symbolic_operators

2.2.0

Generated by Doxygen 1.9.1

1 symbolic_operators Library Documentation 1
1.1 Introduction . . . 1
1.2 Example Usage . . . 1
1.3 Class Explanations . . . 3

2 Namespace Index 5
2.1 Namespace List . . . 5

3 Hierarchical Index 7
3.1 Class Hierarchy . . . 7

4 Class Index 9
4.1 Class List . . . 9

5 File Index 11
5.1 File List . . . 11

6 Namespace Documentation 13
6.1 mrock Namespace Reference . . . 13
6.2 mrock::symbolic_operators Namespace Reference . . . 13
6.2.1 Typedef Documentation . . . 18
6.2.1.1 index_base . . . 19
6.2.1.2 IndexSum . . . 19
6.2.1.3 IntFractional . . . 19
6.2.1.4 momentum_symbols . . . 19
6.2.1.5 MomentumSum . . . 19
6.2.2 Enumeration Type Documentation . . . 19
6.2.2.1 Index . . . 20
6.2.2.2 OperatorType . . . 21
6.2.3 Function Documentation . . . 21
6.2.3.1 char_to_index() . . . 21
6.2.3.2 clean_up() . . . 21
6.2.3.3 clean_wicks() . . . 22
6.2.3.4 clear_duplicates() . . . 22
6.2.3.5 clear_etas() . . . 22
6.2.3.6 commutator() [1/4] . . . 23
6.2.3.7 commutator() [2/4] . . . 23
6.2.3.8 commutator() [3/4] . . . 24
6.2.3.9 commutator() [4/4] . . . 24
6.2.3.10 hermitian_conjugate() . . . 24
6.2.3.11 identify_subexpression() . . . 25
6.2.3.12 identify_wick_operators() . . . 25
6.2.3.13 is_always_zero() [1/2] . . . 25
6.2.3.14 is_always_zero() [2/2] . . . 26

6.2.3.15 is_mutable() . . . 26
6.2.3.16 make_delta() [1/2] . . . 27
6.2.3.17 make_delta() [2/2] . . . 27
6.2.3.18 momentum_order() . . . 28
6.2.3.19 normal_order() . . . 28
6.2.3.20 operator"!=() [1/7] . . . 28
6.2.3.21 operator"!=() [2/7] . . . 29
6.2.3.22 operator"!=() [3/7] . . . 29
6.2.3.23 operator"!=() [4/7] . . . 30
6.2.3.24 operator"!=() [5/7] . . . 30
6.2.3.25 operator"!=() [6/7] . . . 31
6.2.3.26 operator"!=() [7/7] . . . 31
6.2.3.27 operator∗() [1/2] . . . 31
6.2.3.28 operator∗() [2/2] . . . 32
6.2.3.29 operator+() [1/7] . . . 32
6.2.3.30 operator+() [2/7] . . . 33
6.2.3.31 operator+() [3/7] . . . 33
6.2.3.32 operator+() [4/7] . . . 34
6.2.3.33 operator+() [5/7] . . . 34
6.2.3.34 operator+() [6/7] . . . 35
6.2.3.35 operator+() [7/7] . . . 35
6.2.3.36 operator+=() [1/3] . . . 35
6.2.3.37 operator+=() [2/3] . . . 36
6.2.3.38 operator+=() [3/3] . . . 36
6.2.3.39 operator-() [1/6] . . . 37
6.2.3.40 operator-() [2/6] . . . 37
6.2.3.41 operator-() [3/6] . . . 38
6.2.3.42 operator-() [4/6] . . . 38
6.2.3.43 operator-() [5/6] . . . 39
6.2.3.44 operator-() [6/6] . . . 39
6.2.3.45 operator-=() [1/3] . . . 39
6.2.3.46 operator-=() [2/3] . . . 40
6.2.3.47 operator-=() [3/3] . . . 40
6.2.3.48 operator<() . . . 41
6.2.3.49 operator<<() [1/19] . . . 41
6.2.3.50 operator<<() [2/19] . . . 42
6.2.3.51 operator<<() [3/19] . . . 42
6.2.3.52 operator<<() [4/19] . . . 42
6.2.3.53 operator<<() [5/19] . . . 43
6.2.3.54 operator<<() [6/19] . . . 43
6.2.3.55 operator<<() [7/19] . . . 44
6.2.3.56 operator<<() [8/19] . . . 44

6.2.3.57 operator<<() [9/19] . . . 45
6.2.3.58 operator<<() [10/19] . . . 45
6.2.3.59 operator<<() [11/19] . . . 45
6.2.3.60 operator<<() [12/19] . . . 46
6.2.3.61 operator<<() [13/19] . . . 46
6.2.3.62 operator<<() [14/19] . . . 47
6.2.3.63 operator<<() [15/19] . . . 47
6.2.3.64 operator<<() [16/19] . . . 48
6.2.3.65 operator<<() [17/19] . . . 48
6.2.3.66 operator<<() [18/19] . . . 48
6.2.3.67 operator<<() [19/19] . . . 49
6.2.3.68 operator==() [1/7] . . . 49
6.2.3.69 operator==() [2/7] . . . 50
6.2.3.70 operator==() [3/7] . . . 50
6.2.3.71 operator==() [4/7] . . . 51
6.2.3.72 operator==() [5/7] . . . 51
6.2.3.73 operator==() [6/7] . . . 52
6.2.3.74 operator==() [7/7] . . . 52
6.2.3.75 operator>() . . . 53
6.2.3.76 operator>>() . . . 53
6.2.3.77 prepare_wick() . . . 53
6.2.3.78 remove_delta_is_one() . . . 54
6.2.3.79 remove_delta_squared() . . . 54
6.2.3.80 remove_double_occurances() . . . 54
6.2.3.81 rename_momenta() . . . 55
6.2.3.82 to_string_without_prefactor() . . . 55
6.2.3.83 wick_processor() . . . 56
6.2.3.84 wicks_theorem() . . . 56
6.2.4 Variable Documentation . . . 56
6.2.4.1 string_to_index . . . 56
6.2.4.2 string_to_wick . . . 56
6.3 sym_op_test Namespace Reference . . . 57

7 Class Documentation **59**

7.1 mrock::symbolic_operators::bad_term_exception Class Reference . . . 59
7.1.1 Detailed Description . . . 60
7.1.2 Constructor & Destructor Documentation . . . 60
7.1.2.1 bad_term_exception() [1/2] . . . 60
7.1.2.2 bad_term_exception() [2/2] . . . 60
7.1.3 Member Function Documentation . . . 61
7.1.3.1 which_term() . . . 61
7.1.4 Member Data Documentation . . . 61

7.1.4.1 _term . . . 61
7.2 mrock::symbolic_operators::Coefficient Struct Reference . . . 62
7.2.1 Detailed Description . . . 64
7.2.2 Constructor & Destructor Documentation . . . 64
7.2.2.1 Coefficient() [1/5] . . . 64
7.2.2.2 Coefficient() [2/5] . . . 64
7.2.2.3 Coefficient() [3/5] . . . 64
7.2.2.4 Coefficient() [4/5] . . . 65
7.2.2.5 Coefficient() [5/5] . . . 65
7.2.3 Member Function Documentation . . . 66
7.2.3.1 apply_custom_symmetry() . . . 66
7.2.3.2 Constant() . . . 66
7.2.3.3 depends_on() . . . 67
7.2.3.4 depends_on_momentum() . . . 67
7.2.3.5 depends_on_two_momenta() . . . 67
7.2.3.6 hermitian_conjugate() . . . 68
7.2.3.7 hermitian_conjugate_inplace() . . . 68
7.2.3.8 HoneyComb() . . . 68
7.2.3.9 invert_momentum() . . . 69
7.2.3.10 parse_interaction_string() . . . 69
7.2.3.11 parse_string() . . . 69
7.2.3.12 RealInteraction() . . . 70
7.2.3.13 RealInversionSymmetric() . . . 70
7.2.3.14 remove_momentum_contribution() . . . 71
7.2.3.15 serialize() . . . 71
7.2.3.16 use_symmetric_interaction_exchange() . . . 72
7.2.3.17 use_symmetric_interaction_inversion() . . . 72
7.2.3.18 uses_index() . . . 72
7.2.4 Member Data Documentation . . . 72
7.2.4.1 custom_symmetry . . . 72
7.2.4.2 indizes . . . 73
7.2.4.3 inversion_symmetry . . . 73
7.2.4.4 is_daggered . . . 73
7.2.4.5 is_real . . . 73
7.2.4.6 is_symmetrized_interaction . . . 73
7.2.4.7 momenta . . . 74
7.2.4.8 name . . . 74
7.2.4.9 Q_changes_sign . . . 74
7.3 mrock::symbolic_operators::IndexComparison Struct Reference . . . 74
7.3.1 Detailed Description . . . 75
7.3.2 Member Data Documentation . . . 75
7.3.2.1 any_identical . . . 75

7.3.2.2 base . . . 75
7.3.2.3 other . . . 75
7.4 mrock::symbolic_operators::IndexWrapper Struct Reference . . . 75
7.4.1 Detailed Description . . . 76
7.4.2 Constructor & Destructor Documentation . . . 76
7.4.2.1 IndexWrapper() [1/4] . . . 76
7.4.2.2 IndexWrapper() [2/4] . . . 76
7.4.2.3 IndexWrapper() [3/4] . . . 77
7.4.2.4 IndexWrapper() [4/4] . . . 77
7.4.3 Member Function Documentation . . . 77
7.4.3.1 operator<=>() . . . 77
7.4.3.2 serialize() . . . 78
7.4.3.3 VECTOR_WRAPPER_FILL_MEMBERS() . . . 78
7.4.4 Member Data Documentation . . . 78
7.4.4.1 indizes . . . 78
7.5 mrock::symbolic_operators::InversionSymmetry Class Reference . . . 79
7.5.1 Detailed Description . . . 79
7.5.2 Member Function Documentation . . . 80
7.5.2.1 apply_to() . . . 80
7.6 mrock::symbolic_operators::KroneckerDelta< T > Class Template Reference . . . 80
7.6.1 Detailed Description . . . 80
7.6.2 Member Function Documentation . . . 81
7.6.2.1 isOne() . . . 81
7.6.2.2 serialize() . . . 81
7.6.3 Member Data Documentation . . . 82
7.6.3.1 first . . . 82
7.6.3.2 second . . . 82
7.7 mrock::symbolic_operators::Momentum Struct Reference . . . 82
7.7.1 Detailed Description . . . 84
7.7.2 Constructor & Destructor Documentation . . . 84
7.7.2.1 Momentum() [1/7] . . . 84
7.7.2.2 Momentum() [2/7] . . . 84
7.7.2.3 Momentum() [3/7] . . . 85
7.7.2.4 Momentum() [4/7] . . . 85
7.7.2.5 Momentum() [5/7] . . . 85
7.7.2.6 Momentum() [6/7] . . . 86
7.7.2.7 Momentum() [7/7] . . . 86
7.7.3 Member Function Documentation . . . 86
7.7.3.1 add_in_place() . . . 86
7.7.3.2 differs_only_in_Q() . . . 87
7.7.3.3 first_momentum_is() . . . 87
7.7.3.4 first_momentum_is_negative() . . . 88

7.7.3.5 flip_momentum() . . . 88
7.7.3.6 flip_single() . . . 88
7.7.3.7 is_used_at() . . . 88
7.7.3.8 is_zero() . . . 89
7.7.3.9 last_momentum_is() . . . 89
7.7.3.10 last_momentum_is_negative() . . . 89
7.7.3.11 multiply_by() . . . 90
7.7.3.12 operator"!=() . . . 90
7.7.3.13 operator*=() . . . 90
7.7.3.14 operator+=() . . . 91
7.7.3.15 operator-=() . . . 91
7.7.3.16 operator==() . . . 91
7.7.3.17 remove_contribution() . . . 92
7.7.3.18 remove_zeros() . . . 92
7.7.3.19 replace_occurances() . . . 92
7.7.3.20 serialize() . . . 93
7.7.3.21 sort() . . . 93
7.7.3.22 to_string() . . . 93
7.7.3.23 uses() . . . 93
7.7.3.24 VECTOR_WRAPPER_FILL_MEMBERS() . . . 94
7.7.4 Member Data Documentation . . . 94
7.7.4.1 add_Q . . . 94
7.7.4.2 momentum_list . . . 94
7.8 mrock::symbolic_operators::MomentumList Class Reference . . . 95
7.8.1 Detailed Description . . . 96
7.8.2 Member Typedef Documentation . . . 96
7.8.2.1 _parent . . . 96
7.8.3 Constructor & Destructor Documentation . . . 96
7.8.3.1 MomentumList() [1/5] . . . 97
7.8.3.2 MomentumList() [2/5] . . . 97
7.8.3.3 MomentumList() [3/5] . . . 97
7.8.3.4 MomentumList() [4/5] . . . 97
7.8.3.5 MomentumList() [5/5] . . . 98
7.8.4 Member Function Documentation . . . 98
7.8.4.1 flip_momentum() . . . 98
7.8.4.2 flip_single() . . . 98
7.8.4.3 multiply_by() . . . 99
7.8.4.4 operator*=() . . . 99
7.8.4.5 remove_zeros() . . . 99
7.8.4.6 replace_occurances() . . . 100
7.8.4.7 serialize() . . . 101
7.8.4.8 sort() . . . 101

7.9 mrock::symbolic_operators::MomentumSymbol Struct Reference . . . 102
7.9.1 Detailed Description . . . 103
7.9.2 Constructor & Destructor Documentation . . . 103
7.9.2.1 MomentumSymbol() [1/3] . . . 103
7.9.2.2 MomentumSymbol() [2/3] . . . 103
7.9.2.3 MomentumSymbol() [3/3] . . . 103
7.9.3 Member Function Documentation . . . 104
7.9.3.1 operator<=>() . . . 104
7.9.3.2 serialize() . . . 104
7.9.4 Member Data Documentation . . . 104
7.9.4.1 factor . . . 105
7.9.4.2 name . . . 105
7.10 mrock::symbolic_operators::MomentumSymbol::name_type Struct Reference . . . 105
7.10.1 Detailed Description . . . 106
7.10.2 Constructor & Destructor Documentation . . . 106
7.10.2.1 name_type() [1/2] . . . 106
7.10.2.2 name_type() [2/2] . . . 106
7.10.3 Member Function Documentation . . . 106
7.10.3.1 operator char() . . . 106
7.10.3.2 operator<=>() [1/2] . . . 107
7.10.3.3 operator<=>() [2/2] . . . 107
7.10.3.4 serialize() . . . 107
7.10.4 Member Data Documentation . . . 108
7.10.4.1 _n . . . 108
7.11 mrock::symbolic_operators::Operator Struct Reference . . . 108
7.11.1 Detailed Description . . . 110
7.11.2 Constructor & Destructor Documentation . . . 110
7.11.2.1 Operator() [1/5] . . . 110
7.11.2.2 Operator() [2/5] . . . 110
7.11.2.3 Operator() [3/5] . . . 111
7.11.2.4 Operator() [4/5] . . . 111
7.11.2.5 Operator() [5/5] . . . 112
7.11.3 Member Function Documentation . . . 112
7.11.3.1 add_momentum() [1/2] . . . 112
7.11.3.2 add_momentum() [2/2] . . . 113
7.11.3.3 Boson() [1/2] . . . 113
7.11.3.4 Boson() [2/2] . . . 113
7.11.3.5 first_index() . . . 114
7.11.3.6 hermitian_conjugate() . . . 114
7.11.3.7 hermitian_conjugate_inplace() . . . 114
7.11.3.8 remove_momentum_contribution() . . . 114
7.11.3.9 serialize() . . . 115

7.11.3.10 set_first_index() . . . 115
7.11.3.11 with_momentum() [1/2] . . . 115
7.11.3.12 with_momentum() [2/2] . . . 116
7.11.4 Member Data Documentation . . . 116
7.11.4.1 indizes . . . 116
7.11.4.2 is_daggered . . . 117
7.11.4.3 is_fermion . . . 117
7.11.4.4 momentum . . . 117
7.12 mrock::symbolic_operators::PhaseSymmetry< operators> Class Template Reference . . . 117
7.12.1 Detailed Description . . . 118
7.12.2 Member Function Documentation . . . 119
7.12.2.1 apply_to() . . . 119
7.13 mrock::symbolic_operators::TemplateResult::SingleResult Struct Reference . . . 119
7.13.1 Detailed Description . . . 120
7.13.2 Member Function Documentation . . . 120
7.13.2.1 clear_delta_equals_one() . . . 121
7.13.2.2 contains_impossible_delta() . . . 121
7.13.3 Member Data Documentation . . . 121
7.13.3.1 factor . . . 121
7.13.3.2 index_deltas . . . 121
7.13.3.3 op . . . 122
7.14 mrock::symbolic_operators::SpinSymmetry Class Reference . . . 122
7.14.1 Detailed Description . . . 123
7.14.2 Member Function Documentation . . . 123
7.14.2.1 apply_to() . . . 123
7.15 mrock::symbolic_operators::SumContainer Struct Reference . . . 123
7.15.1 Detailed Description . . . 124
7.15.2 Member Function Documentation . . . 125
7.15.2.1 append() [1/3] . . . 125
7.15.2.2 append() [2/3] . . . 125
7.15.2.3 append() [3/3] . . . 125
7.15.2.4 has_momentum() . . . 126
7.15.2.5 has_spins() . . . 126
7.15.2.6 push_back() [1/2] . . . 126
7.15.2.7 push_back() [2/2] . . . 127
7.15.2.8 serialize() . . . 127
7.15.3 Member Data Documentation . . . 127
7.15.3.1 momenta . . . 128
7.15.3.2 spins . . . 128
7.16 mrock::symbolic_operators::SymbolicSum< SumIndex > Struct Template Reference . . . 128
7.16.1 Detailed Description . . . 129
7.16.2 Constructor & Destructor Documentation . . . 129

7.16.2.1 SymbolicSum() [1/5] . . . 129
7.16.2.2 SymbolicSum() [2/5] . . . 129
7.16.2.3 SymbolicSum() [3/5] . . . 130
7.16.2.4 SymbolicSum() [4/5] . . . 130
7.16.2.5 SymbolicSum() [5/5] . . . 130
7.16.3 Member Function Documentation . . . 130
7.16.3.1 is_summed_over() . . . 131
7.16.3.2 operator<=>() . . . 131
7.16.3.3 serialize() . . . 131
7.16.3.4 VECTOR_WRAPPER_FILL_MEMBERS() . . . 132
7.16.4 Member Data Documentation . . . 132
7.16.4.1 summations . . . 132
7.17 sym_op_test::SymOpTest Struct Reference . . . 132
7.17.1 Detailed Description . . . 133
7.17.2 Constructor & Destructor Documentation . . . 133
7.17.2.1 SymOpTest() . . . 133
7.17.3 Member Function Documentation . . . 133
7.17.3.1 load_and_test() . . . 133
7.17.3.2 perform_comparison() . . . 134
7.17.3.3 perform_test() . . . 134
7.17.3.4 save_as_comparison() . . . 134
7.17.4 Member Data Documentation . . . 134
7.17.4.1 COMPARE_DIR . . . 134
7.18 mrock::symbolic_operators::TemplateResult Struct Reference . . . 135
7.18.1 Detailed Description . . . 136
7.18.2 Constructor & Destructor Documentation . . . 136
7.18.2.1 TemplateResult() [1/2] . . . 136
7.18.2.2 TemplateResult() [2/2] . . . 136
7.18.3 Member Function Documentation . . . 137
7.18.3.1 add_index_delta() . . . 137
7.18.3.2 add_index_delta_range() . . . 137
7.18.3.3 clean_up() . . . 137
7.18.3.4 clear_impossible() . . . 138
7.18.3.5 create_branch() . . . 138
7.18.3.6 null_result() . . . 138
7.18.3.7 operation_on_each() . . . 138
7.18.3.8 operation_on_range() . . . 139
7.18.3.9 operator bool() . . . 139
7.18.4 Member Data Documentation . . . 139
7.18.4.1 momentum_delta . . . 140
7.18.4.2 results . . . 140
7.19 mrock::symbolic_operators::Term Class Reference . . . 140

7.19.1 Detailed Description . . . 143
7.19.2 Constructor & Destructor Documentation . . . 143
7.19.2.1 Term() [1/10] . . . 143
7.19.2.2 Term() [2/10] . . . 144
7.19.2.3 Term() [3/10] . . . 144
7.19.2.4 Term() [4/10] . . . 145
7.19.2.5 Term() [5/10] . . . 145
7.19.2.6 Term() [6/10] . . . 145
7.19.2.7 Term() [7/10] . . . 147
7.19.2.8 Term() [8/10] . . . 147
7.19.2.9 Term() [9/10] . . . 148
7.19.2.10 Term() [10/10] . . . 148
7.19.3 Member Function Documentation . . . 148
7.19.3.1 compute_sums() . . . 148
7.19.3.2 contains_boson() . . . 148
7.19.3.3 contains_fermion() . . . 149
7.19.3.4 count_bosons() . . . 149
7.19.3.5 count_fermions() . . . 149
7.19.3.6 discard_zero_momenta() . . . 149
7.19.3.7 flip_sign() . . . 150
7.19.3.8 get_operators() . . . 150
7.19.3.9 hermitian_conjugate() . . . 150
7.19.3.10 hermitian_conjugate_inplace() . . . 150
7.19.3.11 invert_momentum() . . . 150
7.19.3.12 invert_momentum_sum() . . . 151
7.19.3.13 is_equal() . . . 151
7.19.3.14 is_identity() . . . 151
7.19.3.15 is_normal_ordered() . . . 152
7.19.3.16 perform_operator_swap() . . . 152
7.19.3.17 print() . . . 152
7.19.3.18 remove_momentum_contribution() . . . 153
7.19.3.19 rename_indizes() . . . 153
7.19.3.20 rename_momenta() . . . 153
7.19.3.21 rename_sums() . . . 154
7.19.3.22 serialize() . . . 154
7.19.3.23 set_deltas() . . . 154
7.19.3.24 sort() . . . 154
7.19.3.25 swap_momenta() . . . 155
7.19.3.26 to_string_without_prefactor() . . . 155
7.19.3.27 transform_momentum_sum() . . . 155
7.19.4 Friends And Related Function Documentation . . . 156
7.19.4.1 commutator . . . 156

