word& Word::push\_back (int *gen*) [inline]**

Multiply the word by a one-letter word defined by *gen* on the right. The result is being reduced.

Definition at line 172 of file Word.h.

References `ObjectOf< WordRep >::change()`, and `WordRep::push_back()`.

Referenced by `DCBraidReduction::shorten()`.

#### **9.158.4.46 Word& Word::push\_front (const Word & *w*)**

Multiply the word by a word on the left. The result is being reduced.

#### **9.158.4.47 Word& Word::push\_front (int *gen*) [inline]**

Multiply the word by a one-letter word defined by *gen* on the left. The result is being reduced.

Definition at line 174 of file Word.h.

References `ObjectOf< WordRep >::change()`, and `WordRep::push_front()`.

**9.158.4.48** `static Word Word::randomWord (int n, int wLenMin, int wLenMax) [static]`

Generates a pseudo randomly reduced word of a length in [*wLenMin*,*wLenMax*] and over the alphabet  $X = \{x_1, \dots, x_n\}$ .

**9.158.4.49** `static Word Word::randomWord (int n, int wLen) [static]`

Generates a pseudo randomly reduced word of the length *wLen* over the alphabet  $X = \{x_1, \dots, x_n\}$ .

Referenced by `RandomPairGenerator::getFalsePair()`, and `RandomPairGenerator::getTruePair()`.

**9.158.4.50** `template<class ConstIntIterator > void Word::replace (WordIterator it, ConstIntIterator B, ConstIntIterator E) [inline]`

Replace a subword of a word starting at a position *it* by a word [B,E). The length of the word does not increase if [B,E) is longer than the terminal segment of the word [*it*,`end()` (p. 650)). In that case terminal symbols of [B,E) are ignored.

**9.158.4.51** `void Word::replace (int pos, const Generator & g)`

Replace a generator at a position *pos* by *g*.

**9.158.4.52** `void Word::replace (WordIterator it, const Generator & g)`

Replace a generator at a position *it* by *g*.

**9.158.4.53** `Word Word::replaceGenerators (const vector< Word > & images) const`

Returns a word in which generators are replaced by words. Replace generators with the words (*images*) contained in the vector

#### Parameters:

*images*

**9.158.4.54 Word Word::segment (int *from*, int *to*) const [inline]**

Definition at line 253 of file Word.h.

References `ObjectOf< Rep >::change()`.

**9.158.4.55 Word Word::terminalSegment (int *len*) const [inline]**

Get a terminal segment of the word of length *len*.

Definition at line 246 of file Word.h.

References `ObjectOf< Rep >::change()`.

**9.158.5 Friends And Related Function Documentation****9.158.5.1 ostream& operator<< (ostream & *os*, const Word & *w*) [friend]**

Reimplemented in `ThompsonGroupFNormalForm` (p. 559).

Definition at line 322 of file Word.h.

**9.158.5.2 istream& operator>> (istream & *is*, Word & *w*) [friend]**

Definition at line 328 of file Word.h.

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

- `Elt/include/Word.h`

## 9.159 WordDraw Class Reference

CREATES A PPM image of a table for braid word.

```
#include <WordDraw.h>
```

### Public Member Functions

- **WordDraw** (int *n*, const **Word** &*w*, bool *draw\_grid*=true)
- **WordDraw** (int *n*, const list< **Word** > &*w*, bool *draw\_grid*=true)
- ~**WordDraw** ()
- void **saveTo** (const string &*f\_name*)

### Private Member Functions

- void **drawCompressedBraid** (const **Word** &*theWord*, int *vert\_offset*=0)
- void **drawGenerator** (**Generator** *g*, int *pos*, int *vert\_offset*)
- void **drawVerticalGrid** (int *vpos*, int *color*)
- void **drawHorizontalGrid** (int *hpos*)

### Private Attributes

- int **N**
- **CImage** \* **theImage**
- int **ss**
- int **theLength**
- int **betBraids**

### 9.159.1 Detailed Description

CREATES A PPM image of a table for braid word.

Definition at line 30 of file WordDraw.h.

### 9.159.2 Constructor & Destructor Documentation

#### 9.159.2.1 **WordDraw::WordDraw** (int *n*, const **Word** &*w*, bool *draw\_grid* = true) [inline]

Definition at line 33 of file WordDraw.h.



References drawCompressedBraid(), drawHorizontalGrid(), drawVerticalGrid(), AImage::getHeight(), AImage::getWidth(), Word::length(), CImage::setBluePixel(), CImage::setGreenPixel(), CImage::setRedPixel(), ss, ssConst, theImage, and theLength.

#### 9.159.2.2 WordDraw::WordDraw (int *n*, const list< Word > & *w*, bool *draw\_grid* = true) [inline]

Definition at line 60 of file WordDraw.h.

References betBraids, drawCompressedBraid(), drawHorizontalGrid(), drawVerticalGrid(), ss, ssConst, theImage, and theLength.

#### 9.159.2.3 WordDraw::~~WordDraw () [inline]

Definition at line 101 of file WordDraw.h.

References theImage.

### 9.159.3 Member Function Documentation

#### 9.159.3.1 void WordDraw::drawCompressedBraid (const Word & *theWord*, int *vert\_offset* = 0) [inline, private]

Definition at line 113 of file WordDraw.h.

References Word::begin(), drawGenerator(), Word::end(), N, and theLength.

Referenced by WordDraw().

#### 9.159.3.2 void WordDraw::drawGenerator (Generator *g*, int *pos*, int *vert\_offset*) [inline, private]

Definition at line 128 of file WordDraw.h.

References CImage::setBluePixel(), CImage::setGreenPixel(), CImage::setRedPixel(), ss, and theImage.

Referenced by drawCompressedBraid().

#### 9.159.3.3 void WordDraw::drawHorizontalGrid (int *hpos*) [inline, private]

Definition at line 152 of file WordDraw.h.

References CImage::setBluePixel(), CImage::setGreenPixel(), CImage::setRedPixel(), ss, theImage, and theLength.

Referenced by WordDraw().

#### **9.159.3.4 void WordDraw::drawVerticalGrid (int *vpos*, int *color*) [inline, private]**

Definition at line 143 of file WordDraw.h.

References AImage::getHeight(), CImage::setBluePixel(), CImage::setGreenPixel(), CImage::setRedPixel(), and theImage.

Referenced by WordDraw().

#### **9.159.3.5 void WordDraw::saveTo (const string &*f\_name*) [inline]**

Definition at line 102 of file WordDraw.h.

References CImage::saveTo(), and theImage.

### **9.159.4 Member Data Documentation**

#### **9.159.4.1 int WordDraw::betBraids [private]**

Definition at line 165 of file WordDraw.h.

Referenced by WordDraw().

#### **9.159.4.2 int WordDraw::N [private]**

Definition at line 161 of file WordDraw.h.

Referenced by drawCompressedBraid().

#### **9.159.4.3 int WordDraw::ss [private]**

Definition at line 163 of file WordDraw.h.

Referenced by drawGenerator(), drawHorizontalGrid(), and WordDraw().

#### **9.159.4.4 CImage\* WordDraw::theImage [private]**

Definition at line 162 of file WordDraw.h.

Referenced by drawGenerator(), drawHorizontalGrid(), drawVerticalGrid(), saveTo(), WordDraw(), and ~WordDraw().

#### 9.159.4.5 int WordDraw::theLength [private]

Definition at line 164 of file WordDraw.h.

Referenced by drawCompressedBraid(), drawHorizontalGrid(), and WordDraw().

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

- Graphics/include/**WordDraw.h**

## 9.160 WordIterator Class Reference

```
#include <PowerWordIterator.h>
```

### Public Types

- typedef pair< int, int > **PII**
- typedef pair< int, int > **PII**

### Public Member Functions

- **WordIterator** ()
- **WordIterator** (Word &w, bool begin=true)
- bool **operator!=** (const **WordIterator** &WI) const
- bool **operator==** (const **WordIterator** &WI) const
- const **WordIterator** & **operator++** ()
- **WordIterator** **operator++** (int doomy)
- const **WordIterator** & **operator--** ()
- **WordIterator** **operator--** (int doomy)
- **PII** **operator\*** () const
- **WordIterator** ()
- **WordIterator** (Word &w, bool begin=true)
- bool **operator!=** (const **WordIterator** &WI) const
- bool **operator==** (const **WordIterator** &WI) const
- const **WordIterator** & **operator++** ()
- **WordIterator** **operator++** (int doomy)
- const **WordIterator** & **operator--** ()
- **WordIterator** **operator--** (int doomy)
- int **operator\*** () const

### Private Member Functions

- **WordIterator** (list< int > &lst, list< int >::iterator it)

### Private Attributes

- Word \* **theWord**
- list< **PII** >::iterator **theIterator**
- int **theOffset**
- list< int > \* **theList**
- list< int >::iterator **theIterator**

## Friends

- class **ConstWordIterator**
- class **WordRep**
- class **Word**

### 9.160.1 Detailed Description

Definition at line 23 of file PowerWordIterator.h.

### 9.160.2 Member Typedef Documentation

#### 9.160.2.1 `typedef pair< int , int > WordIterator::PII`

Definition at line 30 of file WordIterator.h.

#### 9.160.2.2 `typedef pair< int , int > WordIterator::PII`

Definition at line 28 of file PowerWordIterator.h.

### 9.160.3 Constructor & Destructor Documentation

9.160.3.1 `WordIterator::WordIterator ()`

9.160.3.2 `WordIterator::WordIterator (Word & w, bool begin = true)`

9.160.3.3 `WordIterator::WordIterator ()`

9.160.3.4 `WordIterator::WordIterator (Word & w, bool begin = true)`

9.160.3.5 `WordIterator::WordIterator (list< int > & lst, list< int >::iterator it) [private]`

### 9.160.4 Member Function Documentation

9.160.4.1 `bool WordIterator::operator!= (const WordIterator & WI) const`

9.160.4.2 `bool WordIterator::operator!= (const WordIterator & WI) const`

9.160.4.3 `int WordIterator::operator* () const`

9.160.4.4 `PII WordIterator::operator* () const`

9.160.4.5 `WordIterator WordIterator::operator++ (int doomy)`

9.160.4.6 `const WordIterator& WordIterator::operator++ ()`

9.160.4.7 `WordIterator WordIterator::operator++ (int doomy)`

9.160.4.8 `const WordIterator& WordIterator::operator++ ()`

9.160.4.9 `WordIterator WordIterator::operator-- (int doomy)`

9.160.4.10 `const WordIterator& WordIterator::operator-- ()`

9.160.4.11 `WordIterator WordIterator::operator-- (int doomy)`

9.160.4.12 `const WordIterator& WordIterator::operator-- ()`

9.160.4.13 `bool WordIterator::operator== (const WordIterator & WI) const`

9.160.4.14 `bool WordIterator::operator== (const WordIterator & WI) const`

### 9.160.5 Friends And Related Function Documentation

9.160.5.1 `ConstWordIterator [friend]`

**9.160.5.2 friend class Word [friend]**

Definition at line 27 of file WordIterator.h.

**9.160.5.3 friend class WordRep [friend]**

Definition at line 26 of file WordIterator.h.

**9.160.6 Member Data Documentation****9.160.6.1 list< int >::iterator WordIterator::theIterator [private]**

Definition at line 76 of file WordIterator.h.

**9.160.6.2 list< PII >::iterator WordIterator::theIterator [private]**

Definition at line 70 of file PowerWordIterator.h.

**9.160.6.3 list< int >\* WordIterator::theList [private]**

Definition at line 75 of file WordIterator.h.

**9.160.6.4 int WordIterator::theOffset [private]**

Definition at line 71 of file PowerWordIterator.h.