7.19.4.2 normal_order . . . 156
7.19.4.3 operator<< . . . 156
7.19.4.4 WickTerm . . . 157
7.19.5 Member Data Documentation . . . 157
7.19.5.1 _TERM_TRACKER_ATTRIBUTE . . . 157
7.19.5.2 coefficients . . . 157
7.19.5.3 delta_indizes . . . 158
7.19.5.4 delta_momenta . . . 158
7.19.5.5 multiplicity . . . 158
7.19.5.6 operators . . . 158
7.19.5.7 sums . . . 158
7.20 mrock::symbolic_operators::TermLoader Struct Reference . . . 159
7.20.1 Detailed Description . . . 159
7.20.2 Member Function Documentation . . . 159
7.20.2.1 load() . . . 159
7.20.3 Member Data Documentation . . . 160
7.20.3.1 M . . . 160
7.20.3.2 N . . . 160
7.21 mrock::symbolic_operators::WickOperator Class Reference . . . 160
7.21.1 Detailed Description . . . 161
7.21.2 Constructor & Destructor Documentation . . . 161
7.21.2.1 WickOperator() [1/4] . . . 161
7.21.2.2 WickOperator() [2/4] . . . 162
7.21.2.3 WickOperator() [3/4] . . . 162
7.21.2.4 WickOperator() [4/4] . . . 162
7.21.3 Member Function Documentation . . . 163
7.21.3.1 depends_on() . . . 163
7.21.3.2 remove_momentum_contribution() . . . 163
7.21.3.3 serialize() . . . 164
7.21.3.4 uses_index() . . . 164
7.21.4 Member Data Documentation . . . 164
7.21.4.1 indizes . . . 164
7.21.4.2 is_daggered . . . 165
7.21.4.3 momentum . . . 165
7.21.4.4 type . . . 165
7.22 mrock::symbolic_operators::WickOperatorTemplate Class Reference . . . 165
7.22.1 Detailed Description . . . 166
7.22.2 Member Function Documentation . . . 166
7.22.2.1 _handle_num_type() . . . 166
7.22.2.2 _handle_sc_type() . . . 167
7.22.2.3 create_from_operators() . . . 167
7.22.3 Member Data Documentation . . . 167

7.22.3.1 indexComparison . . . 168
7.22.3.2 is_sc_type . . . 168
7.22.3.3 momentum_difference . . . 168
7.22.3.4 type . . . 168
7.23 mrock::symbolic_operators::WickSymmetry Class Reference . . . 169
7.23.1 Detailed Description . . . 169
7.23.2 Constructor & Destructor Documentation . . . 170
7.23.2.1 ∼WickSymmetry() . . . 170
7.23.3 Member Function Documentation . . . 170
7.23.3.1 apply_to() . . . 170
7.24 mrock::symbolic_operators::WickTerm Class Reference . . . 170
7.24.1 Detailed Description . . . 172
7.24.2 Constructor & Destructor Documentation . . . 173
7.24.2.1 WickTerm() [1/5] . . . 173
7.24.2.2 WickTerm() [2/5] . . . 173
7.24.2.3 WickTerm() [3/5] . . . 174
7.24.2.4 WickTerm() [4/5] . . . 174
7.24.2.5 WickTerm() [5/5] . . . 174
7.24.3 Member Function Documentation . . . 175
7.24.3.1 compute_sums() . . . 175
7.24.3.2 discard_zero_momenta() . . . 175
7.24.3.3 get_factor() . . . 175
7.24.3.4 get_first_coefficient() . . . 176
7.24.3.5 handled() . . . 176
7.24.3.6 has_single_coefficient() . . . 176
7.24.3.7 include_template_result() . . . 176
7.24.3.8 includes_type() . . . 177
7.24.3.9 invert_momentum() . . . 177
7.24.3.10 invert_momentum_sum() . . . 177
7.24.3.11 is_bilinear() . . . 178
7.24.3.12 is_identity() . . . 178
7.24.3.13 is_quartic() . . . 178
7.24.3.14 remove_momentum_contribution() . . . 178
7.24.3.15 rename_sums() . . . 179
7.24.3.16 serialize() . . . 179
7.24.3.17 set_deltas() . . . 179
7.24.3.18 sort() . . . 180
7.24.3.19 string_parser() . . . 180
7.24.3.20 uses_index() . . . 180
7.24.3.21 which_operator_depends_on() . . . 181
7.24.4 Member Data Documentation . . . 181
7.24.4.1 coefficients . . . 181

7.24.4.2 delta_indizes . . . 181
7.24.4.3 delta_momenta . . . 182
7.24.4.4 multiplicity . . . 182
7.24.4.5 operators . . . 182
7.24.4.6 sums . . . 182
7.24.4.7 temporary_operators . . . 182
7.25 mrock::symbolic_operators::WickTermCollector Class Reference . . . 183
7.25.1 Detailed Description . . . 183
7.25.2 Member Function Documentation . . . 184
7.25.2.1 serialize() . . . 184

8 File Documentation **185**

8.1 include/mrock/symbolic_operators/Coefficient.hpp File Reference . . . 185
8.1.1 Detailed Description . . . 185
8.2 include/mrock/symbolic_operators/IndexWrapper.hpp File Reference . . . 186
8.2.1 Detailed Description . . . 187
8.3 include/mrock/symbolic_operators/KroneckerDelta.hpp File Reference . . . 187
8.3.1 Detailed Description . . . 188
8.4 include/mrock/symbolic_operators/KroneckerDeltaUtility.hpp File Reference . . . 188
8.4.1 Detailed Description . . . 189
8.5 include/mrock/symbolic_operators/Momentum.hpp File Reference . . . 189
8.5.1 Detailed Description . . . 190
8.6 include/mrock/symbolic_operators/MomentumList.hpp File Reference . . . 190
8.6.1 Detailed Description . . . 191
8.7 include/mrock/symbolic_operators/MomentumSymbol.hpp File Reference . . . 191
8.7.1 Detailed Description . . . 192
8.8 include/mrock/symbolic_operators/Operator.hpp File Reference . . . 192
8.8.1 Detailed Description . . . 192
8.9 include/mrock/symbolic_operators/OperatorType.hpp File Reference . . . 192
8.9.1 Detailed Description . . . 193
8.10 include/mrock/symbolic_operators/SumContainer.hpp File Reference . . . 193
8.10.1 Detailed Description . . . 194
8.11 include/mrock/symbolic_operators/SymbolicSum.hpp File Reference . . . 194
8.11.1 Detailed Description . . . 195
8.12 include/mrock/symbolic_operators/Term.hpp File Reference . . . 195
8.12.1 Detailed Description . . . 196
8.12.2 Macro Definition Documentation . . . 196
8.12.2.1 _TERM_TRACKER_ATTRIBUTE . . . 196
8.12.2.2 _TERM_TRACKER_PARAMETER . . . 197
8.12.2.3 CLEAR_TRACKED . . . 197
8.12.2.4 IF_IS_TERM_TRACKED . . . 197
8.13 include/mrock/symbolic_operators/TermLoader.hpp File Reference . . . 197

8.13.1 Detailed Description . . . 197
8.14 include/mrock/symbolic_operators/Wick.hpp File Reference . . . 198
8.14.1 Detailed Description . . . 198
8.15 include/mrock/symbolic_operators/WickOperator.hpp File Reference . . . 198
8.15.1 Detailed Description . . . 199
8.16 include/mrock/symbolic_operators/WickOperatorTemplate.hpp File Reference . . . 199
8.16.1 Detailed Description . . . 200
8.17 include/mrock/symbolic_operators/WickSymmetry.hpp File Reference . . . 200
8.17.1 Detailed Description . . . 200
8.18 include/mrock/symbolic_operators/WickTerm.hpp File Reference . . . 200
8.18.1 Detailed Description . . . 202
8.19 mainpage.dox File Reference . . . 202
8.20 sources/Coefficient.cpp File Reference . . . 202
8.21 sources/IndexWrapper.cpp File Reference . . . 202
8.22 sources/Momentum.cpp File Reference . . . 203
8.23 sources/MomentumList.cpp File Reference . . . 203
8.24 sources/Operator.cpp File Reference . . . 203
8.25 sources/OperatorType.cpp File Reference . . . 204
8.26 sources/SumContainer.cpp File Reference . . . 204
8.27 sources/Term.cpp File Reference . . . 204
8.27.1 Macro Definition Documentation . . . 205
8.27.1.1 fill_reciever . . . 205
8.28 sources/TermLoader.cpp File Reference . . . 205
8.29 sources/Wick.cpp File Reference . . . 206
8.30 sources/WickOperator.cpp File Reference . . . 206
8.31 sources/WickOperatorTemplate.cpp File Reference . . . 207
8.32 sources/WickSymmetry.cpp File Reference . . . 207
8.33 sources/WickTerm.cpp File Reference . . . 207
8.33.1 Macro Definition Documentation . . . 208
8.33.1.1 L_SPIN . . . 208
8.33.1.2 LEFT . . . 208
8.33.1.3 R_SPIN . . . 209
8.33.1.4 RIGHT . . . 209
8.34 tests/bosons.cpp File Reference . . . 209
8.34.1 Detailed Description . . . 209
8.34.2 Function Documentation . . . 209
8.34.2.1 main() . . . 210
8.34.3 Variable Documentation . . . 210
8.34.3.1 begin_align . . . 210
8.34.3.2 COMPARE_DIR . . . 210
8.34.3.3 end_align . . . 210
8.34.3.4 file_names . . . 210

8.35 tests/compare_test.hpp File Reference 211
8.36 tests/continuum.cpp File Reference 211
8.36.1 Detailed Description 211
8.36.2 Function Documentation 211
8.36.2.1 main() 211

Index **213**

Chapter 1

symbolic_operators Library Documentation

1.1 Introduction

The `symbolic_operators` library provides tools for symbolic manipulation of creation and annihilation operators. It includes classes for defining Hamiltonians, performing commutation operations, and applying symmetries.

1.2 Example Usage

The following example demonstrates how to define a Hamiltonian using bosonic operators and perform commutation operations. We will use the Hamiltonian

$$H = \sum_k \gamma(k) b^\dagger_{k,A} b_{k,B} \tag{1.1}$$
$$+ \sum_k \gamma^*(k) b^\dagger_{k,B} b_{k,A} \tag{1.2}$$
$$+ \frac{1}{2} \sum_k \Gamma(k) b^\dagger_{k,A} b^\dagger_{-k,B} \tag{1.3}$$
$$+ \frac{1}{2} \sum_k \Gamma^*(k) b_{-k,B} b_{k,A} \tag{1.4}$$
$$- \sum_k \sum_\sigma \mu_\sigma b^\dagger_{k,\sigma} b_{k,\sigma}. \tag{1.5}$$

We start by the defining some abbreviations and setting the necessary includes:

```
#include <mrock/symbolic_operators/Term.hpp>
#include <vector>
#include <string>
#include <iostream>
using namespace mrock::symbolic_operators;
const std::string begin_align = "\\begin{align*}\n\t";
const std::string end_align = "\\end{align*}\n";
```

Now, beginning in our main function, we define our Hamiltonian by first defining its comprising terms and setting putting them together:

```
int main(int argc, char** argv) {
    // Define the Hamiltonian
    // The hopping term (Eq.  1)
    const Term hopping(1, Coefficient::HoneyComb("\\gamma", Momentum('k'), false, false),
        MomentumSum{'k'},
        std::vector<Operator>({
            Operator::Boson(Momentum('k'), Index::TypeA, true),
            Operator::Boson(Momentum('k'), Index::TypeB, false)
        }));
```

```
    // The Bogoliubov term (Eq.  3)
    const Term bogo(IntFractional(1, 2), Coefficient::HoneyComb("\\gamma'", Momentum('k'), false, false),
        MomentumSum{'k'},
        std::vector<Operator>({
            Operator::Boson(Momentum('k'), Index::TypeA, true),
            Operator::Boson(Momentum('k', -1), Index::TypeB, true)
        }));
    // The chemical potential term (Eq.  5)
    const Term chemical_potential(-1, Coefficient::Constant("\\mu", Index::Sigma),
        SumContainer{ MomentumSum{'k'}, IndexSum{Index::Sigma} },
        std::vector<Operator>({
            Operator::Boson(Momentum('k'), Index::Sigma, true),
            Operator::Boson(Momentum('k'), Index::Sigma, false)
        }));
    const std::vector<Term> hamiltonian {
        hopping,                        // Eq.  1
        hopping.hermitian_conjugate(),  // Eq.  2
        bogo,                           // Eq.  3
        bogo.hermitian_conjugate(),     // Eq.  4
        chemical_potential              // Eq.  5
    };

    std::cout « begin_align « "H =" « hamiltonian « end_align « std::endl;
```

Let es explain what is happening here. As an example we consider the hopping terms. It is created using a constructor of `Term`:

```
Term(Integer or IntFractional,
```

We used `1` ⇒ the Term has a constant prefactor of 1. Compare the Bogoliubov terms, where we used `IntFractional(1,2)`, representing 1/2.

```
Coefficient::HoneyComb(name, Momentum('k'), not a complex conjugate, not real) //
```

Creates a coefficient without inversion symmetry that is not real, but also not a complex conjugate. It has the momentum `k` ⇒ $\gamma(k)$.

```
MomentumSum{'k'},
```

Represents $\sum_k$, i.e., we sum over all momenta k. Compare with the chemical potential term, where we also sum over an arbitrary index called σ.

```
    std::vector<Operator>({
        Operator::Boson(Momentum('k'), Index::TypeA, true),
        Operator::Boson(Momentum('k'), Index::TypeB, false)
    })
);
```

Pass the operators of the term, i.e., $b^\dagger_{k,A} b_{k,B}$. The first operator is $b^\dagger_{k,A}$ while while the second one is $b_{k,B}$. See also the other terms for comparison.

After printing the Hamiltonian and confirming it is, what we want it to be, we define some terms to commute with:

```
// Define the commutation targets
const Term to_commute_1(1, std::vector<Operator>({
        Operator::Boson(Momentum('l'), Index::TypeA, true),
        Operator::Boson(Momentum('l', -1), Index::TypeB, true)
    }));
const std::vector<Term> to_commute_2{
    Term(1, SumContainer{ MomentumSum{'q'} },
        std::vector<Operator>({
            Operator::Boson(Momentum("l+q"), Index::TypeA, true),
            Operator::Boson(Momentum("l"), Index::TypeA, false)
        })),
    Term(1, SumContainer{ MomentumSum{'q'} },
        std::vector<Operator>({
            Operator::Boson(Momentum("l-q"), Index::TypeB, true),
            Operator::Boson(Momentum("l"), Index::TypeB, false)
        }))
    };
```

And finially, we perform the commutation, clean up, and print the results to the console within a LaTeX-align environment:

```
    // Compute the commutators
    std::vector<Term> result_1 = commutator(hamiltonian, to_commute_1);
    clean_up(result_1);
    std::vector<Term> result_2 = commutator(hamiltonian, to_commute_2);
    clean_up(result_2);
    std::cout « begin_align « "[H, " « to_commute_1.to_string_without_prefactor() « "] = " « result_1 «
      end_align « std::endl;
    std::cout « begin_align « "[H, " « to_string_without_prefactor(to_commute_2) « "]  = " « result_2 «
      end_align « std::endl;
    return 0;
}
```

1.3 Class Explanations

- Coefficient: Represents a coefficient that can have various symmetries.
- Momentum: Represents a momentum.
- Term: Represents a term in symbolic operator expressions.
- WickSymmetry: Abstract base class for symmetries of the expectation values occurring in Wick's theorem.
- WickTerm: Represents a term consisting of expectation values after applying Wick's theorem. Currently only implemented for Fermions.

Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

mrock 13
mrock::symbolic_operators 13
sym_op_test 57

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

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

mrock::symbolic_operators::Coefficient . . . 62
mrock::symbolic_operators::IndexComparison . . . 74
mrock::symbolic_operators::IndexWrapper . . . 75
mrock::symbolic_operators::KroneckerDelta< T > . . . 80
mrock::symbolic_operators::KroneckerDelta< mrock::symbolic_operators::Momentum > . . . 80
mrock::symbolic_operators::Momentum . . . 82
mrock::symbolic_operators::MomentumSymbol . . . 102
mrock::symbolic_operators::MomentumSymbol::name_type . . . 105
mrock::symbolic_operators::Operator . . . 108
std::runtime_error
    mrock::symbolic_operators::bad_term_exception . . . 59
mrock::symbolic_operators::TemplateResult::SingleResult . . . 119
mrock::symbolic_operators::SumContainer . . . 123
mrock::symbolic_operators::SymbolicSum< SumIndex > . . . 128
mrock::symbolic_operators::SymbolicSum< Index > . . . 128
mrock::symbolic_operators::SymbolicSum< MomentumSymbol::name_type > . . . 128
sym_op_test::SymOpTest . . . 132
mrock::symbolic_operators::TemplateResult . . . 135
mrock::symbolic_operators::Term . . . 140
mrock::symbolic_operators::TermLoader . . . 159
mrock::utility::VectorWrapper
    mrock::symbolic_operators::MomentumList . . . 95
    mrock::symbolic_operators::WickTermCollector . . . 183
mrock::symbolic_operators::WickOperator . . . 160
mrock::symbolic_operators::WickOperatorTemplate . . . 165
mrock::symbolic_operators::WickSymmetry . . . 169
    mrock::symbolic_operators::InversionSymmetry . . . 79
    mrock::symbolic_operators::PhaseSymmetry< operators> . . . 117
    mrock::symbolic_operators::SpinSymmetry . . . 122
mrock::symbolic_operators::WickTerm . . . 170

Chapter 4

Class Index

4.1 Class List

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

mrock::symbolic_operators::bad_term_exception
An exception class for bad terms . . . 59
mrock::symbolic_operators::Coefficient
Represents a coefficient. Various symmetries are pre defined (e.g. inversion symmetry) and can be toggled on or off A custom symmetry can also be provided . . . 62
mrock::symbolic_operators::IndexComparison
A structure for comparing indices. E.g. <n_k> merely requires that the spin indizes of the composing operators are identical, but <f_k> requires the first index to be spin down . . . 74
mrock::symbolic_operators::IndexWrapper
A wrapper for a vector of Index values . . . 75
mrock::symbolic_operators::InversionSymmetry
A symmetry where expectation values for k and -k are the same . . . 79
mrock::symbolic_operators::KroneckerDelta< T >
A structure representing the Kronecker Delta . . . 80
mrock::symbolic_operators::Momentum
Represents a collection of momentum symbols with associated operations . . . 82
mrock::symbolic_operators::MomentumList
A wrapper class for a vector of Momentum objects with additional functionalities . . . 95
mrock::symbolic_operators::MomentumSymbol
Represents a symbolic momentum with a factor and a name . . . 102
mrock::symbolic_operators::MomentumSymbol::name_type
Represents a name as a single character with comparison and serialization capabilities, but without arithmetic operations (it does not make sense to add or multiply names) . . . 105
mrock::symbolic_operators::Operator
Represents a symbolic operator with momentum, indices, and properties . . . 108
mrock::symbolic_operators::PhaseSymmetry< operators>
A symmetry where <operator^+> = <operator> . . . 117
mrock::symbolic_operators::TemplateResult::SingleResult
A structure for storing a single result . . . 119
mrock::symbolic_operators::SpinSymmetry
A symmetry where expectation values for spin up and down are the same . . . 122
mrock::symbolic_operators::SumContainer
A container for holding symbolic sums of momenta and spins . . . 123
mrock::symbolic_operators::SymbolicSum< SumIndex >
A struct representing a symbolic summation operation . . . 128

sym_op_test::SymOpTest 132
mrock::symbolic_operators::TemplateResult
A structure for storing the result of a template operation 135
mrock::symbolic_operators::Term
Represents a term in symbolic operator expressions 140
mrock::symbolic_operators::TermLoader
A structure to load and manage Wick terms 159
mrock::symbolic_operators::WickOperator
A structure representing a Wick operator 160
mrock::symbolic_operators::WickOperatorTemplate
A template for creating Wick operators 165
mrock::symbolic_operators::WickSymmetry
An abstract base class for Wick symmetries 169
mrock::symbolic_operators::WickTerm
A structure representing a Wick term 170
mrock::symbolic_operators::WickTermCollector
A wrapper for a vector of WickTerm objects 183

Chapter 5

File Index

5.1 File List

Here is a list of all files with brief descriptions:

include/mrock/symbolic_operators/Coefficient.hpp
Defines the Coefficient structure used in symbolic operators . . . 185
include/mrock/symbolic_operators/IndexWrapper.hpp
Defines the Index enum and the IndexWrapper class for handling indizes . . . 186
include/mrock/symbolic_operators/KroneckerDelta.hpp
Defines the KroneckerDelta structure used in symbolic operators . . . 187
include/mrock/symbolic_operators/KroneckerDeltaUtility.hpp
Utility functions for manipulating KroneckerDelta objects . . . 188
include/mrock/symbolic_operators/Momentum.hpp
Defines the Momentum structure and related operations for symbolic manipulation of momentum symbols . . . 189
include/mrock/symbolic_operators/MomentumList.hpp
Defines the MomentumList class for handling a list of Momentum objects . . . 190
include/mrock/symbolic_operators/MomentumSymbol.hpp
Defines the MomentumSymbol structure and related operators for symbolic operations . . . 191
include/mrock/symbolic_operators/Operator.hpp
Defines the Operator struct and related functions for symbolic operators . . . 192
include/mrock/symbolic_operators/OperatorType.hpp
Defines the OperatorType enum and related functions for symbolic operators . . . 192
include/mrock/symbolic_operators/SumContainer.hpp
Defines the SumContainer structure and related operators for symbolic operations . . . 193
include/mrock/symbolic_operators/SymbolicSum.hpp
Defines the SymbolicSum template struct for symbolic summation operations . . . 194
include/mrock/symbolic_operators/Term.hpp
Defines the Term class and related functions for symbolic operators . . . 195
include/mrock/symbolic_operators/TermLoader.hpp
Header file for the TermLoader structure in the symbolic_operators namespace . . . 197
include/mrock/symbolic_operators/Wick.hpp
Functions for applying Wick's theorem and manipulating Wick terms . . . 198
include/mrock/symbolic_operators/WickOperator.hpp
Defines the WickOperator structure used in symbolic operators . . . 198
include/mrock/symbolic_operators/WickOperatorTemplate.hpp
Defines templates for creating Wick operators from a set of normal operators . . . 199
include/mrock/symbolic_operators/WickSymmetry.hpp
Defines symmetries for Wick terms . . . 200

include/mrock/symbolic_operators/WickTerm.hpp
Defines the WickTerm structure and related functions . . . 200
sources/Coefficient.cpp . . . 202
sources/IndexWrapper.cpp . . . 202
sources/Momentum.cpp . . . 203
sources/MomentumList.cpp . . . 203
sources/Operator.cpp . . . 203
sources/OperatorType.cpp . . . 204
sources/SumContainer.cpp . . . 204
sources/Term.cpp . . . 204
sources/TermLoader.cpp . . . 205
sources/Wick.cpp . . . 206
sources/WickOperator.cpp . . . 206
sources/WickOperatorTemplate.cpp . . . 207
sources/WickSymmetry.cpp . . . 207
sources/WickTerm.cpp . . . 207
tests/bosons.cpp
Example code for defining and using bosonic operators . . . 209
tests/compare_test.hpp . . . 211
tests/continuum.cpp
Example code for defining and using continuum operators . . . 211

Chapter 6

Namespace Documentation

6.1 mrock Namespace Reference

Namespaces

- symbolic_operators

6.2 mrock::symbolic_operators Namespace Reference

Classes

- struct Coefficient

  Represents a coefficient. Various symmetries are pre defined (e.g. inversion symmetry) and can be toggled on or off A custom symmetry can also be provided.
- struct IndexWrapper

  A wrapper for a vector of Index values.
- class KroneckerDelta

  A structure representing the Kronecker Delta.
- struct Momentum

  Represents a collection of momentum symbols with associated operations.
- class MomentumList

  A wrapper class for a vector of Momentum objects with additional functionalities.
- struct MomentumSymbol

  Represents a symbolic momentum with a factor and a name.
- struct Operator

  Represents a symbolic operator with momentum, indices, and properties.
- struct SumContainer

  A container for holding symbolic sums of momenta and spins.
- struct SymbolicSum

  A struct representing a symbolic summation operation.
- class Term

  Represents a term in symbolic operator expressions.
- struct TermLoader

  A structure to load and manage Wick terms.

- class WickOperator

  A structure representing a Wick operator.
- struct IndexComparison

  A structure for comparing indices. E.g. <n_k> merely requires that the spin indizes of the composing operators are identical, but <f_k> requires the first index to be spin down.
- struct TemplateResult

  A structure for storing the result of a template operation.
- class WickOperatorTemplate

  A template for creating Wick operators.
- class WickSymmetry

  An abstract base class for Wick symmetries.
- class SpinSymmetry

  A symmetry where expectation values for spin up and down are the same.
- class InversionSymmetry

  A symmetry where expectation values for k and -k are the same.
- class PhaseSymmetry

  A symmetry where <operator^+> = <operator>.
- class WickTerm

  A structure representing a Wick term.
- class WickTermCollector

  A wrapper for a vector of WickTerm objects.
- class bad_term_exception

  An exception class for bad terms.

Typedefs

- typedef unsigned char index_base

  Defines the base type for the Index enum as unsigned char.
- typedef std::vector< MomentumSymbol > momentum_symbols

  Alias for a vector of MomentumSymbol.
- typedef SymbolicSum< Index > IndexSum

  Typedef for SymbolicSum with Index type.
- typedef SymbolicSum< MomentumSymbol::name_type > MomentumSum

  Typedef for SymbolicSum with MomentumSymbol::name_type type.
- using IntFractional = mrock::utility::Fractional< int >

Enumerations

- enum class Index : index_base {
  SpinUp = 0 , SpinDown , Sigma , SigmaPrime ,
  GeneralSpin_S , GeneralSpin_SPrime , TypeA , TypeB ,
  TypeC , char_a = 97 , UndefinedIndex = 254 , NoIndex = 255 }

  Enumeration representing various symbolic indices.
- enum OperatorType {
  Number_Type = 0 , CDW_Type , SC_Type , Eta_Type ,
  Undefined_Type = 255 }

Functions

- bool operator== (const Coefficient &lhs, const Coefficient &rhs)

  Equality operator for Coefficient.
- bool operator!= (const Coefficient &lhs, const Coefficient &rhs)

  Inequality operator for Coefficient.
- constexpr Index char_to_index (unsigned char c)

  Converts a character to an Index.
- constexpr bool is_mutable (const Index idx)

  Checks if the given index represents a variable (mutable). 'Mutable' means that it is associated with a sum or similar. An example is sigma; it is commonly summed over as a representation of spins. Then expressions like delta_{sigma,up} can be evaluated to be one if sigma=up. An Index like SpinUp is set to be non-mutable. This allows us to evaluate delta_{up,down}=0.
- std::ostream & operator<< (std::ostream &os, const Index index)

  Overloads the stream insertion operator for the Index enum.
- std::ostream & operator<< (std::ostream &os, const IndexWrapper &indizes)

  Overloads the stream insertion operator for the IndexWrapper struct.
- template<typename T >
  constexpr auto make_delta (const T &first, const T &second)

  Creates a KroneckerDelta object.
- template<typename T >
  constexpr auto make_delta (std::decay_t< T > &&first, std::decay_t< T > &&second)

  Creates a KroneckerDelta object with rvalue references.
- template<typename T >
  bool operator== (const KroneckerDelta< T > &lhs, const KroneckerDelta< T > &rhs)

  Equality operator for KroneckerDelta.
- template<typename T >
  bool operator!= (const KroneckerDelta< T > &lhs, const KroneckerDelta< T > &rhs)

  Inequality operator for KroneckerDelta.
- template<typename T >
  requires mrock::utility::defines_plus< T >::value KroneckerDelta< T > & operator+= (KroneckerDelta< T > &lhs, T &rhs)

  Addition assignment operator for KroneckerDelta.
- template<typename T >
  requires mrock::utility::defines_minus< T >::value KroneckerDelta< T > & operator-= (KroneckerDelta< T > &lhs, const T &rhs)

  Subtraction assignment operator for KroneckerDelta.
- template<typename T >
  requires mrock::utility::defines_plus< T >::value KroneckerDelta< T > operator+ (KroneckerDelta< T > lhs, T const &rhs)

  Addition operator for KroneckerDelta.
- template<typename T >
  requires mrock::utility::defines_minus< T >::value KroneckerDelta< T > operator- (KroneckerDelta< T > lhs, T const &rhs)

  Subtraction operator for KroneckerDelta.
- template<typename T >
  std::ostream & operator<< (std::ostream &os, const KroneckerDelta< T > &delta)

  Stream insertion operator for KroneckerDelta.
- template<class T >
  void remove_delta_squared (std::vector< KroneckerDelta< T >> &deltas)

  Removes squared KroneckerDelta objects from the vector. Note that delta_{a,b}^N = delta_{a,b}.
- template<class T >
  void remove_delta_is_one (std::vector< KroneckerDelta< T >> &deltas)

Removes KroneckerDelta objects that are one from the vector. Note that delta_{a,a} = 1.

- bool is_always_zero (const std::vector< KroneckerDelta< Index >> &deltas)

  Checks if the vector of KroneckerDelta<Index> objects is always zero.
- bool is_always_zero (const std::vector< KroneckerDelta< Momentum >> &deltas)

  Checks if the vector of KroneckerDelta<Momentum> objects is always zero.
- void remove_double_occurances (KroneckerDelta< Momentum > &delta)

  Removes double occurrences in a KroneckerDelta<Momentum> object.
- bool momentum_order (const Momentum &lhs, const Momentum &rhs)

  Compares two Momentum objects for ordering.
- Momentum operator+ (Momentum lhs, const Momentum &rhs)

  Adds two Momentum objects.
- Momentum operator- (Momentum lhs, const Momentum &rhs)

  Subtracts one Momentum from another.
- Momentum operator* (Momentum lhs, const int rhs)

  Multiplies a Momentum by an integer factor.
- Momentum operator* (const int lhs, Momentum rhs)

  Multiplies an integer factor by a Momentum.
- Momentum operator- (Momentum rhs)

  Negates a Momentum.
- bool operator> (const Momentum &lhs, const Momentum &rhs)

  Compares two Momentum objects for greater-than ordering.
- bool operator< (const Momentum &lhs, const Momentum &rhs)

  Compares two Momentum objects for less-than ordering.
- std::ostream & operator<< (std::ostream &os, const Momentum &momentum)

  Outputs a Momentum to an output stream.
- std::ostream & operator<< (std::ostream &os, const MomentumList &momenta)

  Outputs the MomentumList to an output stream.
- std::ostream & operator<< (std::ostream &os, const MomentumSymbol::name_type name)

  Outputs the name_type to an output stream.
- std::istream & operator>> (std::istream &is, MomentumSymbol::name_type &name)

  Inputs a name_type from an input stream.
- std::string operator+ (const std::string &str, const MomentumSymbol::name_type sym)

  Concatenates a string and a name_type.
- std::string operator+ (const MomentumSymbol::name_type sym, const std::string &str)

  Concatenates a name_type and a string.
- bool operator== (const Operator &lhs, const Operator &rhs)

  Equality operator for Operator.
- bool operator!= (const Operator &lhs, const Operator &rhs)

  Inequality operator for Operator.
- std::ostream & operator<< (std::ostream &os, const Operator &op)

  Stream insertion operator for Operator.
- std::ostream & operator<< (std::ostream &os, const std::vector< Operator > &ops)

  Stream insertion operator for a vector of Operators.
- std::ostream & operator<< (std::ostream &os, const OperatorType op)

  Overloads the stream insertion operator for the Index enum.
- bool operator== (const SumContainer &lhs, const SumContainer &rhs)

  Equality operator for SumContainer.
- bool operator!= (const SumContainer &lhs, const SumContainer &rhs)

  Inequality operator for SumContainer.
- std::ostream & operator<< (std::ostream &os, const SumContainer &sums)

  Stream insertion operator for SumContainer.

- template<class SumIndex >
  std::ostream & operator<< (std::ostream &os, SymbolicSum< SumIndex > const &sum)

  Outputs the SymbolicSum object to an output stream.
- std::vector< Term > commutator (const std::vector< Term > &left, const std::vector< Term > &right)

  Computes the commutator of two sets of terms: $[A, B] = AB - BA$.
- std::vector< Term > commutator (const Term &left, const std::vector< Term > &right)

  Computes the commutator of a term and a set of terms: $[A, B] = AB - BA$.
- std::vector< Term > commutator (const std::vector< Term > &left, const Term &right)

  Computes the commutator of a set of terms and a term: $[A, B] = AB - BA$.
- bool operator== (const Term &lhs, const Term &rhs)

  Checks if two terms are equal.
- bool operator!= (const Term &lhs, const Term &rhs)

  Checks if two terms are not equal.
- std::ostream & operator<< (std::ostream &os, const Coefficient &coeff)

  Outputs a coefficient to a stream.
- std::ostream & operator<< (std::ostream &os, const std::vector< Coefficient > &coeffs)

  Outputs a vector of coefficients to a stream.
- std::ostream & operator<< (std::ostream &os, const std::vector< Term > &terms)

  Outputs a vector of terms to a stream.
- void clear_duplicates (std::vector< Term > &terms)

  Clears duplicate terms from a vector.
- void clean_up (std::vector< Term > &terms)

  Sorts the terms, adds identical ones together and removes those that are equal to 0.
- void hermitian_conjugate (std::vector< Term > &terms)

  Applies the Hermitian conjugate to a vector of terms.
- void rename_momenta (std::vector< Term > &terms, const MomentumSymbol::name_type what, const MomentumSymbol::name_type to)

  Renames momenta in a vector of terms.
- std::string to_string_without_prefactor (const std::vector< Term > &terms)

  Converts a vector of terms to a string without the prefactor.
- WickTermCollector identify_wick_operators (const WickTerm &source, const std::vector< WickOperatorTemplate > &operator_templates)

  Identifies Wick operators in a given Wick term.
- void wicks_theorem (const std::vector< Term > &terms, const std::vector< WickOperatorTemplate > &operator_templates, WickTermCollector &reciever)

  Applies Wick's theorem to a set of terms.
- void clear_etas (WickTermCollector &terms)

  Clears eta terms from the WickTermCollector. Intended for use if <eta>=0.
- void clean_wicks (WickTermCollector &terms, const std::vector< std::unique_ptr< WickSymmetry >> &symmetries=std::vector< std::unique_ptr< WickSymmetry >>{})

  Cleans Wick terms using the provided symmetries.
- std::ostream & operator<< (std::ostream &os, const WickOperator &op)

  Stream insertion operator for WickOperator.
- std::ostream & operator<< (std::ostream &os, const std::vector< WickOperator > &ops)

  Stream insertion operator for a vector of WickOperator objects.
- bool operator== (const WickOperator &lhs, const WickOperator &rhs)

  Equality operator for WickOperator.
- bool operator!= (const WickOperator &lhs, const WickOperator &rhs)

  Inequality operator for WickOperator.
- bool operator== (const WickTerm &lhs, const WickTerm &rhs)

  Equality operator for WickTerm.

- bool operator!= (const WickTerm &lhs, const WickTerm &rhs)

  Inequality operator for WickTerm.
- WickTermCollector & operator+= (WickTermCollector &lhs, const WickTerm &rhs)

  Addition assignment operator for WickTermCollector and WickTerm.
- WickTermCollector & operator-= (WickTermCollector &lhs, const WickTerm &rhs)

  Subtraction assignment operator for WickTermCollector and WickTerm.
- WickTermCollector & operator+= (WickTermCollector &lhs, const WickTermCollector &rhs)

  Addition assignment operator for two WickTermCollector objects.
- WickTermCollector & operator-= (WickTermCollector &lhs, const WickTermCollector &rhs)

  Subtraction assignment operator for two WickTermCollector objects.
- WickTermCollector operator+ (WickTermCollector lhs, const WickTerm &rhs)

  Addition operator for WickTermCollector and WickTerm.
- WickTermCollector operator- (WickTermCollector lhs, const WickTerm &rhs)

  Subtraction operator for WickTermCollector and WickTerm.
- WickTermCollector operator+ (const WickTerm &lhs, WickTermCollector rhs)

  Addition operator for WickTerm and WickTermCollector.
- WickTermCollector operator- (const WickTerm &lhs, WickTermCollector rhs)

  Subtraction operator for WickTerm and WickTermCollector.
- WickTermCollector operator+ (WickTermCollector lhs, const WickTermCollector &rhs)

  Addition operator for two WickTermCollector objects.
- WickTermCollector operator- (WickTermCollector lhs, const WickTermCollector &rhs)

  Subtraction operator for two WickTermCollector objects.
- std::ostream & operator<< (std::ostream &os, const WickTerm &term)

  Stream insertion operator for WickTerm.
- std::ostream & operator<< (std::ostream &os, const WickTermCollector &terms)

  Stream insertion operator for WickTermCollector.
- momentum_symbols::value_type identify_subexpression (const std::string &sub)
- void normal_order (std::vector< Term > &terms)
- std::vector< Term > commutator (const Term &left, const Term &right)
- std::ostream & operator<< (std::ostream &os, const Term &term)
- void wick_processor (const std::vector< Operator > &remaining, WickTermCollector &reciever_list, std::variant< WickTerm, Term > buffer)
- WickTermCollector prepare_wick (const std::vector< Term > &terms)

Variables

- const std::map< std::string, Index > string_to_index

  A map that associates string representations with their corresponding Index values.
- const std::map< std::string, OperatorType > string_to_wick

  A map that associates string representations with their corresponding Wick operators values.

6.2.1 Typedef Documentation

6.2.1.1 index_base

`mrock::symbolic_operators::index_base`

Defines the base type for the Index enum as unsigned char.

Definition at line 18 of file IndexWrapper.hpp.

6.2.1.2 IndexSum

`mrock::symbolic_operators::IndexSum`

Typedef for SymbolicSum with Index type.

Definition at line 18 of file SumContainer.hpp.

6.2.1.3 IntFractional

```
using mrock::symbolic_operators::IntFractional = typedef mrock::utility::Fractional<int>
```

Definition at line 31 of file Term.hpp.

6.2.1.4 momentum_symbols

`mrock::symbolic_operators::momentum_symbols`

Alias for a vector of MomentumSymbol.

Definition at line 25 of file Momentum.hpp.

6.2.1.5 MomentumSum

`mrock::symbolic_operators::MomentumSum`

Typedef for SymbolicSum with MomentumSymbol::name_type type.

Definition at line 24 of file SumContainer.hpp.

6.2.2 Enumeration Type Documentation

6.2.2.1 Index

```
enum mrock::symbolic_operators::Index : index_base [strong]
```

Enumeration representing various symbolic indices.

The indices include:

- SpinUp: Represents spin up (0).
- SpinDown: Represents spin down (1).
- Sigma: Represents sigma (2).
- SigmaPrime: Represents sigma prime (3).
- GeneralSpin_S: Represents general spin S (4).
- GeneralSpin_SPrime: Represents general spin S prime (5).
- TypeA: Represents type A (6).
- TypeB: Represents type B (7).
- TypeC: Represents type C (8).
- char_a: Represents the lowercase ASCII character 'a' (97).
- UndefinedIndex: Represents an undefined index (254).
- NoIndex: Represents no index (255).
- Not explicitly represented, but defined implementation-wise are any lower-case ASCII characters.char_to_↩ index. These can be obtained via static_cast<Index>(char) or using the char_to_index() function

The indices include:

- Number_Type: <n> (0).
- CDW_Type: <c_{k+Q}^+ c_k> (1).
- SC_Type: <c_{-k down} c_{k up}> (2).
- Eta_Type: <c_{-k-Q down} c_{k up}> (3).
- Undefined_Type: Represents an undefined type (4).

Enumerator

SpinUp	
SpinDown	
Sigma	
SigmaPrime	
GeneralSpin_S	
GeneralSpin_SPrime	
TypeA	
TypeB	
TypeC	
char_a	
UndefinedIndex	
NoIndex	

Definition at line 41 of file IndexWrapper.hpp.

6.2.2.2 OperatorType

```
enum mrock::symbolic_operators::OperatorType
```

Enumerator

Number_Type	
CDW_Type	
SC_Type	
Eta_Type	
Undefined_Type	

Definition at line 22 of file OperatorType.hpp.

6.2.3 Function Documentation

6.2.3.1 char_to_index()

```
constexpr Index mrock::symbolic_operators::char_to_index (
            unsigned char c )  [constexpr]
```

Converts a character to an Index.

This function is designed for lower-case letters but works for any ASCII representable character.

Parameters

c	The character to convert.

Returns

The corresponding Index value.

Definition at line 64 of file IndexWrapper.hpp.

6.2.3.2 clean_up()

```
void mrock::symbolic_operators::clean_up (
            std::vector< Term > & terms )
```

Sorts the terms, adds identical ones together and removes those that are equal to 0.

Parameters

terms	The vector of terms.

Definition at line 767 of file Term.cpp.

6.2.3.3 clean_wicks()

```
void mrock::symbolic_operators::clean_wicks (
            WickTermCollector & terms,
            const std::vector< std::unique_ptr< WickSymmetry >> & symmetries = std::vector<std↩
::unique_ptr<WickSymmetry>>{} )
```

Cleans Wick terms using the provided symmetries.

Parameters

terms	The WickTermCollector containing the terms.
symmetries	The vector of unique pointers to WickSymmetry objects.

Definition at line 137 of file Wick.cpp.

6.2.3.4 clear_duplicates()

```
void mrock::symbolic_operators::clear_duplicates (
            std::vector< Term > & terms )
```

Clears duplicate terms from a vector.

Parameters

terms	The vector of terms.

Definition at line 854 of file Term.cpp.

6.2.3.5 clear_etas()

```
void mrock::symbolic_operators::clear_etas (
            WickTermCollector & terms )
```

Clears eta terms from the WickTermCollector. Intended for use if <eta>=0.

Parameters

terms	The WickTermCollector containing the terms.

Definition at line 118 of file Wick.cpp.