**9.160.6.5 Word\* WordIterator::theWord [private]**

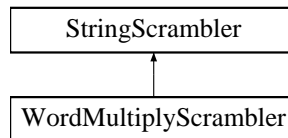
Definition at line 69 of file PowerWordIterator.h.

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

- Elt/include/**PowerWordIterator.h**
- Elt/include/**WordIterator.h**

## 9.161 WordMultiplyScrambler Class Reference

#include <StringScramblers.h> Inheritance diagram for WordMultiplyScrambler::



### Public Member Functions

- **WordMultiplyScrambler** (int *gs*, double *f*)
- **Word scramble** (const **Word** &) const
- double **fracChanged** () const

### Private Attributes

- double **theChangeFrac**
- int **numGens**

#### 9.161.1 Detailed Description

Definition at line 77 of file StringScramblers.h.

#### 9.161.2 Constructor & Destructor Documentation

##### 9.161.2.1 WordMultiplyScrambler::WordMultiplyScrambler (int *gs*, double *f*) [inline]

Definition at line 80 of file StringScramblers.h.

#### 9.161.3 Member Function Documentation

##### 9.161.3.1 double WordMultiplyScrambler::fracChanged () const [inline, virtual]

Implements **StringScrambler** (p. 519).

Definition at line 82 of file StringScramblers.h.



References theChangeFrac.

### 9.161.3.2 Word WordMultiplyScrambler::scramble (const Word &) const [virtual]

Implements StringScrambler (p. 519).

## 9.161.4 Member Data Documentation

### 9.161.4.1 int WordMultiplyScrambler::numGens [private]

Definition at line 86 of file StringScramblers.h.

### 9.161.4.2 double WordMultiplyScrambler::theChangeFrac [private]

Definition at line 85 of file StringScramblers.h.

Referenced by fracChanged().

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

- StringSimilarity/include/StringScramblers.h

## 9.162 WordPairComparison Class Reference

Implements a probabilistic measure for comparing two words.

```
#include <SimilarityMeasures.h>
```

### Public Member Functions

- **WordPairComparison** (int rank, const **StringSimilarityMeasure** \*sm)  
*Constructor.*
- vector< double > **distrEstimate** (int minLen, int maxLen, int nSamples=1000) const  
*Estimates the distribution of the distances (similarities) between two randomly generated words.*
- double **comparePair** (const **Word** &w1, const **Word** &w2, const vector< double > &measureDistr) const  
*Compare the pair of words using probabilistic measure.*
- double **comparePair** (const **Word** &w1, const **Word** &w2) const  
*Compare the pair of words using similarity measure.*

### Private Member Functions

- **WordPairComparison** (const **WordPairComparison** &)
- **WordPairComparison** & **operator=** (const **WordPairComparison** &)

### Private Attributes

- const **StringSimilarityMeasure** \* **pSSM**
- int **theRank**

#### 9.162.1 Detailed Description

Implements a probabilistic measure for comparing two words. Basic Idea: ...

Definition at line 37 of file SimilarityMeasures.h.

## 9.162.2 Constructor & Destructor Documentation

### 9.162.2.1 WordPairComparison::WordPairComparison (int *rank*, const StringSimilarityMeasure \* *sm*) [inline]

Constructor.

**Parameters:**

- rank* - the rank of a free group
- sm* - pointer to the corresponding similarity measure.

Definition at line 45 of file SimilarityMeasures.h.

### 9.162.2.2 WordPairComparison::WordPairComparison (const WordPairComparison &) [private]

## 9.162.3 Member Function Documentation

### 9.162.3.1 double WordPairComparison::comparePair (const Word & *w1*, const Word & *w2*) const

Compare the pair of words using similarity measure.

**Parameters:**

- w1* - the first word
- w2* - the second word.

**Returns:**

- the distance (value of the similarity measure)

### 9.162.3.2 double WordPairComparison::comparePair (const Word & *w1*, const Word & *w2*, const vector< double > & *measureDistr*) const

Compare the pair of words using probabilistic measure.

**Parameters:**

- w1* - the first word
- w2* - the second word.
- measureDistr* - distribution of the distance between two random words

**Returns:**

- an estimate of the probability of words *w1* and *w2* been generated independently

### 9.162.3.3 `vector<double> WordPairComparison::distrEstimate (int minLen, int maxLen, int nSamples = 1000) const`

Estimates the distribution of the distances (similarities) between two randomly generated words.

#### Parameters:

*minLen* - the minimal length of a randomly generated word

*maxLen* - the maximal length of a randomly generated word

*nSamples* - the number of pair samples to be generated (1000 is the default value).

#### Returns:

vector of the sorted distances (similarity measures)

### 9.162.3.4 `WordPairComparison& WordPairComparison::operator= (const WordPairComparison &) [private]`

## 9.162.4 Member Data Documentation

### 9.162.4.1 `const StringSimilarityMeasure* WordPairComparison::pSSM [private]`

Definition at line 76 of file SimilarityMeasures.h.

### 9.162.4.2 `int WordPairComparison::theRank [private]`

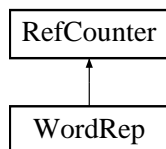
Definition at line 77 of file SimilarityMeasures.h.

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

- StringSimilarity/include/SimilarityMeasures.h

## 9.163 WordRep Class Reference

#include <WordRep.h> Inheritance diagram for WordRep::



### Public Member Functions

- **WordRep** \* **clone** () const
- const list< int > & **getList** () const
- list< int > & **getList** ()
- ostream & **printOn** (ostream &os) const

### Private Types

- typedef pair< int, int > **PII**

### Private Member Functions

- **WordRep** ()
- **WordRep** (const **WordRep** &wr)
- **WordRep** (const list< int > &gens)
- **WordRep** (const vector< int > &gens)
- template<class IntIterator >  
  **WordRep** (const IntIterator &B, const IntIterator &E)
- **WordRep** (int g)
- **WordRep operator=** (const **WordRep** wr)
- **WordRep & operator\*=** (const **WordRep** &w)
- bool **operator<** (const **WordRep** &wr) const
- bool **operator>** (const **WordRep** &wr) const
- bool **operator==** (const **WordRep** &wr) const
- **WordRep & operator^=** (const **WordRep** &conjugator)
- Conjugate a word by another word.*
- **WordRep operator^** (const **WordRep** &conjugator) const
- **WordRep & operator^=** (int power)
- Raise a word into a power.*

- **WordRep operator<sup>^</sup>** (int power) const
- **WordRep operator\*** (const **WordRep** &w) const
- bool **doesContain** (int gen) const
- int **length** () const
- int **exponentSum** (int gen) const
- int **isIn** (int gen) const
- int **getPower** (**WordRep** &base) const
- **WordRep inverse** () const
- void **clear** ()

*Make a word trivial.*

- void **freelyReduce** (**WordIterator** beg, **WordIterator** end)
- void **cyclicallyReduce** ()
- void **cyclicallyReduce** (**WordRep** &conjugator)
- void **cyclicLeftShift** ()
- void **cyclicRightShift** ()
- void **cyclicallyPermute** (int n)
- void **segment** (int from, int to)
- void **initialSegment** (int to)
- void **terminalSegment** (int from)
- template<class ConstIntIterator >  
void **insert** (int pos, ConstIntIterator B, ConstIntIterator E)
- template<class ConstIntIterator >  
void **insert** (**WordIterator** it, ConstIntIterator B, ConstIntIterator E)
- void **insert** (int pos, int g)
- void **insert** (**WordIterator** it, int g)
- void **replace** (**WordIterator** it, const **Generator** &g)
- void **replace** (int pos, const **Generator** &g)
- template<class ConstIntIterator >  
void **replace** (**WordIterator** it, ConstIntIterator B, ConstIntIterator E)
- void **push\_back** (int g)
- void **push\_front** (int g)
- void **pop\_back** ()
- void **pop\_front** ()

## Private Attributes

- list< int > **theElements**

## Friends

- class **Word**

### 9.163.1 Detailed Description

Definition at line 30 of file WordRep.h.

### 9.163.2 Member Typedef Documentation

#### 9.163.2.1 `typedef pair< int , int > WordRep::PII [private]`

Definition at line 33 of file WordRep.h.

### 9.163.3 Constructor & Destructor Documentation

#### 9.163.3.1 `WordRep::WordRep () [private]`

Referenced by clone().

#### 9.163.3.2 `WordRep::WordRep (const WordRep & wr) [private]`

#### 9.163.3.3 `WordRep::WordRep (const list< int > & gens) [private]`

#### 9.163.3.4 `WordRep::WordRep (const vector< int > & gens) [private]`

#### 9.163.3.5 `template<class IntIterator > WordRep::WordRep (const IntIterator & B, const IntIterator & E) [inline, private]`

Definition at line 49 of file WordRep.h.

References push\_back().

#### 9.163.3.6 `WordRep::WordRep (int g) [private]`

### 9.163.4 Member Function Documentation

#### 9.163.4.1 `void WordRep::clear () [inline, private]`

Make a word trivial.

Definition at line 138 of file WordRep.h.

References theElements.

**9.163.4.2 WordRep\* WordRep::clone () const [inline]**

Definition at line 108 of file WordRep.h.

References WordRep().

**9.163.4.3 void WordRep::cyclicallyPermute (int *n*) [private]****9.163.4.4 void WordRep::cyclicallyReduce (WordRep & *conjugator*) [private]****9.163.4.5 void WordRep::cyclicallyReduce () [private]****9.163.4.6 void WordRep::cyclicLeftShift () [private]**

Referenced by Word::cyclicLeftShift().

**9.163.4.7 void WordRep::cyclicRightShift () [private]**

Referenced by Word::cyclicRightShift().

**9.163.4.8 bool WordRep::doesContain (int *gen*) const [private]**

Referenced by Word::doesContain().

**9.163.4.9 int WordRep::exponentSum (int *gen*) const [private]**

Referenced by Word::exponentSum().

**9.163.4.10 void WordRep::freelyReduce (WordIterator *beg*, WordIterator *end*) [private]****9.163.4.11 list< int >& WordRep::getList () [inline]**

Definition at line 111 of file WordRep.h.

References theElements.

**9.163.4.12 const list< int >& WordRep::getList () const [inline]**

Definition at line 110 of file WordRep.h.

References theElements.



Referenced by Word::getList().

**9.163.4.13** `int WordRep::getPower (WordRep & base) const [private]`

Referenced by Word::getPower().

**9.163.4.14** `void WordRep::initialSegment (int to) [private]`

**9.163.4.15** `void WordRep::insert (WordIterator it, int g) [private]`

**9.163.4.16** `void WordRep::insert (int pos, int g) [private]`

**9.163.4.17** `template<class ConstIntIterator > void WordRep::insert  
(WordIterator it, ConstIntIterator B, ConstIntIterator E)  
[inline, private]`

**9.163.4.18** `template<class ConstIntIterator > void WordRep::insert (int pos,  
ConstIntIterator B, ConstIntIterator E) [inline, private]`

**9.163.4.19** `WordRep WordRep::inverse () const [private]`

Referenced by Word::inverse(), and Word::operator-().

**9.163.4.20** `int WordRep::isIn (int gen) const [private]`

Referenced by Word::isIn().

**9.163.4.21** `int WordRep::length () const [inline, private]`

Definition at line 118 of file WordRep.h.

References theElements.

Referenced by Word::length().

**9.163.4.22** `WordRep WordRep::operator* (const WordRep & w) const  
[inline, private]`

Definition at line 93 of file WordRep.h.

**9.163.4.23** `WordRep& WordRep::operator*=(const WordRep & w)`  
[private]

**9.163.4.24** `bool WordRep::operator<(const WordRep & wr) const`  
[private]

**9.163.4.25** `WordRep WordRep::operator=(const WordRep wr)` [private]

**9.163.4.26** `bool WordRep::operator==(const WordRep & wr) const`  
[private]

**9.163.4.27** `bool WordRep::operator>(const WordRep & wr) const`  
[private]

**9.163.4.28** `WordRep WordRep::operator^(int power) const` [inline,  
private]

Definition at line 86 of file WordRep.h.