6.2.3.6 commutator() [1/4]

```
std::vector< Term > mrock::symbolic_operators::commutator (
            const std::vector< Term > & left,
            const std::vector< Term > & right )
```

Computes the commutator of two sets of terms: $[A, B] = AB - BA$.

Parameters

left	The left-hand side terms.
right	The right-hand side terms.

Returns

The result of [left, right]

Definition at line 715 of file Term.cpp.

6.2.3.7 commutator() [2/4]

```
std::vector< Term > mrock::symbolic_operators::commutator (
            const std::vector< Term > & left,
            const Term & right )  [inline]
```

Computes the commutator of a set of terms and a term: $[A, B] = AB - BA$.

Parameters

left	The left-hand side terms.
right	The right-hand side term.

Returns

The result of [left, right]

Definition at line 505 of file Term.hpp.

6.2.3.8 commutator() [3/4]

```
std::vector< Term > mrock::symbolic_operators::commutator (
            const Term & left,
            const std::vector< Term > & right )  [inline]
```

Computes the commutator of a term and a set of terms: $[A, B] = AB - BA$.

Parameters

left	The left-hand side term.
right	The right-hand side terms.

Returns

The result of [left, right]

Definition at line 501 of file Term.hpp.

6.2.3.9 commutator() [4/4]

```
std::vector<Term> mrock::symbolic_operators::commutator (
            const Term & left,
            const Term & right )
```

Parameters

left	The left term.
right	The right term.

Returns

The commutation result.

Definition at line 694 of file Term.cpp.

6.2.3.10 hermitian_conjugate()

```
void mrock::symbolic_operators::hermitian_conjugate (
            std::vector< Term > & terms )  [inline]
```

Applies the Hermitian conjugate to a vector of terms.

Parameters

terms	The vector of terms.

Definition at line 509 of file Term.hpp.

6.2.3.11 identify_subexpression()

```
momentum_symbols::value_type mrock::symbolic_operators::identify_subexpression (
            const std::string & sub )  [inline]
```

Definition at line 8 of file Momentum.cpp.

6.2.3.12 identify_wick_operators()

```
WickTermCollector mrock::symbolic_operators::identify_wick_operators (
            const WickTerm & source,
            const std::vector< WickOperatorTemplate > & operator_templates )
```

Identifies Wick operators in a given Wick term.

Parameters

source	The source Wick term.
operator_templates	The vector of Wick operator templates.

Returns

WickTermCollector The collected Wick terms.

Definition at line 64 of file Wick.cpp.

6.2.3.13 is_always_zero() [1/2]

```
bool mrock::symbolic_operators::is_always_zero (
            const std::vector< KroneckerDelta< Index >> & deltas )  [inline]
```

Checks if the vector of KroneckerDelta<Index> objects is always zero.

Parameters

deltas	The vector of KroneckerDelta<Index> objects.

Returns

true if the vector is always zero.

false otherwise.

Definition at line 54 of file KroneckerDeltaUtility.hpp.

6.2.3.14 is_always_zero() [2/2]

```
bool mrock::symbolic_operators::is_always_zero (
            const std::vector< KroneckerDelta< Momentum >> & deltas )  [inline]
```

Checks if the vector of KroneckerDelta<Momentum> objects is always zero.

Parameters

deltas	The vector of KroneckerDelta<Momentum> objects.

Returns

true if the vector is always zero.

false otherwise.

Definition at line 67 of file KroneckerDeltaUtility.hpp.

6.2.3.15 is_mutable()

```
constexpr bool mrock::symbolic_operators::is_mutable (
            const Index idx )  [constexpr]
```

Checks if the given index represents a variable (mutable). 'Mutable' means that it is associated with a sum or similar. An example is sigma; it is commonly summed over as a representation of spins. Then expressions like delta_↩ {sigma,up} can be evaluated to be one if sigma=up. An Index like SpinUp is set to be non-mutable. This allows us to evaluate delta_{up,down}=0.

An index is considered mutable if it is between 2 and 5 (inclusive).

Parameters

idx	The index to check.

Returns

True if the index is mutable, false otherwise.

Definition at line 122 of file IndexWrapper.hpp.

6.2.3.16 make_delta() [1/2]

```
template<typename T >
constexpr auto mrock::symbolic_operators::make_delta (
            const T & first,
            const T & second )  [constexpr]
```

Creates a KroneckerDelta object.

Template Parameters

T	The type of the elements.

Parameters

first	The first element.
second	The second element.

Returns

constexpr auto A KroneckerDelta object.

Definition at line 61 of file KroneckerDelta.hpp.

6.2.3.17 make_delta() [2/2]

```
template<typename T >
constexpr auto mrock::symbolic_operators::make_delta (
            std::decay_t< T > && first,
            std::decay_t< T > && second )  [constexpr]
```

Creates a KroneckerDelta object with rvalue references.

Template Parameters

T	The type of the elements.

Parameters

first	The first element.
second	The second element.

Returns

constexpr auto A KroneckerDelta object.

Definition at line 74 of file KroneckerDelta.hpp.

6.2.3.18 momentum_order()

```
bool mrock::symbolic_operators::momentum_order (
            const Momentum & lhs,
            const Momentum & rhs )
```

Compares two Momentum objects for ordering.

Parameters

lhs	The left-hand side Momentum.
rhs	The right-hand side Momentum.

Returns

True if lhs is ordered before rhs, false otherwise.

Definition at line 176 of file Momentum.cpp.

6.2.3.19 normal_order()

```
void mrock::symbolic_operators::normal_order (
            std::vector< Term > & terms )
```

Parameters

terms	The terms to normal order.

Definition at line 606 of file Term.cpp.

6.2.3.20 operator"!=() [1/7]

```
bool mrock::symbolic_operators::operator!= (
            const Coefficient & lhs,
            const Coefficient & rhs )  [inline]
```

Inequality operator for Coefficient.

Parameters

lhs	The left-hand side Coefficient.
rhs	The right-hand side Coefficient.

Returns

True if the coefficients are not equal, false otherwise.

Definition at line 234 of file Coefficient.hpp.

6.2.3.21 operator"!=() [2/7]

```
template<typename T >
bool mrock::symbolic_operators::operator!= (
            const KroneckerDelta< T > & lhs,
            const KroneckerDelta< T > & rhs )
```

Inequality operator for KroneckerDelta.

Template Parameters

T	The type of the elements.

Parameters

lhs	The left-hand side KroneckerDelta.
rhs	The right-hand side KroneckerDelta.

Returns

true if the two KroneckerDelta objects are not equal.

false otherwise.

Definition at line 104 of file KroneckerDelta.hpp.

6.2.3.22 operator"!=() [3/7]

```
bool mrock::symbolic_operators::operator!= (
            const Operator & lhs,
            const Operator & rhs )  [inline]
```

Inequality operator for Operator.

Parameters

lhs	The left-hand side operator.
rhs	The right-hand side operator.

Returns

True if the operators are not equal, false otherwise.

Definition at line 186 of file Operator.hpp.

6.2.3.23 operator"!=() [4/7]

```
bool mrock::symbolic_operators::operator!= (
            const SumContainer & lhs,
            const SumContainer & rhs )  [inline]
```

Inequality operator for SumContainer.

Parameters

lhs	The left-hand side SumContainer.
rhs	The right-hand side SumContainer.

Returns

True if both SumContainers are not equal, false otherwise.

Definition at line 114 of file SumContainer.hpp.

6.2.3.24 operator"!=() [5/7]

```
bool mrock::symbolic_operators::operator!= (
            const Term & lhs,
            const Term & rhs )  [inline]
```

Checks if two terms are not equal.

Parameters

lhs	The left-hand side term.
rhs	The right-hand side term.

Returns

True if not equal, false otherwise.

Definition at line 412 of file Term.hpp.

6.2.3.25 operator"!=() [6/7]

```
bool mrock::symbolic_operators::operator!= (
            const WickOperator & lhs,
            const WickOperator & rhs )  [inline]
```

Inequality operator for WickOperator.

Parameters

lhs	The left-hand side WickOperator.
rhs	The right-hand side WickOperator.

Returns

true if the two WickOperator objects are not equal.

false otherwise.

Definition at line 554 of file WickTerm.hpp.

6.2.3.26 operator"!=() [7/7]

```
bool mrock::symbolic_operators::operator!= (
            const WickTerm & lhs,
            const WickTerm & rhs )  [inline]
```

Inequality operator for WickTerm.

Parameters

lhs	The left-hand side WickTerm.
rhs	The right-hand side WickTerm.

Returns

true if the two WickTerm objects are not equal.

false otherwise.

Definition at line 565 of file WickTerm.hpp.

6.2.3.27 operator*() [1/2]

```
Momentum mrock::symbolic_operators::operator* (
            const int lhs,
            Momentum rhs )  [inline]
```

Multiplies an integer factor by a Momentum.

Parameters

lhs	The factor.
rhs	The Momentum.

Returns

The result of the multiplication.

Definition at line 290 of file Momentum.hpp.

6.2.3.28 operator∗() [2/2]

```
Momentum mrock::symbolic_operators::operator* (
            Momentum lhs,
            const int rhs )  [inline]
```

Multiplies a Momentum by an integer factor.

Parameters

lhs	The Momentum.
rhs	The factor.

Returns

The result of the multiplication.

Definition at line 279 of file Momentum.hpp.

6.2.3.29 operator+() [1/7]

```
std::string mrock::symbolic_operators::operator+ (
            const MomentumSymbol::name_type sym,
            const std::string & str )  [inline]
```

Concatenates a name_type and a string.

Parameters

sym	The name_type to concatenate.
str	The string to concatenate.

Returns

The concatenated string.

Definition at line 136 of file MomentumSymbol.hpp.

6.2.3.30 operator+() [2/7]

```
std::string mrock::symbolic_operators::operator+ (
            const std::string & str,
            const MomentumSymbol::name_type sym )  [inline]
```

Concatenates a string and a name_type.

Parameters

str	The string to concatenate.
sym	The name_type to concatenate.

Returns

The concatenated string.

Definition at line 126 of file MomentumSymbol.hpp.

6.2.3.31 operator+() [3/7]

```
WickTermCollector mrock::symbolic_operators::operator+ (
            const WickTerm & lhs,
            WickTermCollector rhs )  [inline]
```

Addition operator for WickTerm and WickTermCollector.

Parameters

lhs	The left-hand side WickTerm.
rhs	The right-hand side WickTermCollector.

Returns

WickTermCollector The resulting WickTermCollector.

Definition at line 411 of file WickTerm.hpp.

6.2.3.32 operator+() [4/7]

```
template<typename T >
requires mrock::utility::defines_plus<T>::value KroneckerDelta<T> mrock::symbolic_operators↩
::operator+ (
            KroneckerDelta< T > lhs,
            T const & rhs )  [inline]
```

Addition operator for KroneckerDelta.

Template Parameters

T	The type of the elements.

Parameters

lhs	The left-hand side KroneckerDelta.
rhs	The right-hand side element.

Returns

KroneckerDelta<T> The resulting KroneckerDelta.

Definition at line 147 of file KroneckerDelta.hpp.

6.2.3.33 operator+() [5/7]

```
Momentum mrock::symbolic_operators::operator+ (
            Momentum lhs,
            const Momentum & rhs )  [inline]
```

Adds two Momentum objects.

Parameters

lhs	The left-hand side Momentum.
rhs	The right-hand side Momentum.

Returns

The result of the addition.

Definition at line 257 of file Momentum.hpp.

6.2.3.34 operator+() [6/7]

```
WickTermCollector mrock::symbolic_operators::operator+ (
            WickTermCollector lhs,
            const WickTerm & rhs )  [inline]
```

Addition operator for WickTermCollector and WickTerm.

Parameters

lhs	The left-hand side WickTermCollector.
rhs	The right-hand side WickTerm.

Returns

WickTermCollector The resulting WickTermCollector.

Definition at line 393 of file WickTerm.hpp.

6.2.3.35 operator+() [7/7]

```
WickTermCollector mrock::symbolic_operators::operator+ (
            WickTermCollector lhs,
            const WickTermCollector & rhs )  [inline]
```

Addition operator for two WickTermCollector objects.

Parameters

lhs	The left-hand side WickTermCollector.
rhs	The right-hand side WickTermCollector.

Returns

WickTermCollector The resulting WickTermCollector.

Definition at line 429 of file WickTerm.hpp.

6.2.3.36 operator+=() [1/3]

```
template<typename T >
requires mrock::utility::defines_plus<T>::value KroneckerDelta<T>& mrock::symbolic_operators↩
::operator+= (
            KroneckerDelta< T > & lhs,
            T & rhs )  [inline]
```

Addition assignment operator for KroneckerDelta.

Template Parameters

T	The type of the elements.

Parameters

lhs	The left-hand side KroneckerDelta.
rhs	The right-hand side element.

Returns

KroneckerDelta<T>& The updated KroneckerDelta.

Definition at line 117 of file KroneckerDelta.hpp.

6.2.3.37 operator+=() [2/3]

```
WickTermCollector & mrock::symbolic_operators::operator+= (
            WickTermCollector & lhs,
            const WickTerm & rhs )
```

Addition assignment operator for WickTermCollector and WickTerm.

Parameters

lhs	The left-hand side WickTermCollector.
rhs	The right-hand side WickTerm.

Returns

WickTermCollector& The updated WickTermCollector.

Definition at line 645 of file WickTerm.cpp.

6.2.3.38 operator+=() [3/3]

```
WickTermCollector & mrock::symbolic_operators::operator+= (
            WickTermCollector & lhs,
            const WickTermCollector & rhs )
```

Addition assignment operator for two WickTermCollector objects.

Parameters

lhs	The left-hand side WickTermCollector.
rhs	The right-hand side WickTermCollector.

Returns

WickTermCollector& The updated WickTermCollector.

Definition at line 671 of file WickTerm.cpp.

6.2.3.39 operator-() [1/6]

```
WickTermCollector mrock::symbolic_operators::operator- (
            const WickTerm & lhs,
            WickTermCollector rhs )  [inline]
```

Subtraction operator for WickTerm and WickTermCollector.

Parameters

lhs	The left-hand side WickTerm.
rhs	The right-hand side WickTermCollector.

Returns

WickTermCollector The resulting WickTermCollector.

Definition at line 420 of file WickTerm.hpp.

6.2.3.40 operator-() [2/6]

```
template<typename T >
requires mrock::utility::defines_minus<T>::value KroneckerDelta<T> mrock::symbolic_operators↩
::operator- (
            KroneckerDelta< T > lhs,
            T const & rhs )  [inline]
```

Subtraction operator for KroneckerDelta.

Template Parameters

T	The type of the elements.

Parameters

lhs	The left-hand side KroneckerDelta.
rhs	The right-hand side element.

Returns

KroneckerDelta<T> The resulting KroneckerDelta.

Definition at line 160 of file KroneckerDelta.hpp.

6.2.3.41 operator-() [3/6]

```
Momentum mrock::symbolic_operators::operator- (
            Momentum lhs,
            const Momentum & rhs )  [inline]
```

Subtracts one Momentum from another.

Parameters

lhs	The left-hand side Momentum.
rhs	The right-hand side Momentum.

Returns

The result of the subtraction.

Definition at line 268 of file Momentum.hpp.

6.2.3.42 operator-() [4/6]

```
Momentum mrock::symbolic_operators::operator- (
            Momentum rhs )  [inline]
```

Negates a Momentum.

Parameters

rhs	The Momentum.

Returns

The negated Momentum.

Definition at line 300 of file Momentum.hpp.

6.2.3.43 operator-() [5/6]

```
WickTermCollector mrock::symbolic_operators::operator- (
            WickTermCollector lhs,
            const WickTerm & rhs )  [inline]
```

Subtraction operator for WickTermCollector and WickTerm.

Parameters

lhs	The left-hand side WickTermCollector.
rhs	The right-hand side WickTerm.

Returns

WickTermCollector The resulting WickTermCollector.

Definition at line 402 of file WickTerm.hpp.

6.2.3.44 operator-() [6/6]

```
WickTermCollector mrock::symbolic_operators::operator- (
            WickTermCollector lhs,
            const WickTermCollector & rhs )  [inline]
```

Subtraction operator for two WickTermCollector objects.

Parameters

lhs	The left-hand side WickTermCollector.
rhs	The right-hand side WickTermCollector.

Returns

WickTermCollector The resulting WickTermCollector.

Definition at line 438 of file WickTerm.hpp.

6.2.3.45 operator-=() [1/3]

```
template<typename T >
requires mrock::utility::defines_minus<T>::value KroneckerDelta<T>& mrock::symbolic_operators↩
::operator-= (
            KroneckerDelta< T > & lhs,
            const T & rhs )  [inline]
```

Subtraction assignment operator for KroneckerDelta.

Template Parameters

T	The type of the elements.

Parameters

lhs	The left-hand side KroneckerDelta.
rhs	The right-hand side element.

Returns

KroneckerDelta<T>& The updated KroneckerDelta.

Definition at line 132 of file KroneckerDelta.hpp.

6.2.3.46 operator-=() [2/3]

```
WickTermCollector & mrock::symbolic_operators::operator-= (
            WickTermCollector & lhs,
            const WickTerm & rhs )
```

Subtraction assignment operator for WickTermCollector and WickTerm.

Parameters

lhs	The left-hand side WickTermCollector.
rhs	The right-hand side WickTerm.

Returns

WickTermCollector& The updated WickTermCollector.

Definition at line 658 of file WickTerm.cpp.

6.2.3.47 operator-=() [3/3]

```
WickTermCollector & mrock::symbolic_operators::operator-= (
            WickTermCollector & lhs,
            const WickTermCollector & rhs )
```

Subtraction assignment operator for two WickTermCollector objects.

Parameters

lhs	The left-hand side WickTermCollector.
rhs	The right-hand side WickTermCollector.

Returns

WickTermCollector& The updated WickTermCollector.

Definition at line 678 of file WickTerm.cpp.

6.2.3.48 operator<()

```
bool mrock::symbolic_operators::operator< (
            const Momentum & lhs,
            const Momentum & rhs )
```

Compares two Momentum objects for less-than ordering.

Parameters

lhs	The left-hand side Momentum.
rhs	The right-hand side Momentum.

Returns

True if lhs is less than rhs, false otherwise.

Definition at line 228 of file Momentum.cpp.

6.2.3.49 operator<<() [1/19]

```
std::ostream & mrock::symbolic_operators::operator<< (
            std::ostream & os,
            const Coefficient & coeff )
```

Outputs a coefficient to a stream.

Parameters

os	The output stream.
coeff	The coefficient.

Returns

The output stream.

Definition at line 76 of file Coefficient.cpp.

6.2.3.50 operator<<() [2/19]

```
std::ostream & mrock::symbolic_operators::operator<< (
            std::ostream & os,
            const Index index )
```

Overloads the stream insertion operator for the Index enum.

Parameters

os	The output stream.
index	The Index value to insert into the stream.

Returns

The output stream.

Definition at line 5 of file IndexWrapper.cpp.

6.2.3.51 operator<<() [3/19]

```
std::ostream & mrock::symbolic_operators::operator<< (
            std::ostream & os,
            const IndexWrapper & indizes )
```

Overloads the stream insertion operator for the IndexWrapper struct.

Parameters

os	The output stream.
indizes	The IndexWrapper to insert into the stream.

Returns

The output stream.

Definition at line 48 of file IndexWrapper.cpp.

6.2.3.52 operator<<() [4/19]

```
template<typename T >
std::ostream& mrock::symbolic_operators::operator<< (
            std::ostream & os,
            const KroneckerDelta< T > & delta )  [inline]
```

Stream insertion operator for KroneckerDelta.

Template Parameters

T	The type of the elements.

Parameters

os	The output stream.
delta	The KroneckerDelta object.

Returns

std::ostream& The updated output stream.

Definition at line 173 of file KroneckerDelta.hpp.

6.2.3.53 operator<<() [5/19]

```
std::ostream & mrock::symbolic_operators::operator<< (
            std::ostream & os,
            const Momentum & momentum )
```

Outputs a Momentum to an output stream.

Parameters

os	The output stream.
momentum	The Momentum.

Returns

The output stream.

Definition at line 190 of file Momentum.cpp.

6.2.3.54 operator<<() [6/19]

```
std::ostream & mrock::symbolic_operators::operator<< (
            std::ostream & os,
            const MomentumList & momenta )
```

Outputs the MomentumList to an output stream.

Parameters

os	The output stream.
momenta	The MomentumList to output.

Returns

The output stream.

Definition at line 22 of file MomentumList.cpp.

6.2.3.55 operator<<() [7/19]

```
std::ostream& mrock::symbolic_operators::operator<< (
            std::ostream & os,
            const MomentumSymbol::name_type name )  [inline]
```

Outputs the name_type to an output stream.

Parameters

os	The output stream.
name	The name_type to output.

Returns

The output stream.

Definition at line 110 of file MomentumSymbol.hpp.

6.2.3.56 operator<<() [8/19]

```
std::ostream & mrock::symbolic_operators::operator<< (
            std::ostream & os,
            const Operator & op )
```

Stream insertion operator for Operator.

Parameters

os	The output stream.
op	The operator to insert into the stream.

Returns

The output stream.

Definition at line 4 of file Operator.cpp.

6.2.3.57 operator<<() [9/19]

```
std::ostream & mrock::symbolic_operators::operator<< (
            std::ostream & os,
            const OperatorType op )
```

Overloads the stream insertion operator for the Index enum.

Parameters

os	The output stream.
index	The OperatorType to insert into the stream.

Returns

The output stream.

Definition at line 4 of file OperatorType.cpp.