**9.163.4.29** `WordRep WordRep::operator^(const WordRep & conjugator)`  
const [inline, private]

Definition at line 77 of file WordRep.h.

**9.163.4.30** `WordRep& WordRep::operator^=(int power)` [private]

Raise a word into a power.

**9.163.4.31** `WordRep& WordRep::operator^=(const WordRep & conjugator)`  
[private]

Conjugate a word by another word.

**9.163.4.32** `void WordRep::pop_back()` [inline, private]

Definition at line 196 of file WordRep.h.

References theElements.

Referenced by Word::pop\_back().

**9.163.4.33 void WordRep::pop\_front () [inline, private]**

Definition at line 197 of file WordRep.h.

References theElements.

Referenced by Word::pop\_front().

**9.163.4.34 ostream& WordRep::printOn (ostream & os) const****9.163.4.35 void WordRep::push\_back (int g) [private]**

Referenced by Word::push\_back(), and WordRep().

**9.163.4.36 void WordRep::push\_front (int g) [private]**

Referenced by Word::push\_front().

**9.163.4.37 template<class ConstIntIterator > void WordRep::replace (WordIterator it, ConstIntIterator B, ConstIntIterator E) [inline, private]****9.163.4.38 void WordRep::replace (int pos, const Generator & g) [private]****9.163.4.39 void WordRep::replace (WordIterator it, const Generator & g) [private]****9.163.4.40 void WordRep::segment (int from, int to) [private]****9.163.4.41 void WordRep::terminalSegment (int from) [private]****9.163.5 Friends And Related Function Documentation****9.163.5.1 friend class Word [friend]**

Definition at line 32 of file WordRep.h.

**9.163.6 Member Data Documentation****9.163.6.1 list< int > WordRep::theElements [private]**

Definition at line 207 of file WordRep.h.

Referenced by `clear()`, `getList()`, `length()`, `pop_back()`, and `pop_front()`.

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

- `Elt/include/WordRep.h`

## 9.164 YYSTYPE Union Reference

```
#include <AlphabetBisonGrammar.h>
```

### Public Attributes

- void \* **lst**
- void \* **str**
- int **num**

#### 9.164.1 Detailed Description

Definition at line 49 of file AlphabetBisonGrammar.h.

#### 9.164.2 Member Data Documentation

##### 9.164.2.1 void\* YYSTYPE::lst

Definition at line 55 of file AlphabetBisonGrammar.h.

##### 9.164.2.2 int YYSTYPE::num

Definition at line 57 of file AlphabetBisonGrammar.h.

##### 9.164.2.3 void\* YYSTYPE::str

Definition at line 56 of file AlphabetBisonGrammar.h.

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

- Alphabet/include/**AlphabetBisonGrammar.h**



## Chapter 10

# File Documentation

### 10.1 Alphabet/include/Alphabet.h File Reference

```
#include <iostream>
#include <vector>
#include <string>
#include "Parser.h"
#include <stdlib.h>
#include <sstream>
```

#### Classes

- class **Alphabet**  
*Implements an abstract interface for all alphabet realizations.*
- class **FiniteAlphabet**  
*Implements a finite size alphabet.*
- class **InfiniteAlphabet**  
*Implements an infinite size alphabet.*

#### Functions

- void **readFPPresentation** (istream &in)

## 10.1.1 Function Documentation

### 10.1.1.1 void readFPPresentation (istream & *in*)



## 10.2 Alphabet/include/AlphabetBisonGrammar.h File Reference

### Classes

- union **YYSTYPE**

### Defines

- `#define YYSTYPE_IS_TRIVIAL 1`
- `#define YYSTYPE YYSTYPE`
- `#define YYSTYPE_IS_DECLARED 1`

### Enumerations

- enum **yytokentype** { **ALETTER** = 258, **GENERATOR** = 258, **POWER** = 259 }

### Variables

- **YYSTYPE** aalval

#### 10.2.1 Define Documentation

##### 10.2.1.1 `#define YYSTYPE YYSTYPE`

Definition at line 65 of file AlphabetBisonGrammar.h.

##### 10.2.1.2 `#define YYSTYPE_IS_DECLARED 1`

Definition at line 66 of file AlphabetBisonGrammar.h.

##### 10.2.1.3 `#define YYSTYPE_IS_TRIVIAL 1`

Definition at line 64 of file AlphabetBisonGrammar.h.

## 10.2.2 Enumeration Type Documentation

### 10.2.2.1 enum yytokentype

Enumerator:

*ALETTER*

*GENERATOR*

*POWER*

Definition at line 41 of file AlphabetBisonGrammar.h.

## 10.2.3 Variable Documentation

### 10.2.3.1 YYSTYPE aalval

## 10.3 Alphabet/include/Parser.h File Reference

```
#include <iostream>
```

```
#include <list>
```

### Classes

- class **Parser**
- class **AParser**

### Enumerations

- enum **AInputType** { **NONE**, **SET**, **FREE\_PRE**, **REL\_PRE** }

#### 10.3.1 Enumeration Type Documentation

##### 10.3.1.1 enum AInputType

###### Enumerator:

*NONE*

*SET*

*FREE\_PRE*

*REL\_PRE*

Definition at line 13 of file Parser.h.

## 10.4 Alphabet/include/WordBisonGrammar.h File Reference

### Enumerations

- enum yytokentype { **ALETTER** = 258, **GENERATOR** = 258, **POWER** = 259 }

### Variables

- YYSTYPE yylval

### 10.4.1 Enumeration Type Documentation

#### 10.4.1.1 enum yytokentype

Enumerator:

***ALETTER***

***GENERATOR***

***POWER***

Definition at line 41 of file WordBisonGrammar.h.

### 10.4.2 Variable Documentation

#### 10.4.2.1 YYSTYPE yylval

## 10.5 BraidGroup/include/BKLLeftNormalForm.h File Reference

```
#include "tuples.h"
#include "Permutation.h"
#include "list"
```

### Classes

- class **BKLLeftNormalForm**  
*Birman-Ko-Lee left normal form.*

### Functions

- ostream & **operator<<** (ostream &os, const **BKLLeftNormalForm** &bkl)

#### 10.5.1 Function Documentation

- 10.5.1.1** ostream& **operator<<** (ostream & *os*, const **BKLLeftNormalForm** & *bkl*)

## 10.6 BraidGroup/include/BKLRightNormalForm.h File Reference

```
#include "tuples.h"
#include "Permutation.h"
#include "vector"
#include "list"
```

### Classes

- class **BKLRightNormalForm**  
*(Contains errors!!! Not to be used yet) Birman-Ko-Lee right normal form.*

### Defines

- #define **ERROR\_EXISTS**

### Functions

- ostream & **operator<<** (ostream &os, const **BKLRightNormalForm** &rep)

#### 10.6.1 Define Documentation

##### 10.6.1.1 #define ERROR\_EXISTS

Definition at line 23 of file BKLRightNormalForm.h.

#### 10.6.2 Function Documentation

##### 10.6.2.1 ostream& operator<< (ostream & os, const BKLRightNormalForm & rep)

## 10.7 BraidGroup/include/BraidGroup.h File Reference

```
#include "Word.h"
```

### Classes

- class **BraidGroup**

*Class **BraidGroup** (p. 140) (defines a representation of a Braid Group)//.*

## 10.8 BraidGroup/include/DehornoyForm.h File Reference

```
#include "Word.h"
```

### Classes

- class **DehornoyForm**

*Dehornoy Form of a braid word (aka/ handle free form).*



## 10.9 BraidGroup/include/DehornoyForm\_old.h File Reference

```
#include "Word.h"
```

```
#include "LinkedBraidStructure.h"
```

### Classes

- class **DehornoyForm**

*Dehornoy Form of a braid word (aka/ handle free form).*

## 10.10 BraidGroup/include/LinkedBraidStructure.h File Reference

```
#include "set"  
#include "map"  
#include "list"  
#include "vector"  
#include "tuples.h"
```

### Classes

- struct **BraidNode**  
*Defines a crossing in a linked braid structure//.*
- struct **LinkedBraidStructureTransform**
- class **LinkedBraidStructure**

### Functions

- ostream & **operator<<** (ostream &os, const **BraidNode** &bn)

#### 10.10.1 Function Documentation

##### 10.10.1.1 ostream& operator<< (ostream & os, const BraidNode & bn)

## 10.11 BraidGroup/include/LinkedBraidStructure\_old.h File Reference

```
#include "list"  
#include "vector"
```

### Classes

- struct **BraidNode**  
*Defines a crossing in a linked braid structure//.*
- class **LinkedBraidStructure**

## 10.12 BraidGroup/include/ShortBraidForm.h File Reference

```
#include <vector>
```

### Functions

- **Word shortenBraid** (int  $N$ , const **Word** & $w$ )
- **Word shortBraidForm** (int  $N$ , const **Word** & $w$ )
- **vector< Word > shortBraidSbgpForm** (int  $N$ , const vector< **Word** > & $w$ )
- **LinkedBraidStructure shortenLBS** (LinkedBraidStructure & $lbs$ )

### 10.12.1 Function Documentation

**10.12.1.1 Word shortBraidForm** (int  $N$ , const **Word** & $w$ )

**10.12.1.2 vector< Word > shortBraidSbgpForm** (int  $N$ , const vector< **Word** > & $w$ )

**10.12.1.3 Word shortenBraid** (int  $N$ , const **Word** & $w$ )

Referenced by DCBraidReduction::shorten().

**10.12.1.4 LinkedBraidStructure shortenLBS** (LinkedBraidStructure &  $lbs$ )

## 10.13 BraidGroup/include/ShortBraidForm\_old.h File Reference

```
#include <vector>
```

### Functions

- **Word shortenBraid** (int  $N$ , const **Word** & $w$ )
- **Word shortBraidForm** (int  $N$ , const **Word** & $w$ )
- **vector< Word > shortBraidSbgpForm** (int  $N$ , const vector< **Word** > & $w$ )

### 10.13.1 Function Documentation

**10.13.1.1 Word shortBraidForm** (int  $N$ , const **Word** & $w$ )

**10.13.1.2 vector< Word > shortBraidSbgpForm** (int  $N$ , const vector< **Word** > & $w$ )

**10.13.1.3 Word shortenBraid** (int  $N$ , const **Word** & $w$ )

## 10.14 BraidGroup/include/ThLeftNormalForm.h File Reference

```
#include "tuples.h"
#include "Permutation.h"
#include "vector"
#include "list"
```

### Classes

- class **ThLeftNormalForm**  
*Defines a representation of a left Garside normal form.*

### Functions

- ostream & **operator<<** (ostream &os, const **ThLeftNormalForm** &rep)

#### 10.14.1 Function Documentation

- 10.14.1.1 ostream& **operator<<** (ostream &os, const **ThLeftNormalForm** &rep)

## 10.15 BraidGroup/include/ThRightNormalForm.h File Reference

```
#include "tuples.h"
#include "Permutation.h"
#include "vector"
#include "list"
```

### Classes

- class **ThRightNormalForm**  
*Defines a representation of a right Garside normal form.*

### Functions

- ostream & **operator**<< (ostream &os, const **ThRightNormalForm** &nf)

#### 10.15.1 Function Documentation

- 10.15.1.1** ostream& **operator**<< (ostream &os, const **ThRightNormalForm** &nf)

## 10.16 BraidGroup/include/ThRightNormalFormAlgorithms.h File Reference

```
#include "set"
#include "ThRightNormalForm.h"
```

### Functions

- `pair< vector< ThRightNormalForm >, ThRightNormalForm > getSummitSetRepresentative (int rank, const vector< ThRightNormalForm > &elts)`

*Compute summit set representatives for a tuple of braids.*

- `Permutation getSimpleConjugator (int rank, const vector< ThRightNormalForm > &tuple, const Permutation &start)`

*Compute a simple conjugator for "tuple" starting from "start".*

- `set< Permutation > getSimpleConjugators (int rank, const vector< ThRightNormalForm > &tuple)`

### 10.16.1 Function Documentation

#### 10.16.1.1 Permutation getSimpleConjugator (int *rank*, const vector< ThRightNormalForm > & *tuple*, const Permutation & *start*)

Compute a simple conjugator for "tuple" starting from "start". Algorithm from Gonzalez-Menese, "Improving an algorithm to solve Multiple Simultaneous Conjugacy problem for braid groups". Conjugating by a simple element does not decrease the infimum of the tuple.