6.2.3.58 operator<<() [10/19]

```
std::ostream & mrock::symbolic_operators::operator<< (
            std::ostream & os,
            const std::vector< Coefficient > & coeffs )
```

Outputs a vector of coefficients to a stream.

Parameters

os	The output stream.
coeffs	The coefficients.

Returns

The output stream.

Definition at line 88 of file Coefficient.cpp.

6.2.3.59 operator<<() [11/19]

```
std::ostream & mrock::symbolic_operators::operator<< (
            std::ostream & os,
            const std::vector< Operator > & ops )
```

Stream insertion operator for a vector of Operators.

Parameters

os	The output stream.
ops	The vector of operators to insert into the stream.

Returns

The output stream.

Definition at line 18 of file Operator.cpp.

6.2.3.60 operator<<() [12/19]

```
std::ostream & mrock::symbolic_operators::operator<< (
            std::ostream & os,
            const std::vector< Term > & terms )
```

Outputs a vector of terms to a stream.

Parameters

os	The output stream.
terms	The terms.

Returns

The output stream.

Definition at line 754 of file Term.cpp.

6.2.3.61 operator<<() [13/19]

```
std::ostream & mrock::symbolic_operators::operator<< (
            std::ostream & os,
            const std::vector< WickOperator > & ops )
```

Stream insertion operator for a vector of WickOperator objects.

Parameters

os	The output stream.
ops	The vector of WickOperator objects.

Returns

std::ostream& The updated output stream.

Definition at line 43 of file WickOperator.cpp.

6.2.3.62 operator<<() [14/19]

```
std::ostream& mrock::symbolic_operators::operator<< (
            std::ostream & os,
            const SumContainer & sums )  [inline]
```

Stream insertion operator for SumContainer.

Parameters

os	The output stream.
sums	The SumContainer to insert into the stream.

Returns

Reference to the output stream.

Definition at line 124 of file SumContainer.hpp.

6.2.3.63 operator<<() [15/19]

```
std::ostream& mrock::symbolic_operators::operator<< (
            std::ostream & os,
            const Term & term )
```

Parameters

os	The output stream.
term	The Term object to insert into the stream.

Returns

The output stream.

Definition at line 731 of file Term.cpp.

6.2.3.64 operator<<() [16/19]

```
std::ostream & mrock::symbolic_operators::operator<< (
            std::ostream & os,
            const WickOperator & op )
```

Stream insertion operator for WickOperator.

Parameters

os	The output stream.
op	The WickOperator object.

Returns

std::ostream& The updated output stream.

Definition at line 30 of file WickOperator.cpp.

6.2.3.65 operator<<() [17/19]

```
std::ostream & mrock::symbolic_operators::operator<< (
            std::ostream & os,
            const WickTerm & term )
```

Stream insertion operator for WickTerm.

Parameters

os	The output stream.
term	The WickTerm object.

Returns

std::ostream& The updated output stream.

Definition at line 610 of file WickTerm.cpp.

6.2.3.66 operator<<() [18/19]

```
std::ostream & mrock::symbolic_operators::operator<< (
            std::ostream & os,
            const WickTermCollector & terms )
```

Stream insertion operator for WickTermCollector.

Parameters

os	The output stream.
terms	The WickTermCollector object.

Returns

std::ostream& The updated output stream.

Definition at line 633 of file WickTerm.cpp.

6.2.3.67 operator<<() [19/19]

```
template<class SumIndex >
std::ostream& mrock::symbolic_operators::operator<< (
            std::ostream & os,
            SymbolicSum< SumIndex > const & sum )
```

Outputs the SymbolicSum object to an output stream.

Template Parameters

SumIndex	The type of the summation index.

Parameters

os	The output stream.
sum	The SymbolicSum object to output.

Returns

The output stream.

Definition at line 107 of file SymbolicSum.hpp.

6.2.3.68 operator==() [1/7]

```
bool mrock::symbolic_operators::operator== (
            const Coefficient & lhs,
            const Coefficient & rhs )  [inline]
```

Equality operator for Coefficient.

Parameters

lhs	The left-hand side Coefficient.
rhs	The right-hand side Coefficient.

Returns

True if the coefficients are equal, false otherwise.

Definition at line 221 of file Coefficient.hpp.

6.2.3.69 operator==() [2/7]

```
template<typename T >
bool mrock::symbolic_operators::operator== (
            const KroneckerDelta< T > & lhs,
            const KroneckerDelta< T > & rhs )
```

Equality operator for KroneckerDelta.

Template Parameters

T	The type of the elements.

Parameters

lhs	The left-hand side KroneckerDelta.
rhs	The right-hand side KroneckerDelta.

Returns

true if the two KroneckerDelta objects are equal.

false otherwise.

Definition at line 88 of file KroneckerDelta.hpp.

6.2.3.70 operator==() [3/7]

```
bool mrock::symbolic_operators::operator== (
            const Operator & lhs,
            const Operator & rhs )  [inline]
```

Equality operator for Operator.

Parameters

lhs	The left-hand side operator.
rhs	The right-hand side operator.

Returns

True if the operators are equal, false otherwise.

Definition at line 173 of file Operator.hpp.

6.2.3.71 operator==() [4/7]

```
bool mrock::symbolic_operators::operator== (
            const SumContainer & lhs,
            const SumContainer & rhs )  [inline]
```

Equality operator for SumContainer.

Parameters

lhs	The left-hand side SumContainer.
rhs	The right-hand side SumContainer.

Returns

True if both SumContainers are equal, false otherwise.

Definition at line 104 of file SumContainer.hpp.

6.2.3.72 operator==() [5/7]

```
bool mrock::symbolic_operators::operator== (
            const Term & lhs,
            const Term & rhs )  [inline]
```

Checks if two terms are equal.

Parameters

lhs	The left-hand side term.
rhs	The right-hand side term.

Returns

True if equal, false otherwise.

Definition at line 404 of file Term.hpp.

6.2.3.73 operator==() [6/7]

```
bool mrock::symbolic_operators::operator== (
            const WickOperator & lhs,
            const WickOperator & rhs )  [inline]
```

Equality operator for WickOperator.

Parameters

lhs	The left-hand side WickOperator.
rhs	The right-hand side WickOperator.

Returns

true if the two WickOperator objects are equal.

false otherwise.

Definition at line 548 of file WickTerm.hpp.

6.2.3.74 operator==() [7/7]

```
bool mrock::symbolic_operators::operator== (
            const WickTerm & lhs,
            const WickTerm & rhs )  [inline]
```

Equality operator for WickTerm.

Parameters

lhs	The left-hand side WickTerm.
rhs	The right-hand side WickTerm.

Returns

true if the two WickTerm objects are equal.

false otherwise.

Definition at line 557 of file WickTerm.hpp.

6.2.3.75 operator>()

```
bool mrock::symbolic_operators::operator> (
            const Momentum & lhs,
            const Momentum & rhs )
```

Compares two Momentum objects for greater-than ordering.

Parameters

lhs	The left-hand side Momentum.
rhs	The right-hand side Momentum.

Returns

True if lhs is greater than rhs, false otherwise.

Definition at line 220 of file Momentum.cpp.

6.2.3.76 operator>>()

```
std::istream& mrock::symbolic_operators::operator>> (
            std::istream & is,
            MomentumSymbol::name_type & name )  [inline]
```

Inputs a name_type from an input stream.

Parameters

is	The input stream.
name	The name_type to input.

Returns

The input stream.

Definition at line 118 of file MomentumSymbol.hpp.

6.2.3.77 prepare_wick()

```
WickTermCollector mrock::symbolic_operators::prepare_wick (
            const std::vector< Term > & terms )
```

Definition at line 44 of file Wick.cpp.

6.2.3.78 remove_delta_is_one()

```
template<class T >
void mrock::symbolic_operators::remove_delta_is_one (
            std::vector< KroneckerDelta< T >> & deltas )
```

Removes KroneckerDelta objects that are one from the vector. Note that delta_{a,a} = 1.

Template Parameters

T	The type of the elements.

Parameters

deltas	The vector of KroneckerDelta objects.

Definition at line 40 of file KroneckerDeltaUtility.hpp.

6.2.3.79 remove_delta_squared()

```
template<class T >
void mrock::symbolic_operators::remove_delta_squared (
            std::vector< KroneckerDelta< T >> & deltas )
```

Removes squared KroneckerDelta objects from the vector. Note that delta_{a,b}^N = delta_{a,b}.

Template Parameters

T	The type of the elements.

Parameters

deltas	The vector of KroneckerDelta objects.

Definition at line 21 of file KroneckerDeltaUtility.hpp.

6.2.3.80 remove_double_occurances()

```
void mrock::symbolic_operators::remove_double_occurances (
            KroneckerDelta< Momentum > & delta )  [inline]
```

Removes double occurrences in a KroneckerDelta<Momentum> object.

Parameters

delta	The KroneckerDelta<Momentum> object.

Definition at line 78 of file KroneckerDeltaUtility.hpp.

6.2.3.81 rename_momenta()

```
void mrock::symbolic_operators::rename_momenta (
            std::vector< Term > & terms,
            const MomentumSymbol::name_type what,
            const MomentumSymbol::name_type to )  [inline]
```

Renames momenta in a vector of terms.

Parameters

terms	The vector of terms.
what	The momentum to rename.
to	The new momentum.

Definition at line 514 of file Term.hpp.

6.2.3.82 to_string_without_prefactor()

```
std::string mrock::symbolic_operators::to_string_without_prefactor (
            const std::vector< Term > & terms )
```

Converts a vector of terms to a string without the prefactor.

Parameters

terms	The vector of terms.

Returns

The string representation.

Definition at line 880 of file Term.cpp.

6.2.3.83 wick_processor()

```
void mrock::symbolic_operators::wick_processor (
            const std::vector< Operator > & remaining,
            WickTermCollector & reciever_list,
            std::variant< WickTerm, Term > buffer )
```

Definition at line 9 of file Wick.cpp.

6.2.3.84 wicks_theorem()

```
void mrock::symbolic_operators::wicks_theorem (
            const std::vector< Term > & terms,
            const std::vector< WickOperatorTemplate > & operator_templates,
            WickTermCollector & reciever )
```

Applies Wick's theorem to a set of terms.

Parameters

terms	The vector of terms.
operator_templates	The vector of Wick operator templates.
reciever	The WickTermCollector to receive the results.

Definition at line 107 of file Wick.cpp.

6.2.4 Variable Documentation

6.2.4.1 string_to_index

```
mrock::symbolic_operators::string_to_index  [inline]
```

A map that associates string representations with their corresponding Index values.

Definition at line 72 of file IndexWrapper.hpp.

6.2.4.2 string_to_wick

```
mrock::symbolic_operators::string_to_wick  [inline]
```

Initial value:

```
= {
        {"n", Number_Type}, {"g", CDW_Type}, {"f", SC_Type}, {"\\eta", Eta_Type}
    }
```

A map that associates string representations with their corresponding Wick operators values.

Definition at line 28 of file OperatorType.hpp.

6.3 sym_op_test Namespace Reference

Classes

- struct SymOpTest

Chapter 7

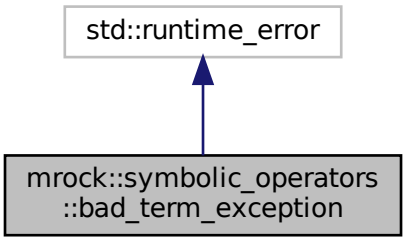
Class Documentation

7.1 mrock::symbolic_operators::bad_term_exception Class Reference

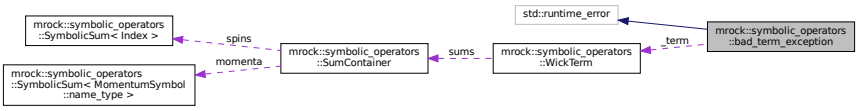
An exception class for bad terms.

`#include <WickTerm.hpp>`

Inheritance diagram for mrock::symbolic_operators::bad_term_exception:



Collaboration diagram for mrock::symbolic_operators::bad_term_exception:



Public Member Functions

- bad_term_exception (const std::string &what_arg, const WickTerm &term)

  Constructs a bad_term_exception object.
- bad_term_exception (const char ∗what_arg, const WickTerm &term)

  Constructs a bad_term_exception object.
- const WickTerm & which_term () const noexcept

  Returns the bad term.

Protected Attributes

- const WickTerm _term

  The bad term.

7.1.1 Detailed Description

An exception class for bad terms.

Definition at line 462 of file WickTerm.hpp.

7.1.2 Constructor & Destructor Documentation

7.1.2.1 bad_term_exception() [1/2]

```
mrock::symbolic_operators::bad_term_exception::bad_term_exception (
            const std::string & what_arg,
            const WickTerm & term )  [inline]
```

Constructs a bad_term_exception object.

Parameters

what_arg	The error message.
term	The bad term.

Definition at line 473 of file WickTerm.hpp.

7.1.2.2 bad_term_exception() [2/2]

```
mrock::symbolic_operators::bad_term_exception::bad_term_exception (
            const char * what_arg,
            const WickTerm & term )  [inline]
```

Constructs a bad_term_exception object.

Parameters

what_arg	The error message.
term	The bad term.

Definition at line 481 of file WickTerm.hpp.

7.1.3 Member Function Documentation

7.1.3.1 which_term()

```
const WickTerm& mrock::symbolic_operators::bad_term_exception::which_term ( ) const  [inline],
[noexcept]
```

Returns the bad term.

Returns

const WickTerm& The bad term.

Definition at line 488 of file WickTerm.hpp.

7.1.4 Member Data Documentation

7.1.4.1 _term

```
const WickTerm mrock::symbolic_operators::bad_term_exception::_term  [protected]
```

The bad term.

Definition at line 464 of file WickTerm.hpp.

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

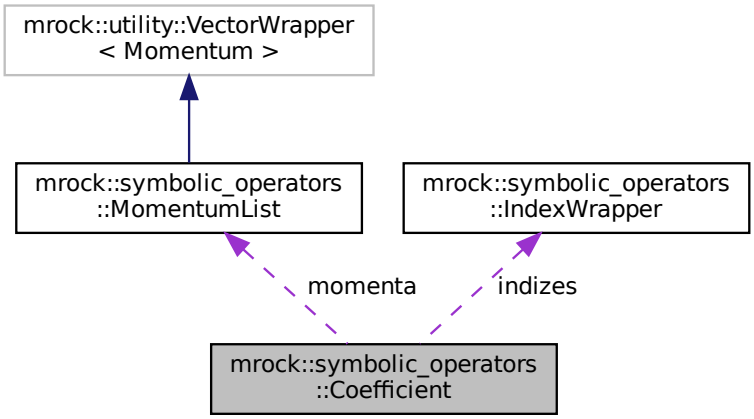
- include/mrock/symbolic_operators/WickTerm.hpp

7.2 mrock::symbolic_operators::Coefficient Struct Reference

Represents a coefficient. Various symmetries are pre defined (e.g. inversion symmetry) and can be toggled on or off A custom symmetry can also be provided.

`#include <Coefficient.hpp>`

Collaboration diagram for mrock::symbolic_operators::Coefficient:



Public Member Functions

- template<class Archive >
  void serialize (Archive &ar, const unsigned int version)

  Serializes the Coefficient object.
- Coefficient ()=default

  Default constructor.
- Coefficient (const std::string &_name)

  Constructs a Coefficient with a given name.
- Coefficient (const std::string &_name, const Momentum &_momentum, const IndexWrapper &_indizes, bool _Q_changes_sign=false, bool _inversion_symmetry=true, bool _is_daggered=false)

  Constructs a Coefficient with a given name, a single momentum, and a set of indizes (can be of size 1)
- Coefficient (const std::string &_name, const Momentum &_momentum, bool _Q_changes_sign=false, bool _inversion_symmetry=true, bool _is_daggered=false)

  Constructs a Coefficient with a given name, and a single momentum.
- Coefficient (const std::string &_name, const MomentumList &_momenta, const IndexWrapper &_↩ indizes=IndexWrapper{}, bool _Q_changes_sign=false, bool _inversion_symmetry=true, bool _is_↩ daggered=false)

  Constructs a Coefficient with a given name, multiple momenta, and a set of indizes (can be of size 1)
- bool uses_index (const Index index) const noexcept

  Checks if the coefficient uses a specific index.
- bool depends_on_momentum () const noexcept

  Checks if the coefficient depends on momentum.
- bool depends_on (const MomentumSymbol::name_type momentum) const noexcept

Checks if the coefficient depends on a specific momentum.

- bool depends_on_two_momenta () const noexcept

  Checks if the coefficient depends on two momenta, e.g, k-l. The Current implementation is restricted to a MomentumList of size 1, i.e., V(k) or V(k-l) but not V(k, l)
- Coefficient & hermitian_conjugate_inplace ()

  Toggles the daggered state of the operator.
- Coefficient hermitian_conjugate () const

  Creates hermitian conjugate of this as a new object.
- void invert_momentum (const MomentumSymbol::name_type what)

  Inverts the momentum of the coefficient.
- void use_symmetric_interaction_exchange ()

  Utilizes V(k, k', q) = V(k', k, -q) symmetry.
- void use_symmetric_interaction_inversion ()

  Utilizes V(k, k', q) = V(-k, -k', -q) symmetry.
- void remove_momentum_contribution (const MomentumSymbol::name_type value)

  Removes a momentum contribution from the coefficient.
- void apply_custom_symmetry ()

  Applies the custom symmetry function if it exists.

Static Public Member Functions

- static Coefficient RealInversionSymmetric (const std::string &name, const MomentumList &momenta, const std::optional< std::function< void(Coefficient &)>> &custom_symmetry=std::nullopt)

  Generates a real and inversion symmetric coefficient.
- static Coefficient RealInteraction (const std::string &name, const MomentumList &momenta, const std::optional< std::function< void(Coefficient &)>> &custom_symmetry=std::nullopt)

  Generates a real Coefficient with V(k, k', q) = V(k', k, -q).
- static Coefficient HoneyComb (const std::string &name, const Momentum &momentum, bool daggered, bool is_real=true, const std::optional< std::function< void(Coefficient &)>> &custom_symmetry=std::nullopt)

  Generates a Coefficient as they occur on a honeycomb lattice.
- static Coefficient Constant (const std::string &name, const IndexWrapper &indizes=IndexWrapper{}, bool is_real=true, bool is_daggered=false)

  Generates a Coefficient that does not depend on any momentum.
- static Coefficient parse_string (const std::string &expression, bool _Q_changes_sign=false, bool _inversion_symmetry=true)

  Parses a string to create a Coefficient.
- static Coefficient parse_interaction_string (const std::string &expression)

  Parses a string to create a standard interaction Coefficient.

Public Attributes

- std::string name

  Name of the coefficient.
- MomentumList momenta

  List of momenta associated with the coefficient.
- IndexWrapper indizes

  Contains all indices, standard: first index = spin, all others arbitrary, e.g., orbitals, bands, etc.
- std::optional< std::function< void(Coefficient &)> > custom_symmetry = std::nullopt

  Optional custom symmetry function.
- bool inversion_symmetry { true }

Indicates if V(k) = V(-k).

- bool is_symmetrized_interaction { }

  Indicates if the interaction is symmetrized, i.e., V(k, k', q) = V(k', k, -q)

- bool Q_changes_sign {}

  Indicates if V(k+Q) = -V(k).

- bool is_real { true }

  Indicates if V^* = V. Default is true.

- bool is_daggered {}

  Indicates if the coefficient is daggered.

7.2.1 Detailed Description

Represents a coefficient. Various symmetries are pre defined (e.g. inversion symmetry) and can be toggled on or off A custom symmetry can also be provided.

Definition at line 21 of file Coefficient.hpp.

7.2.2 Constructor & Destructor Documentation

7.2.2.1 Coefficient() [1/5]

```
mrock::symbolic_operators::Coefficient::Coefficient ( )  [default]
```

Default constructor.

7.2.2.2 Coefficient() [2/5]

```
mrock::symbolic_operators::Coefficient::Coefficient (
            const std::string & _name )  [explicit]
```

Constructs a Coefficient with a given name.

Parameters

_name	The name of the coefficient

Definition at line 96 of file Coefficient.cpp.

7.2.2.3 Coefficient() [3/5]

```
mrock::symbolic_operators::Coefficient::Coefficient (
            const std::string & _name,
```

```
            const Momentum & _momentum,
            const IndexWrapper & _indizes,
            bool _Q_changes_sign = false,
            bool _inversion_symmetry = true,
            bool _is_daggered = false )
```

Constructs a Coefficient with a given name, a single momentum, and a set of indizes (can be of size 1)

Parameters

_name	The name of the coefficient
_momentum	The momentum of the coefficient
_indizes	The indizes of the coefficient
_Q_changes_sign	Toggles the V(k+Q) = -V(k) symmetry. Default is false.
_inversion_symmetry	Toggles inversion symmetry, V(k) = V(-k). Default is true.
_is_daggered	Toggles whether the coefficient is a complex conjugate or not. Default is false.

Definition at line 103 of file Coefficient.cpp.

7.2.2.4 Coefficient() [4/5]

```
mrock::symbolic_operators::Coefficient::Coefficient (
            const std::string & _name,
            const Momentum & _momentum,
            bool _Q_changes_sign = false,
            bool _inversion_symmetry = true,
            bool _is_daggered = false )
```

Constructs a Coefficient with a given name, and a single momentum.

Parameters

_name	The name of the coefficient
_momentum	The momentum of the coefficient
_Q_changes_sign	Toggles the V(k+Q) = -V(k) symmetry. Default is false.
_inversion_symmetry	Toggles inversion symmetry, V(k) = V(-k). Default is true.
_is_daggered	Toggles whether the coefficient is a complex conjugate or not. Default is false.

Definition at line 111 of file Coefficient.cpp.

7.2.2.5 Coefficient() [5/5]

```
mrock::symbolic_operators::Coefficient::Coefficient (
            const std::string & _name,
            const MomentumList & _momenta,
            const IndexWrapper & _indizes = IndexWrapper{},
```

```
            bool _Q_changes_sign = false,
            bool _inversion_symmetry = true,
            bool _is_daggered = false )
```

Constructs a Coefficient with a given name, multiple momenta, and a set of indizes (can be of size 1)

Parameters

_name	The name of the coefficient
_momenta	The momenta of the coefficient, in order. Usually occurs for interactions, i.e., V(k, k', q)
_indizes	The indizes of the coefficient
_Q_changes_sign	Toggles the V(k+Q) = -V(k) symmetry. Default is false.
_inversion_symmetry	Toggles inversion symmetry, V(k) = V(-k). Default is true.
_is_daggered	Toggles whether the coefficient is a complex conjugate or not. Default is false.

Definition at line 119 of file Coefficient.cpp.

7.2.3 Member Function Documentation

7.2.3.1 apply_custom_symmetry()

```
void mrock::symbolic_operators::Coefficient::apply_custom_symmetry ( )
```

Applies the custom symmetry function if it exists.

Definition at line 40 of file Coefficient.cpp.

7.2.3.2 Constant()

```
Coefficient mrock::symbolic_operators::Coefficient::Constant (
            const std::string & name,
            const IndexWrapper & indizes = IndexWrapper{},
            bool is_real = true,
            bool is_daggered = false )  [static]
```

Generates a Coefficient that does not depend on any momentum.

Parameters

name	The name of the coefficient.
indizes	The indizes of the coefficient. Default is no index.
is_daggered	Indicates if the coefficient is daggered. Default is false.
is_real	Indicates if the coefficient is real. Default is true.

Returns

A Coefficient with the symmetries of a honeycomb lattice.

Definition at line 153 of file Coefficient.cpp.

7.2.3.3 depends_on()

```
bool mrock::symbolic_operators::Coefficient::depends_on (
            const MomentumSymbol::name_type momentum ) const  [inline], [noexcept]
```

Checks if the coefficient depends on a specific momentum.

Parameters

momentum	The momentum to check.

Returns

True if it depends on the momentum, false otherwise.

Definition at line 249 of file Coefficient.hpp.

7.2.3.4 depends_on_momentum()

```
bool mrock::symbolic_operators::Coefficient::depends_on_momentum ( ) const  [inline], [noexcept]
```

Checks if the coefficient depends on momentum.

Returns

True if it depends on momentum, false otherwise.

Definition at line 243 of file Coefficient.hpp.

7.2.3.5 depends_on_two_momenta()

```
bool mrock::symbolic_operators::Coefficient::depends_on_two_momenta ( ) const  [inline], [noexcept]
```

Checks if the coefficient depends on two momenta, e.g, k-l. The Current implementation is restricted to a MomentumList of size 1, i.e., V(k) or V(k-l) but not V(k, l)

Returns

True if it depends on two momenta, false otherwise.

Definition at line 257 of file Coefficient.hpp.

7.2.3.6 hermitian_conjugate()

```
Coefficient mrock::symbolic_operators::Coefficient::hermitian_conjugate ( ) const  [inline]
```

Creates hermitian conjugate of this as a new object.

Returns

Returns the new object.

Definition at line 269 of file Coefficient.hpp.

7.2.3.7 hermitian_conjugate_inplace()

```
Coefficient & mrock::symbolic_operators::Coefficient::hermitian_conjugate_inplace ( )  [inline]
```

Toggles the daggered state of the operator.

Returns

A reference to ∗this

Definition at line 261 of file Coefficient.hpp.

7.2.3.8 HoneyComb()

```
Coefficient mrock::symbolic_operators::Coefficient::HoneyComb (
            const std::string & name,
            const Momentum & momentum,
            bool daggered,
            bool is_real = true,
            const std::optional< std::function< void(Coefficient &)>> & custom_symmetry =
std::nullopt )  [static]
```

Generates a Coefficient as they occur on a honeycomb lattice.

Parameters

name	The name of the coefficient.
momentum	The momentum.
daggered	Indicates if the coefficient is daggered.
is_real	Indicates if the coefficient is real. Default is true.
custom_symmetry	Optional custom symmetry function.

Returns

A Coefficient with the symmetries of a honeycomb lattice.

Definition at line 144 of file Coefficient.cpp.

7.2.3.9 invert_momentum()

```
void mrock::symbolic_operators::Coefficient::invert_momentum (
            const MomentumSymbol::name_type what )
```

Inverts the momentum of the coefficient.

Parameters

what	The momentum to invert.

Definition at line 5 of file Coefficient.cpp.

7.2.3.10 parse_interaction_string()

```
Coefficient mrock::symbolic_operators::Coefficient::parse_interaction_string (
            const std::string & expression )  [static]
```

Parses a string to create a standard interaction Coefficient.

Parameters

expression	The string expression.

Returns

A parsed interaction Coefficient.

Definition at line 70 of file Coefficient.cpp.

7.2.3.11 parse_string()

```
Coefficient mrock::symbolic_operators::Coefficient::parse_string (
            const std::string & expression,
            bool _Q_changes_sign = false,
            bool _inversion_symmetry = true )  [static]
```

Parses a string to create a Coefficient.

Parameters

expression	The string expression.
_Q_changes_sign	Indicates if Q changes sign.
_inversion_symmetry	Indicates if inversion symmetry is present.

Returns

A parsed Coefficient.

Definition at line 47 of file Coefficient.cpp.

7.2.3.12 RealInteraction()

```
Coefficient mrock::symbolic_operators::Coefficient::RealInteraction (
            const std::string & name,
            const MomentumList & momenta,
            const std::optional< std::function< void(Coefficient &)>> & custom_symmetry =
std::nullopt )  [static]
```

Generates a real Coefficient with V(k, k', q) = V(k', k, -q).

Parameters

name	The name of the coefficient.
momenta	The list of momenta.
custom_symmetry	Optional custom symmetry function.

Returns

A real interaction Coefficient.

Definition at line 134 of file Coefficient.cpp.

7.2.3.13 RealInversionSymmetric()

```
Coefficient mrock::symbolic_operators::Coefficient::RealInversionSymmetric (
            const std::string & name,
            const MomentumList & momenta,
            const std::optional< std::function< void(Coefficient &)>> & custom_symmetry =
std::nullopt )  [static]
```

Generates a real and inversion symmetric coefficient.

Parameters

_name	The name of the coefficient.
_momenta	The list of momenta.
_custom_symmetry	Optional custom symmetry function.

Returns

A real and inversion symmetric Coefficient.

Definition at line 127 of file Coefficient.cpp.

7.2.3.14 remove_momentum_contribution()

```
void mrock::symbolic_operators::Coefficient::remove_momentum_contribution (
            const MomentumSymbol::name_type value )
```

Removes a momentum contribution from the coefficient.

Parameters

value	The momentum value to remove.

Definition at line 33 of file Coefficient.cpp.

7.2.3.15 serialize()

```
template<class Archive >
void mrock::symbolic_operators::Coefficient::serialize (
            Archive & ar,
            const unsigned int version )  [inline]
```

Serializes the Coefficient object.

Template Parameters

Archive	The archive type.

Parameters

ar	The archive object.
version	The version of the serialization.

Definition at line 39 of file Coefficient.hpp.

7.2.3.16 use_symmetric_interaction_exchange()

```
void mrock::symbolic_operators::Coefficient::use_symmetric_interaction_exchange ( )
```

Utilizes V(k, k', q) = V(k', k, -q) symmetry.

Definition at line 16 of file Coefficient.cpp.

7.2.3.17 use_symmetric_interaction_inversion()

```
void mrock::symbolic_operators::Coefficient::use_symmetric_interaction_inversion ( )
```

Utilizes V(k, k', q) = V(-k, -k', -q) symmetry.

Definition at line 24 of file Coefficient.cpp.

7.2.3.18 uses_index()

```
bool mrock::symbolic_operators::Coefficient::uses_index (
            const Index index ) const  [inline], [noexcept]
```

Checks if the coefficient uses a specific index.

Parameters

index	The index to check.

Returns

True if the index is used, false otherwise.

Definition at line 237 of file Coefficient.hpp.

7.2.4 Member Data Documentation

7.2.4.1 custom_symmetry

```
std::optional<std::function<void(Coefficient&)> > mrock::symbolic_operators::Coefficient↩
::custom_symmetry = std::nullopt
```

Optional custom symmetry function.

Definition at line 25 of file Coefficient.hpp.

7.2.4.2 indizes

```
IndexWrapper mrock::symbolic_operators::Coefficient::indizes
```

Contains all indices, standard: first index = spin, all others arbitrary, e.g., orbitals, bands, etc.

Definition at line 24 of file Coefficient.hpp.

7.2.4.3 inversion_symmetry

```
bool mrock::symbolic_operators::Coefficient::inversion_symmetry { true }
```

Indicates if V(k) = V(-k).

Definition at line 26 of file Coefficient.hpp.

7.2.4.4 is_daggered

```
bool mrock::symbolic_operators::Coefficient::is_daggered {}
```

Indicates if the coefficient is daggered.

Definition at line 30 of file Coefficient.hpp.

7.2.4.5 is_real

```
bool mrock::symbolic_operators::Coefficient::is_real { true }
```

Indicates if $V^{\wedge}* = V$. Default is true.

Definition at line 29 of file Coefficient.hpp.

7.2.4.6 is_symmetrized_interaction

```
bool mrock::symbolic_operators::Coefficient::is_symmetrized_interaction { }
```

Indicates if the interaction is symmetrized, i.e., V(k, k', q) = V(k', k, -q)

Definition at line 27 of file Coefficient.hpp.

7.2.4.7 momenta

```
MomentumList mrock::symbolic_operators::Coefficient::momenta
```

List of momenta associated with the coefficient.

Definition at line 23 of file Coefficient.hpp.

7.2.4.8 name

```
std::string mrock::symbolic_operators::Coefficient::name
```

Name of the coefficient.

Definition at line 22 of file Coefficient.hpp.

7.2.4.9 Q_changes_sign

```
bool mrock::symbolic_operators::Coefficient::Q_changes_sign {}
```

Indicates if V(k+Q) = -V(k).

Definition at line 28 of file Coefficient.hpp.

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

- include/mrock/symbolic_operators/Coefficient.hpp
- sources/Coefficient.cpp

7.3 mrock::symbolic_operators::IndexComparison Struct Reference

A structure for comparing indices. E.g. <n_k> merely requires that the spin indizes of the composing operators are identical, but <f_k> requires the first index to be spin down.

```
#include <WickOperatorTemplate.hpp>
```

Public Attributes

- bool any_identical

  Indicates if any identical indices are identical.
- Index base { Index::UndefinedIndex }

  The base index.
- Index other { Index::UndefinedIndex }

  The other index.

7.3.1 Detailed Description

A structure for comparing indices. E.g. <n_k> merely requires that the spin indizes of the composing operators are identical, but <f_k> requires the first index to be spin down.

Definition at line 21 of file WickOperatorTemplate.hpp.

7.3.2 Member Data Documentation

7.3.2.1 any_identical

```
bool mrock::symbolic_operators::IndexComparison::any_identical
```

Indicates if any identical indices are identical.

Definition at line 22 of file WickOperatorTemplate.hpp.

7.3.2.2 base

```
Index mrock::symbolic_operators::IndexComparison::base { Index::UndefinedIndex }
```

The base index.

Definition at line 23 of file WickOperatorTemplate.hpp.

7.3.2.3 other

```
Index mrock::symbolic_operators::IndexComparison::other { Index::UndefinedIndex }
```

The other index.

Definition at line 24 of file WickOperatorTemplate.hpp.

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

- include/mrock/symbolic_operators/WickOperatorTemplate.hpp

7.4 mrock::symbolic_operators::IndexWrapper Struct Reference

A wrapper for a vector of Index values.

```
#include <IndexWrapper.hpp>
```

Public Member Functions

- template<class Archive >
  void serialize (Archive &ar, const unsigned int version)

  Serializes the IndexWrapper, required for boost support.
- IndexWrapper ()=default

  Default constructor.
- IndexWrapper (Index _spin)

  Constructs an IndexWrapper with a single Index value.
- IndexWrapper (const std::vector< Index > &_indizes)

  Constructs an IndexWrapper with a vector of Index values.
- IndexWrapper (std::vector< Index > &&_indizes)

  Constructs an IndexWrapper with a vector of Index values (move semantics).
- VECTOR_WRAPPER_FILL_MEMBERS (Index, indizes)
- auto operator<=> (const IndexWrapper &rhs) const =default

  Compares two IndexWrapper objects.

Public Attributes

- std::vector< Index > indizes

  The vector of Index values.

7.4.1 Detailed Description

A wrapper for a vector of Index values.

This struct provides serialization support and comparison operators.

Definition at line 141 of file IndexWrapper.hpp.

7.4.2 Constructor & Destructor Documentation

7.4.2.1 IndexWrapper() [1/4]

```
mrock::symbolic_operators::IndexWrapper::IndexWrapper ( )  [default]
```

Default constructor.

7.4.2.2 IndexWrapper() [2/4]

```
mrock::symbolic_operators::IndexWrapper::IndexWrapper (
            Index _spin )  [inline]
```

Constructs an IndexWrapper with a single Index value.

Parameters

_spin	The Index value to initialize with.

Definition at line 165 of file IndexWrapper.hpp.

7.4.2.3 IndexWrapper() [3/4]

```
mrock::symbolic_operators::IndexWrapper::IndexWrapper (
            const std::vector< Index > & _indizes )  [inline]
```

Constructs an IndexWrapper with a vector of Index values.

Parameters

_indizes	The vector of Index values to initialize with.

Definition at line 172 of file IndexWrapper.hpp.

7.4.2.4 IndexWrapper() [4/4]

```
mrock::symbolic_operators::IndexWrapper::IndexWrapper (
            std::vector< Index > && _indizes )  [inline]
```

Constructs an IndexWrapper with a vector of Index values (move semantics).

Parameters

_indizes	The vector of Index values to initialize with.

Definition at line 180 of file IndexWrapper.hpp.

7.4.3 Member Function Documentation

7.4.3.1 operator<=>()

```
auto mrock::symbolic_operators::IndexWrapper::operator<=> (
            const IndexWrapper & rhs ) const  [inline], [default]
```

Compares two IndexWrapper objects.

Parameters

rhs	The other IndexWrapper to compare with.

Returns

The result of the comparison.

7.4.3.2 serialize()

```
template<class Archive >
void mrock::symbolic_operators::IndexWrapper::serialize (
            Archive & ar,
            const unsigned int version )  [inline]
```

Serializes the IndexWrapper, required for boost support.

Template Parameters

Archive	The type of the archive.

Parameters

ar	The archive to serialize to.
version	The version of the serialization.

Definition at line 151 of file IndexWrapper.hpp.

7.4.3.3 VECTOR_WRAPPER_FILL_MEMBERS()

```
mrock::symbolic_operators::IndexWrapper::VECTOR_WRAPPER_FILL_MEMBERS (
            Index ,
            indizes  )
```

7.4.4 Member Data Documentation

7.4.4.1 indizes

```
std::vector<Index> mrock::symbolic_operators::IndexWrapper::indizes
```

The vector of Index values.

Definition at line 142 of file IndexWrapper.hpp.

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

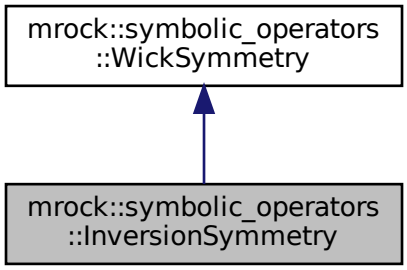
- include/mrock/symbolic_operators/IndexWrapper.hpp

7.5 mrock::symbolic_operators::InversionSymmetry Class Reference

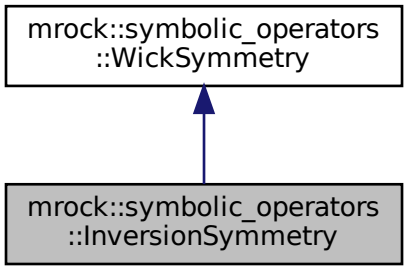
A symmetry where expectation values for k and -k are the same.

`#include <WickSymmetry.hpp>`

Inheritance diagram for mrock::symbolic_operators::InversionSymmetry:



Collaboration diagram for mrock::symbolic_operators::InversionSymmetry:



Public Member Functions

- void apply_to (WickTerm &term) const override

  Applies the inversion symmetry to a Wick term.

7.5.1 Detailed Description

A symmetry where expectation values for k and -k are the same.

Definition at line 70 of file WickSymmetry.hpp.

7.5.2 Member Function Documentation

7.5.2.1 apply_to()

```
void mrock::symbolic_operators::InversionSymmetry::apply_to (
            WickTerm & term ) const  [override], [virtual]
```

Applies the inversion symmetry to a Wick term.

Parameters

term	The Wick term to apply the symmetry to.

Implements mrock::symbolic_operators::WickSymmetry.

Definition at line 15 of file WickSymmetry.cpp.

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

- include/mrock/symbolic_operators/WickSymmetry.hpp
- sources/WickSymmetry.cpp

7.6 mrock::symbolic_operators::KroneckerDelta< T > Class Template Reference

A structure representing the Kronecker Delta.

`#include <KroneckerDelta.hpp>`

Public Member Functions

- template<class Archive >
  void serialize (Archive &ar, const unsigned int version)
  Serializes the KroneckerDelta object.
- constexpr bool isOne () const
  Checks if the Kronecker Delta is one.

Public Attributes

- T first {}
- T second {}

7.6.1 Detailed Description

template<typename T>
class mrock::symbolic_operators::KroneckerDelta< T >

A structure representing the Kronecker Delta.

Template Parameters

T	The type of the elements.

Deltas are represented via the `KroneckerDelta` class template. The template argument specifies what kind of delta is to be used, e.g., `Momentum`. Similar to how `std::pair<T1,T2>` implements `std::make_pair(T1,T2)` a function `make_delta(T,T)` is provided.

Definition at line 24 of file KroneckerDelta.hpp.

7.6.2 Member Function Documentation

7.6.2.1 isOne()

```
template<typename T >
constexpr bool mrock::symbolic_operators::KroneckerDelta< T >::isOne ( ) const  [inline],
[constexpr]
```

Checks if the Kronecker Delta is one.

Returns

true if first equals second.

false otherwise.

Definition at line 47 of file KroneckerDelta.hpp.

7.6.2.2 serialize()

```
template<typename T >
template<class Archive >
void mrock::symbolic_operators::KroneckerDelta< T >::serialize (
            Archive & ar,
            const unsigned int version )  [inline]
```

Serializes the KroneckerDelta object.

Template Parameters

Archive	The type of the archive.

Parameters

ar	The archive object.

Parameters

version	The version of the serialization.

Definition at line 36 of file KroneckerDelta.hpp.

7.6.3 Member Data Documentation

7.6.3.1 first

```
template<typename T >
T mrock::symbolic_operators::KroneckerDelta< T >::first {}
```

Definition at line 25 of file KroneckerDelta.hpp.

7.6.3.2 second

```
template<typename T >
T mrock::symbolic_operators::KroneckerDelta< T >::second {}
```

Definition at line 26 of file KroneckerDelta.hpp.

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

- include/mrock/symbolic_operators/KroneckerDelta.hpp

7.7 mrock::symbolic_operators::Momentum Struct Reference

Represents a collection of momentum symbols with associated operations.

```
#include <Momentum.hpp>
```

Public Member Functions

- template<class Archive >
  void serialize (Archive &ar, const unsigned int version)

  Serialization function for Boost.
- Momentum ()=default

  Default constructor.
- Momentum (const char value, int plus_minus=1, bool Q=false)

  Constructs a Momentum with a single symbol.
- Momentum (const MomentumSymbol::name_type value, int plus_minus=1, bool Q=false)

  Constructs a Momentum with a single symbol.
- Momentum (const momentum_symbols &_momenta, bool Q=false)

  Constructs a Momentum with a list of symbols.
- Momentum (MomentumSymbol const &momentum_symbol, bool Q=false)

  Constructs a Momentum with a single symbol.
- Momentum (const std::string &expression, bool Q=false)

  Constructs a Momentum from a string expression.
- Momentum (char, char)=delete

  Deleted constructor to prevent usage.
- void sort ()

  Sorts the momentum symbols.
- void remove_contribution (const MomentumSymbol::name_type momentum)

  Removes a specific momentum contribution.
- void add_in_place (const Momentum &rhs)

  Adds another Momentum in place.
- void replace_occurances (const MomentumSymbol::name_type replaceWhat, const Momentum &replaceWith)

  Replaces occurrences of a specific momentum with another Momentum.
- void remove_zeros ()

  Removes entries with a zero prefactor.
- void flip_single (const MomentumSymbol::name_type momentum)

  Flips a specific momentum if it exists.
- int is_used_at (const MomentumSymbol::name_type value) const noexcept

  Checks if a specific momentum is used.
- void multiply_by (int factor)

  Multiplies this Momentum by an integer factor.
- void flip_momentum ()

  Flips the momentum by multiplying by -1.
- bool differs_only_in_Q (Momentum rhs) const

  Checks if this Momentum differs from another only in the Q property.
- bool is_zero () const

  Checks if this Momentum is zero.
- bool uses (const MomentumSymbol::name_type what) const noexcept

  Checks if a specific momentum is used.
- bool first_momentum_is_negative () const

  Checks if the first momentum is negative.
- bool first_momentum_is (const MomentumSymbol::name_type what) const

  Checks if the first momentum is a specific value.
- bool last_momentum_is_negative () const

  Checks if the last momentum is negative.
- bool last_momentum_is (const MomentumSymbol::name_type what) const

Checks if the last momentum is a specific value.