#### 10.16.1.2 set< Permutation > getSimpleConjugators (int *rank*, const vector< ThRightNormalForm > & *tuple*)

#### 10.16.1.3 pair< vector< ThRightNormalForm >, ThRightNormalForm > getSummitSetRepresentative (int *rank*, const vector< ThRightNormalForm > & *elts*)

Compute summit set representatives for a tuple of braids.



## 10.17 CryptoAAG/include/AAGChallengeGeneration.h File Reference

```
#include "BraidGroup.h"
#include "ThRightNormalForm.h"
#include <iterator>
#include <iostream>
#include <fstream>
#include "ShortBraidForm.h"
#include "LengthAttack.h"
#include "AAGKeyGeneration.h"
```

### Namespaces

- namespace **AAGChallenge**

### Functions

- **vector< Word > AAGChallenge::getDpSubgroup** (int N, int Pgens, const vector< Word > &Prelts, const pair< vector< Word >, vector< Word > > &genComps, int conjLen)
- **Word AAGChallenge::getDelta** (int i, int c)
- **pair< vector< Word >, vector< Word > > AAGChallenge::getSgGenComponentsRandom** (int Pgens, int c, int k, int len)
- **pair< vector< Word >, vector< Word > > AAGChallenge::getSgGenComponentsSquares** (int Pgens, int c, int k)
- **Word AAGChallenge::randomSubgroupWord** (int N, const vector< Word > &sg)
- **Word AAGChallenge::specialSubgroupWord** (const Word &a, const Word &t, int n)
- **Word AAGChallenge::specialSubgroupWordRandom** (const Word &a, const Word &t, int len)
- **vector< Word > AAGChallenge::generateSubgroup** (int c, int k, int comp\_len)
- **Word AAGChallenge::generateKeyDecomp** (int n)

## 10.18 CryptoAAG/include/AAGKeyGeneration.h File Reference

```
#include "Word.h"  
#include <vector>
```

### Classes

- class **AAGProtocolInstance**

### Functions

- `vector< Word > conjugateSubgroup_PGBF (int N, const vector< Word > &sbgp, Word w, bool useForm=true)`

#### 10.18.1 Function Documentation

- 10.18.1.1** `vector< Word > conjugateSubgroup_PGBF (int N, const vector< Word > &sbgp, Word w, bool useForm = true)`

## 10.19 CryptoAAG/include/LengthAttack.h File Reference

```
#include "Word.h"  
#include <vector>
```

### Classes

- class **LengthAttackBase**  
*Basis interface for classes implementing the Length-Based attack.*
- class **LengthAttack\_A1**  
*Implements Length-based attack on AAG protocol.*
- class **LengthAttack\_A2**  
*Implements Length-based attack on AAG protocol.*
- class **LengthAttack\_A3**  
*Implements Length-based attack on AAG protocol.*

### Defines

- #define **AL1** 1
- #define **AL2** 2
- #define **AL3** 3

### Typedefs

- typedef pair< int, vector< **Word** > > **ELT**

### Enumerations

- enum **findKey\_LengthBasedResult** { **FAILED**, **TIME\_EXPIRED**, **SUCCESSFULL** }

### 10.19.1 Define Documentation

#### 10.19.1.1 `#define AL1 1`

Definition at line 22 of file LengthAttack.h.

Referenced by LengthAttack\_A1::type().

#### 10.19.1.2 `#define AL2 2`

Definition at line 23 of file LengthAttack.h.

Referenced by LengthAttack\_A2::type().

#### 10.19.1.3 `#define AL3 3`

Definition at line 24 of file LengthAttack.h.

Referenced by LengthAttack\_A3::type().

### 10.19.2 Typedef Documentation

#### 10.19.2.1 `typedef pair< int , vector< Word > > ELT`

Definition at line 26 of file LengthAttack.h.

### 10.19.3 Enumeration Type Documentation

#### 10.19.3.1 `enum findKey_LengthBasedResult`

Enumerator:

*FAILED*

*TIME\_EXPIRED*

*SUCCESSFULL*

Definition at line 16 of file LengthAttack.h.

## 10.20 CryptoAAG/include/MajorDump.h File Reference

```
#include <fstream>
```

### Classes

- class **Dump**

## 10.21 CryptoAE/include/AEProtocol.h File Reference

```
#include "Word.h"
#include "ThLeftNormalForm.h"
#include <vector>
#include <utility>
```

### Classes

- struct **TTP\_Conf**  
*Set of parameters required to construct the protocol instance.*
- class **BSets**  
*Implements construction of the initial commuting sets of subgroup generators.*
- class **TPTuple**  
*Implements tuples corresponding to the putput of TTP algorithm.*
- class **MatrixFp**
- class **AEKeyExchange**

### Typedefs

- typedef pair< **MatrixFp**, **Permutation** > **ProdElement**
- typedef pair< int, **Permutation** > **BureauGenerator**

#### 10.21.1 Typedef Documentation

##### 10.21.1.1 typedef pair<int,Permutation> BureauGenerator

Definition at line 199 of file AEProtocol.h.

##### 10.21.1.2 typedef pair<MatrixFp,Permutation> ProdElement

Definition at line 198 of file AEProtocol.h.

## 10.22 CryptoAE/include/TTPAttack.h File Reference

```
#include "Word.h"
#include "AEProtocol.h"
#include "ThLeftNormalForm.h"
#include <vector>
```

### Classes

- class **TTPLBA**
- class **TTPAttack**

*This is an implementation of an attack on TTP algorithm for generating public sets of generators of the Algebraic Eraser protocol.*

### Typedefs

- typedef pair< int, **TTPTuple** > **NODE**

#### 10.22.1 Typedef Documentation

##### 10.22.1.1 typedef pair< int , TTPTuple > NODE

Definition at line 25 of file TTPAttack.h.

## 10.23 CryptoKL/include/KLKeyGeneration.h File Reference

```
#include "Word.h"  
#include <vector>
```

### Classes

- class **KLProtocolInstance**



## 10.24 CryptoShftConj/include/ShftConjKeyGeneration.h File Reference

```
#include "Word.h"  
#include <vector>
```

### Classes

- class **ShftConjKeyInstance**

### Functions

- **Word** generatorShift (const **Word** &w)

#### 10.24.1 Function Documentation

##### 10.24.1.1 Word generatorShift (const Word & w)

## 10.25 CryptoShftConj/include/ShftConjKeyGenerationGarside.h File Reference

```
#include "Word.h"  
#include "ThRightNormalForm.h"  
#include <vector>
```

### Classes

- class **ShftConjKeyInstanceGarside**

*Container for public and private information in Dehornoy's authentication protocol.*

### Functions

- **ThRightNormalForm shiftedConjugation** (const **ThRightNormalForm** &w,  
const **ThRightNormalForm** &c)

### 10.25.1 Function Documentation

- 10.25.1.1 **ThRightNormalForm shiftedConjugation** (const  
**ThRightNormalForm** & w, const **ThRightNormalForm** & c)

## **10.26   CryptoTripleDecomposition/include/TripleDecompositionKeyGeneration.h**

### **File Reference**

```
#include "tuples.h"
#include "Word.h"
#include "ThRightNormalForm.h"
#include <vector>
```

### **Classes**

- class **TripleDecompositionProtocolInstance**

*Definition of the class **TripleDecompositionProtocolInstance** (p. 580).*

## 10.27 `Elt/include/PowerWord.h` File Reference

```
#include "ObjectOf.h"
#include "PowerWord.h"
#include "WordRep.h"
#include "PowerWordIterator.h"
#include <string>
#include <strstream>
```

### Classes

- class **PowerWord**

## 10.28 Elt/include/PowerWordIterator.h File Reference

```
#include "list"
```

### Classes

- class **WordIterator**
- class **ConstWordIterator**

## 10.29 `Elt/include/PowerWordRep.h` File Reference

```
#include "RefCounter.h"
#include "vector"
#include "list"
#include "iostream"
```

### Classes

- class **PowerWordRep**

## 10.30 Elt/include/Word.h File Reference

```
#include "ObjectOf.h"
#include "Word.h"
#include "WordRep.h"
#include "WordIterator.h"
#include <string>
#include "Alphabet.h"
#include <set>
```

### Classes

- class **Word**

*Class **Word** (p. 642) (defines a representation of a **Word** (p. 642) over a group alphabet)//.*

## 10.31 `Elt/include/WordIterator.h` File Reference

```
#include "list"
```

### Classes

- class **WordIterator**
- class **ConstWordIterator**



## 10.32 Elt/include/WordRep.h File Reference

```
#include "RefCounter.h"
#include "vector"
#include "list"
#include "iostream"
```

### Classes

- class **WordRep**

### Typedefs

- typedef int **Generator**

#### 10.32.1 Typedef Documentation

##### 10.32.1.1 typedef int Generator

Definition at line 21 of file WordRep.h.

### 10.33 Equation/include/Equation.h File Reference

```
#include "GraphType.h"  
#include "GraphConcept.h"  
#include "GraphConceptAlgorithms.h"  
#include "Word.h"
```

#### Classes

- class **Equation**

## **10.34 Equation/include/QuadEquatTransformationGraph.h File Reference**

```
#include "Equation.h"
```

### **Classes**

- class **QuadEquationTranformationGraph**

## 10.35 Experiments/include/AAGKeyPairGenerator.h

### File Reference

```
#include "errormsgs.h"
#include "PairDistanceTest.h"
#include "AAGKeyGeneration.h"
```

### Classes

- class **AAGKeyPairGenerator**

## 10.36 Experiments/include/DCBraidReduction.h File Reference

```
#include <list>
#include "Word.h"
#include "ShortBraidForm.h"
#include <iterator>
```

### Classes

- class **Partition**
- class **Perturbation**
- class **UniformPartition**
- struct **CommDivider**
- struct **ICommDivider**
- class **MaxCommutePartition**
- class **DCBraidReduction**

## 10.37 Experiments/include/DDL.h File Reference

```
#include "Word.h"
```

### Classes

- struct **DDLNode**
- struct **DDL**

## 10.38 Experiments/include/FRDFIT.h File Reference

```
#include "Word.h"
```

```
#include "DDL.h"
```

### Classes

- class **FRDFIT**

## 10.39 FreeGroup/include/FreeGroup.h File Reference

```
#include "SubgroupFG.h"  
#include "Alphabet.h"  
#include "Word.h"
```

### Classes

- class **FreeGroup**



## 10.40 FreeGroup/include/StraightLineProgramWord.h File Reference

```
#include "map"
#include "list"
#include "gmpxx.h"
#include "Map.h"
```

### Classes

- class **StraightLineProgramWord**  
*class **StraightLineProgramWord** (p. 507).*
- struct **StraightLineProgramWord::Production**  
*A production for a rule of a composition system.*
- struct **StraightLineProgramWord::Assertion**  
*Structure used in function **equal()** (p. 512) only.*

### Typedefs

- typedef mpz\_class **LongInteger**

#### 10.40.1 Typedef Documentation

##### 10.40.1.1 typedef mpz\_class LongInteger

Definition at line 20 of file StraightLineProgramWord.h.

## 10.41 FreeGroup/include/WhiteheadGraph.h File Reference

```
#include "Word.h"  
#include <vector>  
#include "errormsgs.h"
```

### Classes

- class **CutVertices**  
*Implements an algorithm to find all articulation points of a graph//.*
- class **WhiteheadGraph**  
*Interface class for Whitehead graphs.*
- class **WhiteheadSimpleGraph**  
*Implements Whitehead Simple graph (i.e. no multiple edges or loops allowed).*
- class **WhiteheadMultiGraph**  
*Whitehead Multi-Graph (Not implemented yet).*

### Typedefs

- typedef int **Generator**

#### 10.41.1 Typedef Documentation

##### 10.41.1.1 typedef int Generator

Definition at line 20 of file WhiteheadGraph.h.

## 10.42 SbgpFG/include/WhiteheadGraph.h File Reference

```
#include "GraphType.h"
#include "GraphConcept.h"
#include "GraphConceptAlgorithms.h"
#include "Word.h"
```

### Classes

- class **WhiteheadGraph**  
*Interface class for Whitehead graphs.*

## 10.43 FreeMetabelianGroup/include/FreeMetabelianGroupAlgorithms.h

### File Reference

```
#include "map"
#include "list"
#include "set"
#include "tuples.h"
#include "Word.h"
```

### Classes

- class **FreeMetabelianGroupAlgorithms**

*Static class **FreeMetabelianGroupAlgorithms** (p. 214) encapsulates algorithms for FreeMetabelianGroup groups.*

## 10.44 general/include/BalancedTree.h File Reference

```
#include "list"  
#include "iostream"
```

### Classes

- class **BalancedTree**< **Obj** >  
*Class **BalancedTree** (p. 111) (container class intended to keep an ordered sequence of objects, supports fast  $n \log n$  insertions).*
- struct **BalancedTree**< **Obj** >::**BTNode**

## 10.45 general/include/ConfigFile.h File Reference

```
#include <string.h>
#include <sstream>
#include <iostream>
#include <map>
#include <stdlib.h>
```

### Classes

- struct **ltstr**

*Implements a comparison operator on two strings.*

- class **Value**

*Implements data types of parameters obtained from a configuration file.*

- class **ConfigFile**

*Implements a mechanism for passing parameters from a configuration file.*

### Typedefs

- typedef map< string, **Value**, **ltstr** > **ParameterType**

*Implements a mapping from parameter's name into its value.*

#### 10.45.1 Typedef Documentation

##### 10.45.1.1 typedef map<string, Value, ltstr> ParameterType

Implements a mapping from parameter's name into its value.

Definition at line 85 of file ConfigFile.h.

## 10.46 general/include/dump.h File Reference

```
#include <iostream>
```

```
#include <vector>
```

### Classes

- class **dump**< **T** >

## 10.47 general/include/errormsgs.h File Reference

```
#include <iostream>
```

### Namespaces

- namespace **msgs**  
*Errors output handling.*

### Functions

- void **msgs::error** (const char \*msg)  
*Print a message to the standard error output.*
- void **msgs::error** (const string &msg)  
*Print a message to the standard error output.*
- bool **msgs::error** (bool exp, const char \*msg)  
*Conditional output of an error message to the standard error output.*
- void **msgs::warn** (const char \*msg)  
*Print a message to the standard error output.*



## 10.48 general/include/FormatOutput.h File Reference

```
#include <iostream>
```

### Classes

- struct **PBar**

## 10.49 general/include/ObjectOf.h File Reference

```
#include <iostream>
#include "RefCount.h"
```

### Classes

- class **ObjectOf**< **Rep** >

## 10.50 general/include/Permutation.h File Reference

```
#include <set>
#include <map>
#include <vector>
#include <iostream>
```

### Classes

- class **Permutation**  
*Permutation* (p. 371).
- struct **Permutation::triple**

## 10.51 `general/include/PermutationEnumerator.h` File Reference

```
#include "VectorEnumerator.h"  
#include "Permutation.h"
```

### Classes

- class **PermutationEnumerator**

## 10.52 general/include/ProgressBar.h File Reference

```
#include <iostream>
```

### Classes

- struct **PBar**

## 10.53 general/include/RefCounter.h File Reference

### Classes

- class **RefCounter**

## 10.54 general/include/tuples.h File Reference

```
#include "iostream"
```

### Classes

- class **triple**< T1, T2, T3 >
- class **quadruple**< T1, T2, T3, T4 >
- class **quintuple**< T1, T2, T3, T4, T5 >

## 10.55 `general/include/VectorEnumerator.h` File Reference

```
#include <vector>
#include <iostream>
```

### Classes

- class **VectorEnumerator**  
*Class **VectorEnumerator** (p. 620).*



## 10.56 Graph/include/FSA.h File Reference

```
#include "FSARep.h"  
#include "ObjectOf.h"
```

### Classes

- class **FSA**

### Functions

- ostream & **operator**<< (ostream &os, const **FSA** &g)

#### 10.56.1 Function Documentation

##### 10.56.1.1 ostream& operator<< (ostream & *os*, const **FSA** & *g*)

## 10.57 Graph/include/FSARep.h File Reference

```
#include "RefCounter.h"
#include "map"
#include "set"
#include "list"
#include "vector"
#include <stdlib.h>
```

### Classes

- struct **FSAEdge**
- struct **FSAState**
- struct **FoldDetails**
- class **FSARep**

### Functions

- void **reducePath** (list< **FSAEdge** > &path, int init\_state)  
*to be checked!!!*

#### 10.57.1 Function Documentation

**10.57.1.1** void **reducePath** (list< **FSAEdge** > & *path*, int *init\_state*)

to be checked!!!

## 10.58 Graph/include/Graph.h File Reference

```
#include "ObjectOf.h"
#include "GraphRep.h"
#include "vector"
```

### Classes

- class **Graph**

### Functions

- ostream & **operator<<** (ostream &os, const **Graph** &g)
- **Graph randomGraph** (int N, float edge\_param)  
*Function generates a random (non-directed) graph on N vertices. Each edge has equal probability edge\_param to appear in the result.*
- vector< vector< int > > **lengthTable** (const **Graph** &G)  
*Compute a table of all lengths in the directed graph G.*
- vector< vector< int > > **innerProductTable** (const **Graph** &G, int origin)  
*Compute a table of inner (Gromov's) products in the directed graph G.*
- float **getHyperbolicityConst** (const **Graph** &G)  
*For a finite directed graph G compute a constant of hyperbolicity.*

### 10.58.1 Function Documentation

#### 10.58.1.1 float getHyperbolicityConst (const Graph & G)

For a finite directed graph G compute a constant of hyperbolicity.

#### 10.58.1.2 vector< vector< int > > innerProductTable (const Graph & G, int origin)

Compute a table of inner (Gromov's) products in the directed graph G.

#### 10.58.1.3 vector< vector< int > > lengthTable (const Graph & G)

Compute a table of all lengths in the directed graph G.

**10.58.1.4 ostream& operator<< (ostream & *os*, const Graph & *g*)****10.58.1.5 Graph randomGraph (int *N*, float *edge\_param*)**

Function generates a random (non-directed) graph on *N* vertices. Each edge has equal probability *edge\_param* to appear in the result.

## 10.59 Graph/include/GraphAlgorithms.h File Reference

```
#include "iostream"
#include "map"
#include "set"
#include "list"
#include <stdlib.h>
```

### Functions

- `template<class Graph >`  
`map< int, typename Graph::edge_type > getGeodesicTree_in (const Graph`  
`&graph, int init_st)`
- `template<class Graph >`  
`map< int, typename Graph::edge_type > getGeodesicTree_out (const Graph`  
`&graph, int init_st)`
- `template<class Graph >`  
`map< int, int > getDistances_out (const Graph &graph, int init_st)`
- `template<class edge_type >`  
`list< edge_type > readoffGeodesicTree (const map< int, edge_type > &tree,`  
`int st_num)`

*Function finds a path in the tree starting from the state st\_num to the root.*

- `template<class LabelledGraph , class ConstIterator >`  
`int trace (const LabelledGraph &LG, int init, ConstIterator B, ConstIterator E)`
- `template<class LabelledGraph , class ConstIterator >`  
`pair< bool, list< typename LabelledGraph::edge_type > > trace_path (const`  
`LabelledGraph &LG, int init, ConstIterator B, ConstIterator E)`

### 10.59.1 Function Documentation

#### 10.59.1.1 `template<class Graph > map< int , int > getDistances_out (const Graph & graph, int init_st) [inline]`

Definition at line 148 of file GraphAlgorithms.h.

References `Graph::getStates()`.

**10.59.1.2** `template<class Graph > map< int , typename Graph::edge_type >  
getGeodesicTree_in (const Graph & graph, int init_st) [inline]`

Definition at line 28 of file GraphAlgorithms.h.

References Graph::getStates().

**10.59.1.3** `template<class Graph > map< int , typename Graph::edge_type >  
getGeodesicTree_out (const Graph & graph, int init_st) [inline]`

Definition at line 88 of file GraphAlgorithms.h.

References Graph::getStates().

**10.59.1.4** `template<class edge_type > list< edge_type > readoffGeodesicTree  
(const map< int, edge_type > & tree, int st_num) [inline]`

Function finds a path in the tree starting from the state *st\_num* to the root.

Definition at line 206 of file GraphAlgorithms.h.

**10.59.1.5** `template<class LabelledGraph , class ConstIterator > int trace (const  
LabelledGraph & LG, int init, ConstIterator B, ConstIterator E)  
[inline]`

Definition at line 233 of file GraphAlgorithms.h.

**10.59.1.6** `template<class LabelledGraph , class ConstIterator > pair< bool ,  
list< typename LabelledGraph::edge_type > > trace_path (const  
LabelledGraph & LG, int init, ConstIterator B, ConstIterator E)  
[inline]`

Definition at line 272 of file GraphAlgorithms.h.