- std::string to_string () const

  Converts this Momentum to a string representation.
- bool operator== (const Momentum &rhs) const

  Equality operator.
- bool operator!= (const Momentum &rhs) const

  Inequality operator.
- Momentum & operator+= (const Momentum &rhs)

  Adds another Momentum to this one.
- Momentum & operator-= (const Momentum &rhs)

  Subtracts another Momentum from this one.
- Momentum & operator∗= (const int rhs)

  Multiplies this Momentum by an integer factor.
- VECTOR_WRAPPER_FILL_MEMBERS (MomentumSymbol, momentum_list)

Public Attributes

- momentum_symbols momentum_list

  List of momentum symbols.
- bool add_Q {}

  Flag indicating additional property Q. Q is a special momentum with the property $2Q = 0$. Remeber that momenta are only defined in the first Brillouin zone.

7.7.1 Detailed Description

Represents a collection of momentum symbols with associated operations.

This class represents momenta. It includes addition and substraction operators as well as a `bool add_Q`. Q is defined as $(\pi, \pi, \cdots)$, i.e., $nQ = 0$ for all even n. Besides the normal operators in which you specify the class members, you can also pass a string like "3k+l-p" to the constructor to create that specific momentum. If you want to add Q here, you can do so by passing `true` to the same constructor as a second argument.

Definition at line 37 of file Momentum.hpp.

7.7.2 Constructor & Destructor Documentation

7.7.2.1 Momentum() [1/7]

```
mrock::symbolic_operators::Momentum::Momentum ( )  [default]
```

Default constructor.

7.7.2.2 Momentum() [2/7]

```
mrock::symbolic_operators::Momentum::Momentum (
            const char value,
            int plus_minus = 1,
            bool Q = false )  [explicit]
```

Constructs a Momentum with a single symbol.

Parameters

value	Character representing the symbol.
plus_minus	Factor associated with the symbol.
Q	Additional property Q.

Definition at line 236 of file Momentum.cpp.

7.7.2.3 Momentum() [3/7]

```
mrock::symbolic_operators::Momentum::Momentum (
            const MomentumSymbol::name_type value,
            int plus_minus = 1,
            bool Q = false )  [explicit]
```

Constructs a Momentum with a single symbol.

Parameters

value	Name type of the symbol.
plus_minus	Factor associated with the symbol.
Q	Additional property Q.

Definition at line 239 of file Momentum.cpp.

7.7.2.4 Momentum() [4/7]

```
mrock::symbolic_operators::Momentum::Momentum (
            const momentum_symbols & _momenta,
            bool Q = false )  [explicit]
```

Constructs a Momentum with a list of symbols.

Parameters

_momenta	List of momentum symbols.
Q	Additional property Q.

Definition at line 242 of file Momentum.cpp.

7.7.2.5 Momentum() [5/7]

```
mrock::symbolic_operators::Momentum::Momentum (
```

```
            MomentumSymbol const & momentum_symbol,
            bool Q = false )  [explicit]
```

Constructs a Momentum with a single symbol.

Parameters

momentum_symbol	The momentum symbol.
Q	Additional property Q.

Definition at line 245 of file Momentum.cpp.

7.7.2.6 Momentum() [6/7]

```
mrock::symbolic_operators::Momentum::Momentum (
            const std::string & expression,
            bool Q = false )
```

Constructs a Momentum from a string expression.

Parameters

expression	String representing the momentum expression.
Q	Additional property Q.

Definition at line 248 of file Momentum.cpp.

7.7.2.7 Momentum() [7/7]

```
mrock::symbolic_operators::Momentum::Momentum (
            char ,
            char  )  [delete]
```

Deleted constructor to prevent usage.

7.7.3 Member Function Documentation

7.7.3.1 add_in_place()

```
void mrock::symbolic_operators::Momentum::add_in_place (
            const Momentum & rhs )
```

Adds another Momentum in place.

Parameters

rhs	The other Momentum.

Definition at line 48 of file Momentum.cpp.

7.7.3.2 differs_only_in_Q()

```
bool mrock::symbolic_operators::Momentum::differs_only_in_Q (
            Momentum rhs ) const  [inline]
```

Checks if this Momentum differs from another only in the Q property.

Parameters

rhs	The other Momentum.

Returns

True if they differ only in Q, false otherwise.

Definition at line 346 of file Momentum.hpp.

7.7.3.3 first_momentum_is()

```
bool mrock::symbolic_operators::Momentum::first_momentum_is (
            const MomentumSymbol::name_type what ) const  [inline]
```

Checks if the first momentum is a specific value.

Parameters

what	Name of the momentum.

Returns

True if it matches, false otherwise.

Definition at line 362 of file Momentum.hpp.

7.7.3.4 first_momentum_is_negative()

```
bool mrock::symbolic_operators::Momentum::first_momentum_is_negative ( ) const  [inline]
```

Checks if the first momentum is negative.

Returns

True if negative, false otherwise.

Definition at line 358 of file Momentum.hpp.

7.7.3.5 flip_momentum()

```
void mrock::symbolic_operators::Momentum::flip_momentum ( )  [inline]
```

Flips the momentum by multiplying by -1.

Definition at line 343 of file Momentum.hpp.

7.7.3.6 flip_single()

```
void mrock::symbolic_operators::Momentum::flip_single (
            const MomentumSymbol::name_type momentum )
```

Flips a specific momentum if it exists.

Parameters

momentum	Name of the momentum to flip.

Definition at line 83 of file Momentum.cpp.

7.7.3.7 is_used_at()

```
int mrock::symbolic_operators::Momentum::is_used_at (
            const MomentumSymbol::name_type value ) const  [noexcept]
```

Checks if a specific momentum is used.

Parameters

value	Name of the momentum.

Returns

Position of the momentum in the list, or -1 if not found.

Definition at line 92 of file Momentum.cpp.

7.7.3.8 is_zero()

```
bool mrock::symbolic_operators::Momentum::is_zero ( ) const  [inline]
```

Checks if this Momentum is zero.

Returns

True if zero, false otherwise.

Definition at line 351 of file Momentum.hpp.

7.7.3.9 last_momentum_is()

```
bool mrock::symbolic_operators::Momentum::last_momentum_is (
            const MomentumSymbol::name_type what ) const  [inline]
```

Checks if the last momentum is a specific value.

Parameters

what	Name of the momentum.

Returns

True if it matches, false otherwise.

Definition at line 370 of file Momentum.hpp.

7.7.3.10 last_momentum_is_negative()

```
bool mrock::symbolic_operators::Momentum::last_momentum_is_negative ( ) const  [inline]
```

Checks if the last momentum is negative.

Returns

True if negative, false otherwise.

Definition at line 366 of file Momentum.hpp.

7.7.3.11 multiply_by()

```
void mrock::symbolic_operators::Momentum::multiply_by (
            int factor )  [inline]
```

Multiplies this Momentum by an integer factor.

Parameters

factor	The factor.

Definition at line 340 of file Momentum.hpp.

7.7.3.12 operator"!=()

```
bool mrock::symbolic_operators::Momentum::operator!= (
            const Momentum & rhs ) const  [inline]
```

Inequality operator.

Parameters

rhs	The other Momentum.

Returns

True if not equal, false otherwise.

Definition at line 374 of file Momentum.hpp.

7.7.3.13 operator*=()

```
Momentum & mrock::symbolic_operators::Momentum::operator*= (
            const int rhs )  [inline]
```

Multiplies this Momentum by an integer factor.

Parameters

rhs	The factor.

Returns

Reference to this Momentum.

Definition at line 331 of file Momentum.hpp.

7.7.3.14 operator+=()

```
Momentum & mrock::symbolic_operators::Momentum::operator+= (
            const Momentum & rhs )
```

Adds another Momentum to this one.

Parameters

rhs	The other Momentum.

Returns

Reference to this Momentum.

Definition at line 124 of file Momentum.cpp.

7.7.3.15 operator-=()

```
Momentum & mrock::symbolic_operators::Momentum::operator-= (
            const Momentum & rhs )
```

Subtracts another Momentum from this one.

Parameters

rhs	The other Momentum.

Returns

Reference to this Momentum.

Definition at line 150 of file Momentum.cpp.

7.7.3.16 operator==()

```
bool mrock::symbolic_operators::Momentum::operator== (
            const Momentum & rhs ) const
```

Equality operator.

Parameters

rhs	The other Momentum.

Returns

True if equal, false otherwise.

Definition at line 105 of file Momentum.cpp.

7.7.3.17 remove_contribution()

```
void mrock::symbolic_operators::Momentum::remove_contribution (
            const MomentumSymbol::name_type momentum )
```

Removes a specific momentum contribution.

Parameters

momentum	Name of the momentum to remove.

Definition at line 41 of file Momentum.cpp.

7.7.3.18 remove_zeros()

```
void mrock::symbolic_operators::Momentum::remove_zeros ( )
```

Removes entries with a zero prefactor.

Definition at line 71 of file Momentum.cpp.

7.7.3.19 replace_occurances()

```
void mrock::symbolic_operators::Momentum::replace_occurances (
            const MomentumSymbol::name_type replaceWhat,
            const Momentum & replaceWith )
```

Replaces occurrences of a specific momentum with another Momentum.

Parameters

replaceWhat	Name of the momentum to replace.
replaceWith	The Momentum to replace with.

Definition at line 53 of file Momentum.cpp.

7.7.3.20 serialize()

```
template<class Archive >
void mrock::symbolic_operators::Momentum::serialize (
            Archive & ar,
            const unsigned int version )  [inline]
```

Serialization function for Boost.

Template Parameters

Archive	Type of the archive.

Parameters

ar	Archive to serialize to/from.
version	Version of the serialization.

Definition at line 48 of file Momentum.hpp.

7.7.3.21 sort()

```
void mrock::symbolic_operators::Momentum::sort ( )
```

Sorts the momentum symbols.

Definition at line 26 of file Momentum.cpp.

7.7.3.22 to_string()

```
std::string mrock::symbolic_operators::Momentum::to_string ( ) const
```

Converts this Momentum to a string representation.

Returns

String representation of this Momentum.

Definition at line 99 of file Momentum.cpp.

7.7.3.23 uses()

```
bool mrock::symbolic_operators::Momentum::uses (
            const MomentumSymbol::name_type what ) const  [inline], [noexcept]
```

Checks if a specific momentum is used.

Parameters

what	Name of the momentum.

Returns

True if used, false otherwise.

Definition at line 355 of file Momentum.hpp.

7.7.3.24 VECTOR_WRAPPER_FILL_MEMBERS()

```
mrock::symbolic_operators::Momentum::VECTOR_WRAPPER_FILL_MEMBERS (
            MomentumSymbol ,
            momentum_list  )
```

7.7.4 Member Data Documentation

7.7.4.1 add_Q

```
bool mrock::symbolic_operators::Momentum::add_Q {}
```

Flag indicating additional property Q. Q is a special momentum with the property $2Q = 0$. Remeber that momenta are only defined in the first Brillouin zone.

Definition at line 39 of file Momentum.hpp.

7.7.4.2 momentum_list

```
momentum_symbols mrock::symbolic_operators::Momentum::momentum_list
```

List of momentum symbols.

Definition at line 38 of file Momentum.hpp.

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

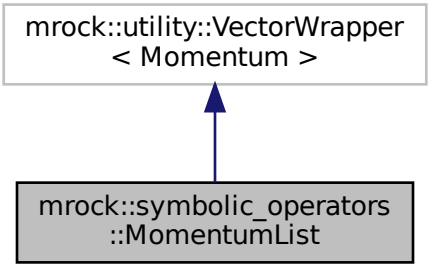
- include/mrock/symbolic_operators/Momentum.hpp
- sources/Momentum.cpp

7.8 mrock::symbolic_operators::MomentumList Class Reference

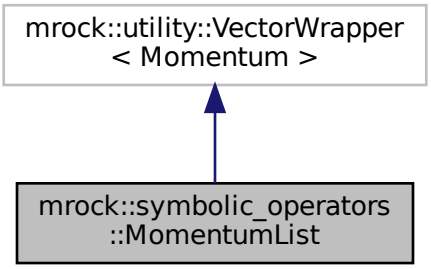
A wrapper class for a vector of Momentum objects with additional functionalities.

`#include <MomentumList.hpp>`

Inheritance diagram for mrock::symbolic_operators::MomentumList:



Collaboration diagram for mrock::symbolic_operators::MomentumList:



Public Member Functions

- template<class Archive >
  void serialize (Archive &ar, const unsigned int version)

  Serializes the MomentumList object, required for boost support.
- MomentumList ()

  Default constructor.
- MomentumList (const Momentum &momentum)

  Constructs a MomentumList with a single Momentum object.
- MomentumList (const Momentum &first, const Momentum &second)

Constructs a MomentumList with two Momentum objects.

- MomentumList (std::initializer_list< Momentum > init)

  Constructs a MomentumList with an initializer list of Momentum objects.
- MomentumList (std::initializer_list< char > init)

  Constructs a MomentumList with an initializer list of characters.
- MomentumList & operator*= (const int rhs)

  Multiplies each Momentum object in the list by a given factor.
- void multiply_by (int factor)

  Multiplies each Momentum object in the list by a given factor.
- void flip_momentum ()

  Flips the momentum of each Momentum object in the list.
- void sort ()

  Sorts the Momentum objects in the list.
- void replace_occurances (const MomentumSymbol::name_type replaceWhat, const Momentum &replaceWith)

  Replaces occurrences of a specific MomentumSymbol name with a given Momentum object.
- void remove_zeros ()

  Removes Momentum objects with zero value from the list.
- void flip_single (const MomentumSymbol::name_type momentum)

  Flips the momentum of a single Momentum object identified by its MomentumSymbol name.

Private Types

- using _parent = mrock::utility::VectorWrapper< Momentum >

7.8.1 Detailed Description

A wrapper class for a vector of Momentum objects with additional functionalities.

Definition at line 17 of file MomentumList.hpp.

7.8.2 Member Typedef Documentation

7.8.2.1 _parent

```
using mrock::symbolic_operators::MomentumList::_parent = mrock::utility::VectorWrapper<Momentum>
[private]
```

Definition at line 20 of file MomentumList.hpp.

7.8.3 Constructor & Destructor Documentation

7.8.3.1 MomentumList() [1/5]

```
mrock::symbolic_operators::MomentumList::MomentumList ( )
```

Default constructor.

Definition at line 36 of file MomentumList.cpp.

7.8.3.2 MomentumList() [2/5]

```
mrock::symbolic_operators::MomentumList::MomentumList (
            const Momentum & momentum )  [explicit]
```

Constructs a MomentumList with a single Momentum object.

Parameters

momentum	The Momentum object to initialize the list with.

Definition at line 38 of file MomentumList.cpp.

7.8.3.3 MomentumList() [3/5]

```
mrock::symbolic_operators::MomentumList::MomentumList (
            const Momentum & first,
            const Momentum & second )
```

Constructs a MomentumList with two Momentum objects.

Parameters

first	The first Momentum object.
second	The second Momentum object.

Definition at line 41 of file MomentumList.cpp.

7.8.3.4 MomentumList() [4/5]

```
mrock::symbolic_operators::MomentumList::MomentumList (
            std::initializer_list< Momentum > init )
```

Constructs a MomentumList with an initializer list of Momentum objects.

Parameters

init	The initializer list of Momentum objects.

Definition at line 44 of file MomentumList.cpp.

7.8.3.5 MomentumList() [5/5]

```
mrock::symbolic_operators::MomentumList::MomentumList (
            std::initializer_list< char > init )
```

Constructs a MomentumList with an initializer list of characters.

Parameters

init	The initializer list of characters.

Definition at line 47 of file MomentumList.cpp.

7.8.4 Member Function Documentation

7.8.4.1 flip_momentum()

```
void mrock::symbolic_operators::MomentumList::flip_momentum ( )  [inline]
```

Flips the momentum of each Momentum object in the list.

Definition at line 124 of file MomentumList.hpp.

7.8.4.2 flip_single()

```
void mrock::symbolic_operators::MomentumList::flip_single (
            const MomentumSymbol::name_type momentum )
```

Flips the momentum of a single Momentum object identified by its MomentumSymbol name.

Parameters

momentum	The MomentumSymbol name of the Momentum object to flip.

Definition at line 16 of file MomentumList.cpp.

7.8.4.3 multiply_by()

```
void mrock::symbolic_operators::MomentumList::multiply_by (
            int factor )  [inline]
```

Multiplies each Momentum object in the list by a given factor.

Parameters

factor	The factor to multiply by.

Definition at line 121 of file MomentumList.hpp.

7.8.4.4 operator*=()

```
MomentumList & mrock::symbolic_operators::MomentumList::operator*= (
            const int rhs )  [inline]
```

Multiplies each Momentum object in the list by a given factor.

Parameters

rhs	The factor to multiply by.

Returns

A reference to the modified MomentumList.

Definition at line 115 of file MomentumList.hpp.

7.8.4.5 remove_zeros()

```
void mrock::symbolic_operators::MomentumList::remove_zeros ( )
```

Removes Momentum objects with zero value from the list.

Definition at line 10 of file MomentumList.cpp.

7.8.4.6 replace_occurances()

```
void mrock::symbolic_operators::MomentumList::replace_occurances (
            const MomentumSymbol::name_type replaceWhat,
            const Momentum & replaceWith )
```

Replaces occurrences of a specific MomentumSymbol name with a given Momentum object.

Parameters

replaceWhat	The MomentumSymbol name to replace.
replaceWith	The Momentum object to replace with.

Definition at line 4 of file MomentumList.cpp.

7.8.4.7 serialize()

```
template<class Archive >
void mrock::symbolic_operators::MomentumList::serialize (
            Archive & ar,
            const unsigned int version )  [inline]
```

Serializes the MomentumList object, required for boost support.

Template Parameters

Archive	The type of the archive.

Parameters

ar	The archive object.
version	The version of the serialization.

Definition at line 30 of file MomentumList.hpp.

7.8.4.8 sort()

```
void mrock::symbolic_operators::MomentumList::sort ( )  [inline]
```

Sorts the Momentum objects in the list.

Definition at line 127 of file MomentumList.hpp.

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

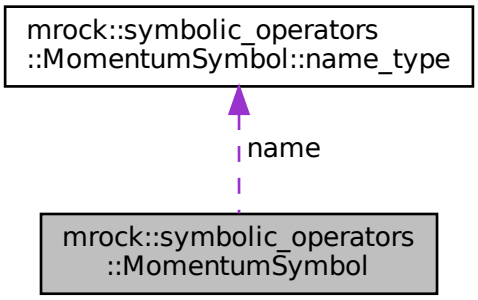
- include/mrock/symbolic_operators/MomentumList.hpp
- sources/MomentumList.cpp

7.9 mrock::symbolic_operators::MomentumSymbol Struct Reference

Represents a symbolic momentum with a factor and a name.

`#include <MomentumSymbol.hpp>`

Collaboration diagram for mrock::symbolic_operators::MomentumSymbol:



Classes

- struct name_type

  Represents a name as a single character with comparison and serialization capabilities, but without arithmetic operations (it does not make sense to add or multiply names).

Public Member Functions

- template<class Archive >
  void serialize (Archive &ar, const unsigned int version)

  Serializes the MomentumSymbol object, required for boost support.
- constexpr MomentumSymbol ()=default
- constexpr MomentumSymbol (int _factor, char _name)

  Constructs a MomentumSymbol with a given factor and name.
- constexpr MomentumSymbol (int _factor, name_type _name)

  Constructs a MomentumSymbol with a given factor and name_type.
- constexpr auto operator<=> (const MomentumSymbol &) const =default

  Compares two MomentumSymbol objects.

Public Attributes

- int factor {}

  The factor associated with the momentum.
- name_type name {}

  The name associated with the momentum.

7.9.1 Detailed Description

Represents a symbolic momentum with a factor and a name.

Definition at line 16 of file MomentumSymbol.hpp.

7.9.2 Constructor & Destructor Documentation

7.9.2.1 MomentumSymbol() [1/3]

```
constexpr mrock::symbolic_operators::MomentumSymbol::MomentumSymbol ( )  [constexpr], [default]
```

7.9.2.2 MomentumSymbol() [2/3]

```
constexpr mrock::symbolic_operators::MomentumSymbol::MomentumSymbol (
            int _factor,
            char _name )  [inline], [constexpr]
```

Constructs a MomentumSymbol with a given factor and name.

Parameters

_factor	The factor to initialize the momentum with.
_name	The name to initialize the momentum with.

Definition at line 87 of file MomentumSymbol.hpp.

7.9.2.3 MomentumSymbol() [3/3]

```
constexpr mrock::symbolic_operators::MomentumSymbol::MomentumSymbol (
            int _factor,
            name_type _name )  [inline], [constexpr]
```

Constructs a MomentumSymbol with a given factor and name_type.

Parameters

_factor	The factor to initialize the momentum with.
_name	The name_type to initialize the momentum with.

Definition at line 94 of file MomentumSymbol.hpp.

7.9.3 Member Function Documentation

7.9.3.1 operator<=>()

```
constexpr auto mrock::symbolic_operators::MomentumSymbol::operator<=> (
            const MomentumSymbol &  ) const  [constexpr], [default]
```

Compares two MomentumSymbol objects.