## 10.60 Graph/include/GraphConcept.h File Reference

```
#include "RefCount.h"
#include "ObjectOf.h"
#include "set"
#include "map"
```

### Classes

- class **Graphs::GraphConceptRep**< VertexType, EdgeType >  
*Representation class for graph types.*
- class **Graphs::GraphConcept**< VertexType, EdgeType >  
*The main class for graph types.*

### Namespaces

- namespace **Graphs**

### Functions

- template<class VertexType , class EdgeType >  
ostream & **Graphs::operator**<< (ostream &os, const GraphConcept< VertexType, EdgeType > &G)  
*Output the graph.*

## 10.61 Graph/include/GraphConceptAlgorithms.h File Reference

```
#include <iostream>
#include <map>
#include <set>
#include <list>
#include <stdlib.h>
```

### Classes

- struct **Graphs::FoldDetails**< VertexType, EdgeType >

### Namespaces

- namespace **Graphs**

### Functions

- template<class Graph >  
map< int, typename **Graph::edge\_type** > **Graphs::getGeodesicTree\_in**  
(const **Graph** &graph, int init\_v)  
*Compute directed geodesic tree "inward" to the vertex init\_v.*
- template<class Graph >  
map< int, typename **Graph::edge\_type** > **Graphs::getGeodesicTree\_out**  
(const **Graph** &graph, int init\_v)  
*Compute geodesic directed tree starting the vertex init\_v.*
- template<class Graph >  
map< int, int > **Graphs::getDistances\_out** (const **Graph** &graph, int init\_v)  
*Compute distances from the vertex init\_v.*
- template<class Graph >  
map< int, int > **Graphs::getDistances\_in** (const **Graph** &graph, int init\_v)  
*Compute distances to the vertex init\_v.*
- template<class edge\_type >  
pair< bool, list< edge\_type > > **Graphs::readoffGeodesicTree** (const map<  
int, edge\_type > &tree, int v)



*Function finds a path in the tree starting from the state st\_num to the root.*

- `template<class LabelledGraph , class ConstIterator >`  
`int Graphs::trace (const LabelledGraph &LG, int init_v, ConstIterator B, ConstIterator E)`
- `template<class LabelledGraph , class ConstIterator >`  
`pair< bool, list< typename LabelledGraph::edge_type > > Graphs::trace_path (const LabelledGraph &LG, int init_v, ConstIterator B, ConstIterator E)`
- `template<class LabelledGraph >`  
`void Graphs::fold (LabelledGraph &G, set< int > candidates, list< FoldDetails< typename LabelledGraph::vertex_type, typename LabelledGraph::edge_type > > *details=NULL)`

*Fold a labelled graph. To apply this function the graph edges must have theLabel defined.*

- `template<class LabelledGraph >`  
`void Graphs::fold (const LabelledGraph &G, int candidate, list< FoldDetails< typename LabelledGraph::vertex_type, typename LabelledGraph::edge_type > > *details=NULL)`
- `template<class LabelledGraph >`  
`void Graphs::fold (const LabelledGraph &G, list< FoldDetails< typename LabelledGraph::vertex_type, typename LabelledGraph::edge_type > > *details=NULL)`
- `template<class LabelledGraph , class FoldDetailsConstIterator >`  
`void Graphs::unfold (LabelledGraph &G, FoldDetailsConstIterator B, FoldDetailsConstIterator E)`

*Revert the folding.*

- `template<class LabelledGraph , class FoldDetailsConstIterator >`  
`void Graphs::liftup (const LabelledGraph &graph, int init_vertex, list< typename LabelledGraph::edge_type > &path, FoldDetailsConstIterator B, FoldDetailsConstIterator E)`

*Revert the folding.*

- `template<class edge_type >`  
`void Graphs::reduce_path (int init_vertex, list< edge_type > &path)`

*Reduce a path (remove all pairs of consequent inverse edges).*

## 10.62 Graph/include/GraphDrawingAttributes.h File Reference

```
#include "tuples.h"  
#include "map"  
#include "set"
```

### Classes

- class **GraphDrawingAttributes**

## 10.63 Graph/include/GraphRep.h File Reference

```
#include "RefCounter.h"  
#include "set"  
#include "map"
```

### Classes

- struct **GraphEdge**  
*Defines non-labelled edge of the directed graph.*
- struct **GraphState**
- class **GraphRep**

## 10.64 Graph/include/GraphType.h File Reference

```
#include "set"
#include "vector"
#include "iostream"
#include "string"
#include "GraphConcept.h"
#include "GraphConceptAlgorithms.h"
#include "GraphDrawingAttributes.h"
#include "RanlibCPP.h"
```

### Classes

- struct **GraphEdge**  
*Defines non-labelled edge of the directed graph.*
- struct **GraphVertex< EdgeType >**  
***Graph** (p. 233) vertex.*
- struct **IntLabeledEdge**  
*Defines labelled (by integer) edge of the directed graph.*
- struct **PlanarGraphEdge**  
*Defines non-labelled edge of the directed planar graph.*
- struct **PlanarGraphIntLabelledEdge**  
*Defines non-labelled edge of the directed planar graph.*

### Typedefs

- typedef **GraphConcept< GraphVertex< GraphEdge >, GraphEdge > Graph**  
*Directed **Graph** (p. 233).*
- typedef **GraphConcept< GraphVertex< IntLabeledEdge >, IntLabeledEdge > IntLabeledGraph**  
*Directed **Graph** (p. 233) with integer labelled edges.*

## Functions

- ostream & **operator**<< (ostream &os, const **GraphEdge** &e)
- ostream & **operator**<< (ostream &os, const **IntLabeledEdge** &e)  
*Output operator for **IntLabeledEdge** (p. 277).*
- void **prepareToFold** (const **IntLabeledEdge** &e1, const **IntLabeledEdge** &e2)  
*Function preparing two edges to a fold (for **IntLabeledEdge** (p. 277) does nothing).*
- template<class ConstIntIterator >  
int **addRay** (**IntLabeledGraph** &G, int v, ConstIntIterator B, ConstIntIterator E, bool direct=false)  
*Function attaches a sequence of edges to a graph consequently labelled by numbers [B,E).*
- template<class ConstIntIterator >  
void **addLoop** (**IntLabeledGraph** &G, int v, ConstIntIterator B, ConstIntIterator E, bool direct=false)  
*Function attaches a loop of edges to a graph consequently labelled by numbers [B,E).*
- **Graph** **randomGraph** (int N, float edge\_param)  
*Create random graph on N vertices where each edge has probability to appear = edge\_param.*
- vector< vector< int > > **lengthTable** (const **Graph** &G)  
*Compute distances between all pairs of vertices.*
- vector< vector< int > > **innerProductTable** (const **Graph** &G, int origin)  
*Compute Gromov's inner product for all pairs of vertices (the origin is the vertex with the smallest number).*
- float **getHyperbolicityConst** (const **Graph** &G)  
*Compute constant of hyperbolicity of a graph (uses Gromov's inner product).*
- string **graphviz\_format** (const **Graph** &G, const **GraphDrawingAttributes** &GDA=**GraphDrawingAttributes**())  
*Get Graphviz graph description.*
- string **graphviz\_format** (const **IntLabeledGraph** &G, const **GraphDrawingAttributes** &GDA=**GraphDrawingAttributes**())  
*Get Graphviz graph description.*

### 10.64.1 Typedef Documentation

#### 10.64.1.1 `typedef GraphConcept< GraphVertex< GraphEdge > , GraphEdge > Graph`

Directed **Graph** (p. 233).

Definition at line 241 of file GraphType.h.

#### 10.64.1.2 `typedef GraphConcept< GraphVertex< IntLabeledEdge > , IntLabeledEdge > IntLabeledGraph`

Directed **Graph** (p. 233) with integer labelled edges.

Definition at line 246 of file GraphType.h.

### 10.64.2 Function Documentation

#### 10.64.2.1 `template<class ConstIntIterator > void addLoop (IntLabeledGraph & G, int v, ConstIntIterator B, ConstIntIterator E, bool direct = false) [inline]`

Function attaches a loop of edges to a graph consequently labelled by numbers [B,E).

Definition at line 275 of file GraphType.h.

References `Graphs::GraphConcept< VertexType, EdgeType >::newEdge()`, and `Graphs::GraphConcept< VertexType, EdgeType >::newVertex()`.

#### 10.64.2.2 `template<class ConstIntIterator > int addRay (IntLabeledGraph & G, int v, ConstIntIterator B, ConstIntIterator E, bool direct = false) [inline]`

Function attaches a sequence of edges to a graph consequently labelled by numbers [B,E).

Definition at line 258 of file GraphType.h.

References `Graphs::GraphConcept< VertexType, EdgeType >::newEdge()`, and `Graphs::GraphConcept< VertexType, EdgeType >::newVertex()`.

#### 10.64.2.3 `float getHyperbolicityConst (const Graph & G)`

Compute constant of hyperbolicity of a graph (uses Gromov's inner product).

**10.64.2.4** `string graphviz_format (const IntLabeledGraph & G, const GraphDrawingAttributes & GDA = GraphDrawingAttributes ())`

Get Graphviz graph description.

**10.64.2.5** `string graphviz_format (const Graph & G, const GraphDrawingAttributes & GDA = GraphDrawingAttributes ())`

Get Graphviz graph description.

**10.64.2.6** `vector< vector< int > > innerProductTable (const Graph & G, int origin)`

Compute Gromov's inner product for all pairs of vertices (the origin is the vertex with the smallest number).

**10.64.2.7** `vector< vector< int > > lengthTable (const Graph & G)`

Compute distances between all pairs of vertices.

**10.64.2.8** `ostream& operator<< (ostream & os, const IntLabeledEdge & e)`

Output operator for **IntLabeledEdge** (p. 277).

**10.64.2.9** `ostream& operator<< (ostream & os, const GraphEdge & e)`

**10.64.2.10** `void prepareToFold (const IntLabeledEdge & e1, const IntLabeledEdge & e2)`

Function preparing two edges to a fold (for **IntLabeledEdge** (p. 277) does nothing).

**10.64.2.11** `Graph randomGraph (int N, float edge_param)`

Create random graph on N vertices where each edge has probability to appear = edge\_param.

## 10.65 Graph/include/RandomFSA.h File Reference

```
#include "FSA.h"
```

### Functions

- **FSA randomFSA** (int N, int L)

*Generate a random (connected admissible) Finite State Automaton.*

### 10.65.1 Function Documentation

#### 10.65.1.1 FSA randomFSA (int N, int L)

Generate a random (connected admissible) Finite State Automaton.



## 10.66 Graphics/include/AImage.h File Reference

```
#include <iostream>
```

```
#include <string>
```

### Classes

- class **LookUpTable**
- class **AImage**
- class **LUTImage**
- class **GRImage**
- class **GRLUTImage**
- class **CImage**
- class **CLUTImage**

### Enumerations

- enum **IMAGE\_TYPE** { **GRAY**, **COLOR** }
- enum **FILE\_TYPE** {  
    **BW**, **PGM**, **PPM**, **JPG**,  
    **BMP**, **FT\_NA** }

### Functions

- **FILE\_TYPE** **getFileType** (const string &in\_file\_name)
- void **convert** (const **CImage** \*ci, **GRImage** \*gi)

### Variables

- const int **MAXGRAY** = 256

## 10.66.1 Enumeration Type Documentation

### 10.66.1.1 enum FILE\_TYPE

#### Enumerator:

***BW***

***PGM***

***PPM***

*JPG*  
*BMP*  
*FT\_NA*

Definition at line 24 of file AImage.h.

#### 10.66.1.2 enum IMAGE\_TYPE

Enumerator:

*GRAY*  
*COLOR*

Definition at line 23 of file AImage.h.

### 10.66.2 Function Documentation

10.66.2.1 void convert (const CImage \* *ci*, GRImage \* *gi*)

10.66.2.2 FILE\_TYPE getFileType (const string & *in\_file\_name*)

### 10.66.3 Variable Documentation

10.66.3.1 const int MAXGRAY = 256

Definition at line 20 of file AImage.h.

## 10.67 Graphics/include/PDFgraphing.h File Reference

```
#include "vector"  
#include "iostream"
```

### Classes

- class **PDFPageObject**  
*Implements interface for PDF drawing objects.*
- class **udPDFPageObjectText**  
*Class for pdf text object.*
- class **PDFPageObjectLine**  
*Class for pdf line object.*
- class **udPDFPageObjectLine**  
*Class implements pdf line object.*
- class **udPDFPageObjectVertLine**  
*Class implements a vertical line pdf object.*
- class **udPDFPageObjectHorizLine**  
*Class implements a horizontal line pdf object.*
- class **udPDFPageObjectRect**  
*Class implements a rectangle pdf object.*
- class **udPDFPageObjectSquare**  
*Class implements a square pdf object.*
- class **udPDFPageObjectCircle**  
*Class implements a square pdf object.*
- class **PDFPage**  
*Class implements a page of a pdf document.*
- class **PDFStructure**  
*Implements a pdf document consisting of several pages.*

## 10.68 Graphics/include/WordDraw.h File Reference

```
#include "AImage.h"
#include "PDFgraphing.h"
#include "Word.h"
#include <fstream>
#include <string>
```

### Classes

- class **WordDraw**  
*CREATES A PPM image of a table for braid word.*
- class **RGB**
- class **BraidDrawPDF**  
*Class for drawing a braid on  $n$  strands as a square table.*

### Variables

- const int **ssConst** = 20

#### 10.68.1 Variable Documentation

##### 10.68.1.1 const int ssConst = 20

Definition at line 22 of file WordDraw.h.

Referenced by WordDraw::WordDraw().

## 10.69 Group/include/AdvDehnAlgorithm.h File Reference

```
#include "FPGroup.h"  
#include "GraphType.h"  
#include "GraphConcept.h"  
#include "GraphConceptAlgorithms.h"
```

### Classes

- class **AdvDehnAlgorithm**

## 10.70 Group/include/FPGroup.h File Reference

```
#include <iostream>
#include <vector>
#include <string>
#include <sstream>
#include "Word.h"
#include "Alphabet.h"
```

### Classes

- class **FPGroup**

*Class **FPGroup** (p. 204) (finitely presented group).*

## 10.71 HigmanGroup/include/BaumslagGersten.h File Reference

### Classes

- struct **BG::BGMonomial**

### Namespaces

- namespace **BG**

### Functions

- bool **BG::solveWPinBG** (PowerCircuit \*pc, std::list< BGMonomial > &input)
- bool **BG::solveWPinBG** (PowerCircuit \*pc, std::string input)

## 10.72 HigmanGroup/include/PowerCircuit.h File Reference

### Classes

- class **PC::Node**
- class **PC::Marking**
- class **PC::PowerCircuit**

### Namespaces

- namespace **PC**



## 10.73 HigmanGroup/include/PowerCircuitCompMatrix.h File Reference

### Classes

- class **PC::PowerCircuitCompMatrix**
- struct **PC::PowerCircuitCompMatrix::RowHdr**
- struct **PC::PowerCircuitCompMatrix::ColHdr**
- struct **PC::PowerCircuitCompMatrix::IntNode**
- struct **PC::PowerCircuitCompMatrix::IntMarking**

### Namespaces

- namespace **PC**

## 10.74 HigmanGroup/include/PowerCircuitGraph.h

### File Reference

#### Classes

- class **PC::PowerCircuitGraph**
- struct **PC::PowerCircuitGraph::IntNode**
- struct **PC::PowerCircuitGraph::IntMarking**
- struct **PC::PowerCircuitGraph::NodeUsedByType**

#### Namespaces

- namespace **PC**

## 10.75 HigmanGroup/include/Sign.h File Reference

### Namespaces

- namespace **PC**

### Typedefs

- typedef char **PC::Sign**

### Functions

- void **PC::addSigns** (const Sign &op1, const Sign &op2, Sign &result, Sign &carry)
- int **PC::compareSigns** (const Sign &op1, const Sign &op2)
- int **PC::signToInt** (const Sign &s)
- Sign **PC::negateSign** (const Sign &s)
- std::string **PC::signToString** (const Sign &s)

### Variables

- const Sign **PC::ZERO** = 0
- const Sign **PC::PLUS** = 1
- const Sign **PC::MINUS** = 2

## 10.76 HigmanGroup/include/SignMatrix.h File Reference

### Classes

- struct **PC::Row**
- struct **PC::Col**
- class **PC::SignMatrix**< **RowHdr**, **ColHdr**, **INTERNAL\_TYPE** >
- struct **PC::SignMatrix**< **RowHdr**, **ColHdr**, **INTERNAL\_TYPE** >::**intRowHdr**
- struct **PC::SignMatrix**< **RowHdr**, **ColHdr**, **INTERNAL\_TYPE** >::**intColHdr**

### Namespaces

- namespace **PC**

### Variables

- const Row **PC::UNDEFINED\_ROW**
- const Col **PC::UNDEFINED\_COL**

## 10.77 Maps/include/Map.h File Reference

```
#include <iostream>
#include <vector>
#include "Word.h"
#include "errormsgs.h"
```

### Classes

- class **Map**  
*Defines a representation of a map between two sets of words.*

### Namespaces

- namespace **RMap**  
*Namespace for random map generating methods.*

### Functions

- **Map RMap::getRandomWhiteheadAuto** (int n)  
*Returns a randomly generated Whitehead automorphism.*
- **Map RMap::getRandomAuto** (int n, int l)  
*Returns a randomly generated automorphism of a given length.*

## 10.78 Maps/include/WhiteheadAutoSet.h File Reference

```
#include "Map.h"
#include "Word.h"
#include <ext/hash_set>
```

### Classes

- struct **compMaps**  
*Implements a comparison operator of two maps.*
- struct **\_\_gnu\_cxx::map\_hash**
- class **AutoSet**  
*Abstract interface for a set of Maps (Automorphisms).*
- class **NielsenAutoSet**  
*Implements a set of Nielsen automorphisms of a free group.*
- class **RestrictedWhiteheadAutoSet**  
*Implements a so-called restricted set of Whitehead automorphisms of a free group.*
- class **WhiteheadAutoSetType2**  
*Implements the set of Whitehead automorphisms of type II.*
- class **WhiteheadMinimization**  
*Implements a greedy procedure of reducing a word to its minimal length.*

### Namespaces

- namespace **\_\_gnu\_cxx**
- namespace **WhiteheadAutoSet**

### Typedefs

- typedef **\_\_gnu\_cxx::hash\_set< Map, \_\_gnu\_cxx::map\_hash, compMaps > SetOfMaps**  
*Implements a set fo **Map** (p. 320) objects.*

## Functions

- **Word** **WhiteheadAutoSet::reduceBy** (const **Word** &w, const **SetOfMaps** &theSet)

*Reduce the length of a given word by automorphisms form a set.*

### 10.78.1 Typedef Documentation

#### 10.78.1.1 `typedef __gnu_cxx::hash_set<Map, __gnu_cxx::map_hash, compMaps> SetOfMaps`

Implements a set fo **Map** (p. 320) objects.

Definition at line 39 of file WhiteheadAutoSet.h.

## 10.79 ranlib/include/cdflib.h File Reference

### Functions

- double **aldiv** (double \*, double \*)
- double **alngam** (double \*)
- double **alnrel** (double \*)
- double **apser** (double \*, double \*, double \*, double \*)
- double **basym** (double \*, double \*, double \*, double \*)
- double **bcorr** (double \*, double \*)
- double **betaln** (double \*, double \*)
- double **bfrac** (double \*, double \*, double \*, double \*, double \*, double \*)
- void **bgrat** (double \*, double \*, double \*, double \*, double \*, double \*, int \*)
- double **bpser** (double \*, double \*, double \*, double \*)
- void **bratio** (double \*, double \*, double \*, double \*, double \*, double \*, int \*)
- double **brcmp1** (int \*, double \*, double \*, double \*, double \*)
- double **brcomp** (double \*, double \*, double \*, double \*)
- double **bup** (double \*, double \*, double \*, double \*, int \*, double \*)
- void **cdfbet** (int \*, double \*, double \*, double \*, double \*, double \*, double \*, double \*, int \*, double \*)
- void **cdfbin** (int \*, double \*, double \*, double \*, double \*, double \*, double \*, double \*, int \*, double \*)
- void **cdfchi** (int \*, double \*, double \*, double \*, double \*, int \*, double \*)
- void **cdfchn** (int \*, double \*, double \*, double \*, double \*, double \*, double \*, int \*, double \*)
- void **cdff** (int \*, double \*, double \*, double \*, double \*, double \*, double \*, int \*, double \*)
- void **cdffnc** (int \*, double \*, double \*, double \*, double \*, double \*, double \*, double \*, int \*, double \*)
- void **cdfgam** (int \*, double \*, double \*, double \*, double \*, double \*, double \*, int \*, double \*)
- void **cdfnbn** (int \*, double \*, double \*, double \*, double \*, double \*, double \*, double \*, int \*, double \*)
- void **cdfnor** (int \*, double \*, double \*, double \*, double \*, double \*, double \*, int \*, double \*)
- void **cdfpoi** (int \*, double \*, double \*, double \*, double \*, int \*, double \*)
- void **cdfst** (int \*, double \*, double \*, double \*, double \*, int \*, double \*)
- void **cumbet** (double \*, double \*, double \*, double \*, double \*, double \*)
- void **cumbin** (double \*, double \*, double \*, double \*, double \*, double \*)
- void **cumchi** (double \*, double \*, double \*, double \*)
- void **cumchn** (double \*, double \*, double \*, double \*, double \*)
- void **cumf** (double \*, double \*, double \*, double \*, double \*)
- void **cumfnc** (double \*, double \*, double \*, double \*, double \*, double \*)



- void **cumgam** (double \*, double \*, double \*, double \*)
- void **cumnbn** (double \*, double \*, double \*, double \*, double \*, double \*)
- void **cumnor** (double \*, double \*, double \*)
- void **cumpoi** (double \*, double \*, double \*, double \*)
- void **cumt** (double \*, double \*, double \*, double \*)
- double **dbetrm** (double \*, double \*)
- double **devlpl** (double[], int \*, double \*)
- double **dexpm1** (double \*)
- double **dinvnr** (double \*p, double \*q)
- static void **E0000** (int, int \*, double \*, double \*, unsigned long \*, unsigned long \*, double \*, double \*, double \*, double \*, double \*, double \*, double \*)
- void **dinvr** (int \*, double \*, double \*, unsigned long \*, unsigned long \*)
- void **dstinv** (double \*, double \*, double \*, double \*, double \*, double \*, double \*, double \*)
- double **dlanor** (double \*)
- double **dln1mx** (double \*)
- double **dln1px** (double \*)
- double **dlnbet** (double \*, double \*)
- double **dlngam** (double \*)
- double **dstrem** (double \*)
- double **dt1** (double \*, double \*, double \*)
- static void **E0001** (int, int \*, double \*, double \*, double \*, double \*, unsigned long \*, unsigned long \*, double \*, double \*, double \*, double \*)
- void **dzror** (int \*, double \*, double \*, double \*, double \*, unsigned long \*, unsigned long \*)
- void **dstzr** (double \*zxlo, double \*zxhi, double \*zabstl, double \*zreltl)
- double **erf1** (double \*)
- double **erfc1** (int \*, double \*)
- double **esum** (int \*, double \*)
- double **exparg** (int \*)
- double **fpser** (double \*, double \*, double \*, double \*)
- double **gam1** (double \*)
- void **gaminv** (double \*, double \*, double \*, double \*, double \*, int \*)
- double **gamln** (double \*)
- double **gamln1** (double \*)
- double **Xgamm** (double \*)
- void **grat1** (double \*, double \*, double \*, double \*, double \*, double \*)
- void **gratio** (double \*, double \*, double \*, double \*, int \*)
- double **gsu1n** (double \*, double \*)
- double **psi** (double \*)
- double **rcomp** (double \*, double \*)
- double **rexp** (double \*)
- double **rlog** (double \*)

- double **rlog1** (double \*)
- double **smpar** (int \*)
- double **stvaln** (double \*)
- double **fifdint** (double)
- double **fifdmax1** (double, double)
- double **fifdmin1** (double, double)
- double **fifdsign** (double, double)
- long **fifdint** (double)
- long **fifmod** (long, long)
- void **ftnstop** (char \*)
- int **ipmpar** (int \*)

## 10.79.1 Function Documentation

**10.79.1.1** `double algdiv (double *, double *)`

**10.79.1.2** `double alngam (double *)`

**10.79.1.3** `double alnrel (double *)`

**10.79.1.4** `double apser (double *, double *, double *, double *)`

**10.79.1.5** `double basym (double *, double *, double *, double *)`

**10.79.1.6** `double bcorr (double *, double *)`

**10.79.1.7** `double betaln (double *, double *)`

**10.79.1.8** `double bfrac (double *, double *, double *, double *, double *, double *)`

**10.79.1.9** `void bgrat (double *, double *, double *, double *, double *, double *, int * i)`

**10.79.1.10** `double bpser (double *, double *, double *, double *)`

**10.79.1.11** `void bratio (double *, double *, double *, double *, double *, double *, int *)`

**10.79.1.12** `double brcmp1 (int *, double *, double *, double *, double *)`

**10.79.1.13** `double brcomp (double *, double *, double *, double *)`

**10.79.1.14** `double bup (double *, double *, double *, double *, int *, double *)`

**10.79.1.15** `void cdfbet (int *, double *, double *, double *, double *, double *, double *, int *, double *)`

**10.79.1.16** `void cdfbin (int *, double *, double *, double *, double *, double *, double *, int *, double *)`

**10.79.1.17** `void cdfchi (int *, double *, double *, double *, double *, int *, double *)`

Referenced by RandLib::chiPValue().

- 10.79.1.18 void cdfchn (int \*, double \*, double \*, double \*, double \*, double \*, int \*, double \*)
- 10.79.1.19 void cdff (int \*, double \*, double \*, double \*, double \*, double \*, int \*, double \*)
- 10.79.1.20 void cdffnc (int \*, double \*, double \*, double \*, double \*, double \*, double \*, int \*, double \*)
- 10.79.1.21 void cdfgam (int \*, double \*, double \*, double \*, double \*, double \*, int \*, double \*)
- 10.79.1.22 void cdfnbn (int \*, double \*, double \*, double \*, double \*, double \*, double \*, int \*, double \*)
- 10.79.1.23 void cdfnor (int \*, double \*, double \*, double \*, double \*, double \*, int \*, double \*)
- 10.79.1.24 void cdfpoi (int \*, double \*, double \*, double \*, double \*, int \*, double \*)
- 10.79.1.25 void cdft (int \*, double \*, double \*, double \*, double \*, int \*, double \*)
- 10.79.1.26 void cumbet (double \*, double \*, double \*, double \*, double \*, double \*)
- 10.79.1.27 void cumbin (double \*, double \*, double \*, double \*, double \*, double \*)
- 10.79.1.28 void cumchi (double \*, double \*, double \*, double \*)
- 10.79.1.29 void cumchn (double \*, double \*, double \*, double \*, double \*)
- 10.79.1.30 void cumf (double \*, double \*, double \*, double \*, double \*)
- 10.79.1.31 void cumfnc (double \*, double \*, double \*, double \*, double \*, double \*)
- 10.79.1.32 void cumgam (double \*, double \*, double \*, double \*)
- 10.79.1.33 void cumnbn (double \*, double \*, double \*, double \*, double \*, double \*)
- 10.79.1.34 void cumnor (double \*, double \*, double \*)
- 10.79.1.35 void cumpoi (double \*, double \*, double \*, double \*)
- 10.79.1.36 void cumt (double \*, double \*, double \*, double \*)
- 10.79.1.37 double dbetrm (double \*, double \*)
- 10.79.1.38 double devlpl (double[], int \*, double \*)
- 10.79.1.39 double dexpm1 (double \*)

## 10.80 ranlib/include/ranlib.h File Reference

### Functions

- void **advnst** (long k)
- float **genbet** (float aa, float bb)
- float **genchi** (float df)
- float **genexp** (float av)
- float **genf** (float dfn, float dfd)
- float **gengam** (float a, float r)
- void **genmn** (float \*parm, float \*x, float \*work)
- void **genmul** (long n, float \*p, long ncat, long \*ix)
- float **gennch** (float df, float xnonc)
- float **gennf** (float dfn, float dfd, float xnonc)
- float **gennor** (float av, float sd)
- void **genprm** (long \*iarray, int larray)
- float **genunf** (float low, float high)
- void **getsd** (long \*iseed1, long \*iseed2)
- void **gscgn** (long getset, long \*g)
- long **ignbin** (long n, float pp)
- long **ignbnb** (long n, float p)
- long **ignlgi** (void)
- long **ignpoi** (float mu)
- long **ignuin** (long low, long high)
- void **initgn** (long isdtyp)
- long **mltmod** (long a, long s, long m)
- void **phrtsd** (char \*phrase, long \*seed1, long \*seed2)
- float **ranf** (void)
- void **setall** (long iseed1, long iseed2)
- void **setant** (long qvalue)
- void **setgm** (float \*meanv, float \*covm, long p, float \*parm)
- void **setsd** (long iseed1, long iseed2)
- float **sexpo** (void)
- float **sgamma** (float a)
- float **snorm** (void)

## 10.80.1 Function Documentation

**10.80.1.1** void advnst (long *k*)

**10.80.1.2** float genbet (float *aa*, float *bb*)

**10.80.1.3** float genchi (float *df*)

**10.80.1.4** float genexp (float *av*)

**10.80.1.5** float genf (float *dfn*, float *dfd*)

**10.80.1.6** float gengam (float *a*, float *r*)

**10.80.1.7** void genmn (float \* *parm*, float \* *x*, float \* *work*)

**10.80.1.8** void genmul (long *n*, float \* *p*, long *ncat*, long \* *ix*)

**10.80.1.9** float gennch (float *df*, float *xnonc*)

**10.80.1.10** float gennf (float *dfn*, float *dfd*, float *xnonc*)

**10.80.1.11** float gennor (float *av*, float *sd*)

**10.80.1.12** void genprm (long \* *iarray*, int *larray*)

**10.80.1.13** float genunf (float *low*, float *high*)

Referenced by RandLibURG::rand().

**10.80.1.14** void getsd (long \* *iseed1*, long \* *iseed2*)

Referenced by RandLibURG::getseed().

**10.80.1.15** void gscgn (long *getset*, long \* *g*)

**10.80.1.16** long ignbin (long *n*, float *pp*)

**10.80.1.17** long ignlgi (void)

**10.80.1.18** long ignnbn (long *n*, float *p*)

**10.80.1.19** long ignpoi (float *mu*)

**10.80.1.20** long ignuin (long *low*, long *high*)

Referenced by RandLibURG::irand().

**10.80.1.21** void initgn (long *isdtyp*)

**10.80.1.22** long mltmod (long *a*, long *s*, long *m*)

**10.80.1.23** void phrtsd (char \* *phrase*, long \* *seed1*, long \* *seed2*)

**10.80.1.24** float ranf (void)

Referenced by RandLibURG::rand().

**10.80.1.25** void setall (long *iseed1*, long *iseed2*)

Referenced by RandLibURG::RandLibURG(), RandLibURG::reset(), and RandLibURG::setSeedPID().

**10.80.1.26** void setant (long *qvalue*)

**10.80.1.27** void setgm (float \* *meanv*, float \* *covm*, long *p*, float \* *parm*)

**10.80.1.28** void setsd (long *iseed1*, long *iseed2*)

**10.80.1.29** float sexpo (void)

**10.80.1.30** float sgamma (float *a*)

**10.80.1.31** float snorm (void)

## 10.81 ranlib/include/RanlibCPP.h File Reference

```
#include "ranlib.h"
#include "cdflib.h"
#include <time.h>
#include <unistd.h>
#include <stdlib.h>
#include <iostream>
```

### Classes

- class **RandLibURG**  
*A wrapper class for RANLIB Library.*
- class **RandLib**  
*Static wrapper class for RANDLIB Library.*



## 10.82 SbgpFG/include/SubgroupFG.h File Reference

```
#include "Word.h"
#include "GraphType.h"
#include "GraphConcept.h"
#include "GraphConceptAlgorithms.h"
#include "vector"
```

### Classes

- class **SubgroupFG**

### Functions

- template<class ConstIterator >  
  **Word substitute** (ConstIterator B, ConstIterator E, const vector< **Word** > &wrds)
- **Word substitute** (const **Word** &w, const vector< **Word** > &wrds)
- ostream & **operator**<< (ostream &os, const **SubgroupFG** &sbgp)

### 10.82.1 Function Documentation

**10.82.1.1** ostream& **operator**<< (ostream & *os*, const SubgroupFG & *sbgp*)

**10.82.1.2** Word substitute (const Word & *w*, const vector< Word > & *wrds*)

**10.82.1.3** template<class ConstIterator > Word substitute (ConstIterator *B*, ConstIterator *E*, const vector< Word > & *wrds*) [**inline**]

Definition at line 206 of file SubgroupFG.h.

## 10.83 StringSimilarity/include/Levenstein.h File Reference

```
#include "Word.h"
```

### Classes

- class **Levenstein**

*A class for computing the **Levenstein** (p. 303) distance between two words.*

## 10.84 StringSimilarity/include/MotivePatternWrapper.h File Reference

```
#include "global.h"  
#include "FreeGroup.h"  
#include "Word.h"  
#include <vector>  
#include <stdlib.h>
```

### Classes

- struct **MotivePattern**
- class **MotivePatternWrapper**

## 10.85 StringSimilarity/include/PairDistanceTest.h File Reference

```
#include "Word.h"
#include "RanlibCPP.h"
#include "errormsgs.h"
#include "SimilarityMeasures.h"
#include <sstream>
#include <vector>
#include <algorithm>
#include <sstream>
```

### Classes

- class **PairGenerator**  
*Abstract interface for classes defining sword pairs similarities.*
- class **RandomPairGenerator**  
*Implements a class for generating pseudo-random pairs of words.*
- class **PairDistanceSimilarityTest**  
*Implements a hypothesis testing that two words a similar according to some given criteria.*

## 10.86 StringSimilarity/include/SimilarityMeasures.h File Reference

```
#include "Word.h"  
#include <vector>
```

### Classes

- class **WordPairComparison**  
*Implements a probabilistic measure for comparing two words.*
- class **StringSimilarityMeasure**  
*Abstract interface for a class implementing string (**Word** (p. 642)) similarity measure.*
- class **HammingDistance**  
*Implements the Hamming distance between two words.*
- class **HammingDistanceCyclic**  
*Implements the Hamming distance between two cyclic words.*
- class **SubwordHammingDistanceCyclic**  
*Implements the Hamming distance between two cyclic words.*
- class **EditingDistance**  
*Implements the Editing (**Levenstein** (p. 303)) distance between two words.*
- class **SubwordEditingDistanceCyclic**  
*Implements the Editing distance between two cyclic words.*

### Functions

- double **getLeftTaleConfidenceValue** (const vector< double > &dist, double p)  
*Returns the left tale confidence value.*

## 10.86.1 Function Documentation

### 10.86.1.1 `double getLeftTaleConfidenceValue (const vector< double > & dist, double p)`

Returns the left tale confidence value. Given an estimate (as a list of sorted samples) of a probability distributions, returns the confidence value for a given `p-value`.

**Parameters:**

*dist* - distribution estimate.

*p* - the p -value.

**Returns:**

the confidence value (probability).

## 10.87 StringSimilarity/include/StringScramblers.h File Reference

```
#include "Word.h"  
#include <vector>
```

### Classes

- class **StringScrambler**
- class **UniformScrambler**
- class **SubwordScrambler**
- class **WordMultiplyScrambler**

## 10.88 TheGrigorchukGroup/include/TheGrigorchukGroupAlgorithms.h

### File Reference

```
#include "map"
#include "list"
#include "set"
#include "tuples.h"
#include "Word.h"
```

### Classes

- class **TheGrigorchukGroupAlgorithms**

*Static class **TheGrigorchukGroupAlgorithms** (p.535) encapsulates algorithms for the original Grigorchuk group.*



## **10.89 ThompsonGroup/include/ThompsonGroupFNormalForm.h File Reference**

```
#include "Word.h"
```

### **Classes**

- class **ThompsonGroupFNormalForm**

*Class **ThompsonGroupFNormalForm** (p. 556) (defines a representation of a normal form of an element of the Thompson's group  $F$  (infinitely generated))//.*