Parameters

other	The other MomentumSymbol object to compare with.

Returns

The result of the comparison.

7.9.3.2 serialize()

```
template<class Archive >
void mrock::symbolic_operators::MomentumSymbol::serialize (
            Archive & ar,
            const unsigned int version )  [inline]
```

Serializes the MomentumSymbol object, required for boost support.

Template Parameters

Archive	The type of the archive.

Parameters

ar	The archive to serialize to.
version	The version of the serialization.

Definition at line 75 of file MomentumSymbol.hpp.

7.9.4 Member Data Documentation

7.9.4.1 factor

```
int mrock::symbolic_operators::MomentumSymbol::factor {}
```

The factor associated with the momentum.

Definition at line 65 of file MomentumSymbol.hpp.

7.9.4.2 name

```
name_type mrock::symbolic_operators::MomentumSymbol::name {}
```

The name associated with the momentum.

Definition at line 66 of file MomentumSymbol.hpp.

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

- include/mrock/symbolic_operators/MomentumSymbol.hpp

7.10 mrock::symbolic_operators::MomentumSymbol::name_type Struct Reference

Represents a name as a single character with comparison and serialization capabilities, but without arithmetic operations (it does not make sense to add or multiply names).

```
#include <MomentumSymbol.hpp>
```

Public Member Functions

- template<class Archive >
  void serialize (Archive &ar, const unsigned int version)
  Serializes the name_type object, required for boost support.
- constexpr name_type ()=default
- constexpr name_type (char n) noexcept
  Constructs a name_type with a given character.
- constexpr auto operator<=> (const name_type &) const =default
  Compares two name_type objects.
- constexpr auto operator<=> (const char other) const
  Compares the name_type object with a character.
- constexpr operator char () const noexcept
  Converts the name_type object to a character.

Public Attributes

- char _n {}
  The character representing the name.

7.10.1 Detailed Description

Represents a name as a single character with comparison and serialization capabilities, but without arithmetic operations (it does not make sense to add or multiply names).

Definition at line 23 of file MomentumSymbol.hpp.

7.10.2 Constructor & Destructor Documentation

7.10.2.1 name_type() [1/2]

```
constexpr mrock::symbolic_operators::MomentumSymbol::name_type::name_type ( )  [constexpr],
[default]
```

7.10.2.2 name_type() [2/2]

```
constexpr mrock::symbolic_operators::MomentumSymbol::name_type::name_type (
            char n )  [inline], [constexpr], [noexcept]
```

Constructs a name_type with a given character.

Parameters

n	The character to initialize the name with.

Definition at line 42 of file MomentumSymbol.hpp.

7.10.3 Member Function Documentation

7.10.3.1 operator char()

```
constexpr mrock::symbolic_operators::MomentumSymbol::name_type::operator char ( ) const  [inline],
[explicit], [constexpr], [noexcept]
```

Converts the name_type object to a character.

Returns

The character representation of the name_type.

Definition at line 62 of file MomentumSymbol.hpp.

7.10.3.2 operator<=>() [1/2]

```
constexpr auto mrock::symbolic_operators::MomentumSymbol::name_type::operator<=> (
            const char other ) const  [inline], [constexpr]
```

Compares the name_type object with a character.

Parameters

other	The character to compare with.

Returns

The result of the comparison.

Definition at line 56 of file MomentumSymbol.hpp.

7.10.3.3 operator<=>() [2/2]

```
constexpr auto mrock::symbolic_operators::MomentumSymbol::name_type::operator<=> (
            const name_type &  ) const  [constexpr], [default]
```

Compares two name_type objects.

Parameters

other	The other name_type object to compare with.

Returns

The result of the comparison.

7.10.3.4 serialize()

```
template<class Archive >
void mrock::symbolic_operators::MomentumSymbol::name_type::serialize (
            Archive & ar,
            const unsigned int version )  [inline]
```

Serializes the name_type object, required for boost support.

Template Parameters

Archive	The type of the archive.

Parameters

ar	The archive to serialize to.
version	The version of the serialization.

Definition at line 33 of file MomentumSymbol.hpp.

7.10.4 Member Data Documentation

7.10.4.1 _n

```
char mrock::symbolic_operators::MomentumSymbol::name_type::_n {}
```

The character representing the name.

Definition at line 24 of file MomentumSymbol.hpp.

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

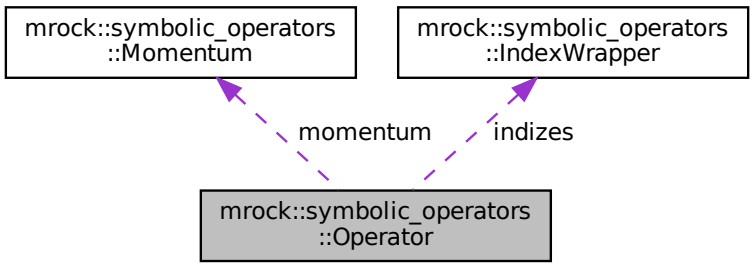
- include/mrock/symbolic_operators/MomentumSymbol.hpp

7.11 mrock::symbolic_operators::Operator Struct Reference

Represents a symbolic operator with momentum, indices, and properties.

```
#include <Operator.hpp>
```

Collaboration diagram for mrock::symbolic_operators::Operator:



Public Member Functions

- template<class Archive >
  void serialize (Archive &ar, const unsigned int version)

  Serializes the Operator object.
- Operator ()=default

  Default constructor.
- Operator (const Momentum &_momentum, const IndexWrapper _indizes, bool _is_daggered, bool _is_↩ fermion=true)

  Constructs an Operator with specified momentum, indices, daggered state, and fermion state.
- Operator (const momentum_symbols &_momentum, const IndexWrapper _indizes, bool _is_daggered, bool _is_fermion=true)

  Constructs an Operator with specified momentum symbols, indices, daggered state, and fermion state.
- Operator (const MomentumSymbol::name_type _momentum, bool add_Q, const IndexWrapper _indizes, bool _is_daggered, bool _is_fermion=true)

  Constructs an Operator with specified momentum symbol name, addition flag, indices, daggered state, and fermion state.
- Operator (const MomentumSymbol::name_type _momentum, int sign, bool add_Q, const IndexWrapper _↩ indizes, bool _is_daggered, bool _is_fermion=true)

  Constructs an Operator with specified momentum symbol name, sign, addition flag, indices, daggered state, and fermion state.
- Operator & hermitian_conjugate_inplace ()

  Toggles the daggered state of the operator.
- Operator hermitian_conjugate () const

  Creates hermitian conjugate of this as a new object.
- Operator with_momentum (Momentum const &new_momentum) const

  Creates a new operator with updated momentum.
- Operator with_momentum (const MomentumSymbol::name_type new_momentum) const

  Creates a new operator with updated momentum symbol name.
- Operator add_momentum (Momentum const &to_add) const

  Creates a new operator by adding momentum.
- Operator add_momentum (const MomentumSymbol::name_type to_add) const

  Creates a new operator by adding momentum symbol name.
- void remove_momentum_contribution (const MomentumSymbol::name_type value)

  Removes a momentum contribution from the operator.
- Index first_index () const

  Returns the first index of the operator.
- void set_first_index (Index index)

  Sets the first index of the operator.

Static Public Member Functions

- static Operator Boson (const Momentum &_momentum, const IndexWrapper _indizes, bool _is_daggered)

  Creates a Boson operator with specified momentum and indices.
- static Operator Boson (const Momentum &_momentum, bool _is_daggered)

  Creates a Boson operator with specified momentum.

Public Attributes

- Momentum momentum

  The momentum associated with the operator.
- IndexWrapper indizes

  Contains all indices, standard: first index = spin, all others arbitrary, e.g., orbitals, bands etc.
- bool is_daggered {}

  Indicates if the operator is daggered (conjugate transpose).
- bool is_fermion { true }

  Indicates if the operator is a fermion. This of course impacts the commutation relation [O', O^+]_{+/-} = delta_{O,O'}, where the plus applies to fermions and the minus to bosons.

7.11.1 Detailed Description

Represents a symbolic operator with momentum, indices, and properties.

This class represents the standard fermionic or bosonic creation and annihilation operators. You can specify its momentum, its indizes and whether it is supposed to be daggered (a creation operator) or not (an annihilation operator).

See also

Momentum, IndexWrapper

Definition at line 23 of file Operator.hpp.

7.11.2 Constructor & Destructor Documentation

7.11.2.1 Operator() [1/5]

```
mrock::symbolic_operators::Operator::Operator ( )  [default]
```

Default constructor.

7.11.2.2 Operator() [2/5]

```
mrock::symbolic_operators::Operator::Operator (
            const Momentum & _momentum,
            const IndexWrapper _indizes,
            bool _is_daggered,
            bool _is_fermion = true )
```

Constructs an Operator with specified momentum, indices, daggered state, and fermion state.

Parameters

_momentum	The momentum of the operator.
_indizes	The indices of the operator.
_is_daggered	The daggered state of the operator.
_is_fermion	The fermion state of the operator (default is true).

Definition at line 26 of file Operator.cpp.

7.11.2.3 Operator() [3/5]

```
mrock::symbolic_operators::Operator::Operator (
            const momentum_symbols & _momentum,
            const IndexWrapper _indizes,
            bool _is_daggered,
            bool _is_fermion = true )
```

Constructs an Operator with specified momentum symbols, indices, daggered state, and fermion state.

Parameters

_momentum	The momentum symbols of the operator.
_indizes	The indices of the operator.
_is_daggered	The daggered state of the operator.
_is_fermion	The fermion state of the operator (default is true).

Definition at line 29 of file Operator.cpp.

7.11.2.4 Operator() [4/5]

```
mrock::symbolic_operators::Operator::Operator (
            const MomentumSymbol::name_type _momentum,
            bool add_Q,
            const IndexWrapper _indizes,
            bool _is_daggered,
            bool _is_fermion = true )
```

Constructs an Operator with specified momentum symbol name, addition flag, indices, daggered state, and fermion state.

Parameters

_momentum	The name of the momentum symbol.
add_Q	Flag to indicate if Q should be added. Q has the property 2Q = 0, e.g., (pi,pi) on a unit square lattice.
_indizes	The indices of the operator.
_is_daggered	The daggered state of the operator.
_is_fermion	The fermion state of the operator (default is true).

Definition at line 32 of file Operator.cpp.

7.11.2.5 Operator() [5/5]

```
mrock::symbolic_operators::Operator::Operator (
            const MomentumSymbol::name_type _momentum,
            int sign,
            bool add_Q,
            const IndexWrapper _indizes,
            bool _is_daggered,
            bool _is_fermion = true )
```

Constructs an Operator with specified momentum symbol name, sign, addition flag, indices, daggered state, and fermion state.

Parameters

_momentum	The name of the momentum symbol.
sign	The sign of the momentum.
add_Q	Flag to indicate if Q should be added. Q has the property 2Q = 0, e.g., (pi,pi) on a unit square lattice.
_indizes	The indices of the operator.
_is_daggered	The daggered state of the operator.
_is_fermion	The fermion state of the operator (default is true).

Definition at line 35 of file Operator.cpp.

7.11.3 Member Function Documentation

7.11.3.1 add_momentum() [1/2]

```
Operator mrock::symbolic_operators::Operator::add_momentum (
            const MomentumSymbol::name_type to_add ) const  [inline]
```

Creates a new operator by adding momentum symbol name.

Parameters

to_add	The momentum symbol name to add.

Returns

A new operator with the added momentum symbol name.

Definition at line 234 of file Operator.hpp.

7.11.3.2 add_momentum() [2/2]

```
Operator mrock::symbolic_operators::Operator::add_momentum (
            Momentum const & to_add ) const  [inline]
```

Creates a new operator by adding momentum.

Parameters

to_add	The momentum to add.

Returns

A new operator with the added momentum.

Definition at line 229 of file Operator.hpp.

7.11.3.3 Boson() [1/2]

```
static Operator mrock::symbolic_operators::Operator::Boson (
            const Momentum & _momentum,
            bool _is_daggered )  [inline], [static]
```

Creates a Boson operator with specified momentum.

Parameters

_momentum	The momentum of the operator.
_is_daggered	The daggered state of the operator.

Returns

A Boson operator.

Definition at line 104 of file Operator.hpp.

7.11.3.4 Boson() [2/2]

```
static Operator mrock::symbolic_operators::Operator::Boson (
            const Momentum & _momentum,
            const IndexWrapper _indizes,
            bool _is_daggered )  [inline], [static]
```

Creates a Boson operator with specified momentum and indices.

Parameters

_momentum	The momentum of the operator.
_indizes	The indices of the operator.
_is_daggered	The daggered state of the operator.

Returns

A Boson operator.

Definition at line 94 of file Operator.hpp.

7.11.3.5 first_index()

```
Index mrock::symbolic_operators::Operator::first_index ( ) const  [inline]
```

Returns the first index of the operator.

Returns

The first index if the operator has indices, otherwise Index::NoIndex.

Definition at line 244 of file Operator.hpp.

7.11.3.6 hermitian_conjugate()

```
Operator mrock::symbolic_operators::Operator::hermitian_conjugate ( ) const  [inline]
```

Creates hermitian conjugate of this as a new object.

Returns

Returns the new object.

Definition at line 212 of file Operator.hpp.

7.11.3.7 hermitian_conjugate_inplace()

```
Operator & mrock::symbolic_operators::Operator::hermitian_conjugate_inplace ( )  [inline]
```

Toggles the daggered state of the operator.

Returns

A reference to ∗this

Definition at line 208 of file Operator.hpp.

7.11.3.8 remove_momentum_contribution()

```
void mrock::symbolic_operators::Operator::remove_momentum_contribution (
            const MomentumSymbol::name_type value )  [inline]
```

Removes a momentum contribution from the operator.

Parameters

value	The momentum symbol name to remove.

Definition at line 239 of file Operator.hpp.

7.11.3.9 serialize()

```
template<class Archive >
void mrock::symbolic_operators::Operator::serialize (
            Archive & ar,
            const unsigned int version )  [inline]
```

Serializes the Operator object.

Template Parameters

Archive	The type of the archive.

Parameters

ar	The archive to serialize to.
version	The version of the serialization.

Definition at line 36 of file Operator.hpp.

7.11.3.10 set_first_index()

```
void mrock::symbolic_operators::Operator::set_first_index (
            Index index )  [inline]
```

Sets the first index of the operator.

Parameters

index	The index to set as the first index.

Definition at line 249 of file Operator.hpp.

7.11.3.11 with_momentum() [1/2]

```
Operator mrock::symbolic_operators::Operator::with_momentum (
            const MomentumSymbol::name_type new_momentum ) const  [inline]
```

Creates a new operator with updated momentum symbol name.

Parameters

new_momentum	The new momentum symbol name to set.

Returns

A new operator with the updated momentum symbol name.

Definition at line 223 of file Operator.hpp.

7.11.3.12 with_momentum() [2/2]

```
Operator mrock::symbolic_operators::Operator::with_momentum (
            Momentum const & new_momentum ) const  [inline]
```

Creates a new operator with updated momentum.

Parameters

new_momentum	The new momentum to set.

Returns

A new operator with the updated momentum.

Definition at line 217 of file Operator.hpp.

7.11.4 Member Data Documentation

7.11.4.1 indizes

```
IndexWrapper mrock::symbolic_operators::Operator::indizes
```

Contains all indices, standard: first index = spin, all others arbitrary, e.g., orbitals, bands etc.

Definition at line 25 of file Operator.hpp.

7.11.4.2 is_daggered

```
bool mrock::symbolic_operators::Operator::is_daggered {}
```

Indicates if the operator is daggered (conjugate transpose).

Definition at line 26 of file Operator.hpp.

7.11.4.3 is_fermion

```
bool mrock::symbolic_operators::Operator::is_fermion { true }
```

Indicates if the operator is a fermion. This of course impacts the commutation relation [O', O^+]_{+/-} = delta_{O,O'}, where the plus applies to fermions and the minus to bosons.

Definition at line 27 of file Operator.hpp.

7.11.4.4 momentum

```
Momentum mrock::symbolic_operators::Operator::momentum
```

The momentum associated with the operator.

Definition at line 24 of file Operator.hpp.

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

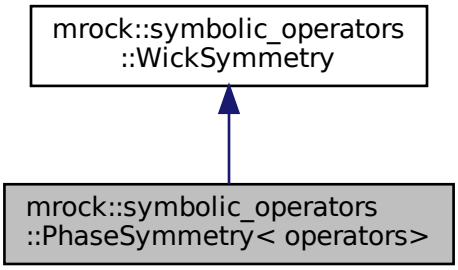
- include/mrock/symbolic_operators/Operator.hpp
- sources/Operator.cpp

7.12 mrock::symbolic_operators::PhaseSymmetry< operators> Class Template Reference

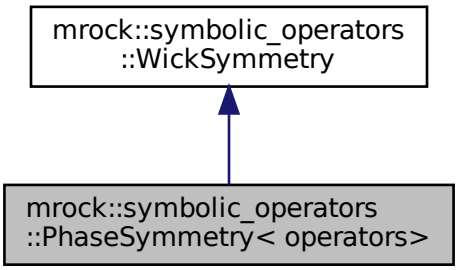
A symmetry where <operator^+> = <operator>.

```
#include <WickSymmetry.hpp>
```

Inheritance diagram for mrock::symbolic_operators::PhaseSymmetry< operators>:



Collaboration diagram for mrock::symbolic_operators::PhaseSymmetry< operators>:



Public Member Functions

- void apply_to (WickTerm &term) const override

  Applies the phase symmetry to a Wick term.

7.12.1 Detailed Description

template<OperatorType... operators>
class mrock::symbolic_operators::PhaseSymmetry< operators>

A symmetry where <operator^+> = <operator>.

Template Parameters

operators	The operator types to which the symmetry applies.

Definition at line 86 of file WickSymmetry.hpp.

7.12.2 Member Function Documentation

7.12.2.1 apply_to()

```
template<OperatorType...  operators>
void mrock::symbolic_operators::PhaseSymmetry< operators>::apply_to (
            WickTerm & term ) const  [inline], [override], [virtual]
```

Applies the phase symmetry to a Wick term.

Parameters

term	The Wick term to apply the symmetry to.

Implements mrock::symbolic_operators::WickSymmetry.

Definition at line 92 of file WickSymmetry.hpp.

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

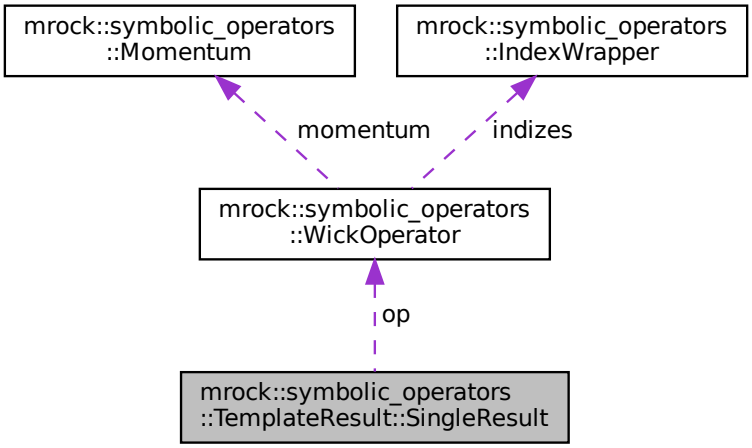
- include/mrock/symbolic_operators/WickSymmetry.hpp

7.13 mrock::symbolic_operators::TemplateResult::SingleResult Struct Reference

A structure for storing a single result.

```
#include <WickOperatorTemplate.hpp>
```

Collaboration diagram for mrock::symbolic_operators::TemplateResult::SingleResult:



Public Member Functions

- void clear_delta_equals_one ()

  Clears KroneckerDelta objects that are one, i.e., delta_{a,a} = 1.
- bool contains_impossible_delta () const

  Checks if the result contains an impossible delta, e.g., delta_{down,up}.

Public Attributes

- int factor {}

  The factor of the result.
- WickOperator op

  The Wick operator.
- std::vector< KroneckerDelta< Index > > index_deltas

  The index deltas.

7.13.1 Detailed Description

A structure for storing a single result.

Definition at line 37 of file WickOperatorTemplate.hpp.

7.13.2 Member Function Documentation

7.13.2.1 clear_delta_equals_one()

```
void mrock::symbolic_operators::TemplateResult::SingleResult::clear_delta_equals_one ( )  [inline]
```

Clears KroneckerDelta objects that are one, i.e., delta_{a,a} = 1.

Definition at line 190 of file WickOperatorTemplate.hpp.

7.13.2.2 contains_impossible_delta()

```
bool mrock::symbolic_operators::TemplateResult::SingleResult::contains_impossible_delta ( )
const  [inline]
```

Checks if the result contains an impossible delta, e.g., delta_{down,up}.

Returns

true if the result contains an impossible delta and false otherwise.

Definition at line 196 of file WickOperatorTemplate.hpp.

7.13.3 Member Data Documentation

7.13.3.1 factor

```
int mrock::symbolic_operators::TemplateResult::SingleResult::factor {}
```

The factor of the result.

Definition at line 38 of file WickOperatorTemplate.hpp.

7.13.3.2 index_deltas

```
std::vector<KroneckerDelta<Index> > mrock::symbolic_operators::TemplateResult::SingleResult↩
::index_deltas
```

The index deltas.

Definition at line 40 of file WickOperatorTemplate.hpp.

7.13.3.3 op

```
WickOperator mrock::symbolic_operators::TemplateResult::SingleResult::op
```

The Wick operator.

Definition at line 39 of file WickOperatorTemplate.hpp.

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

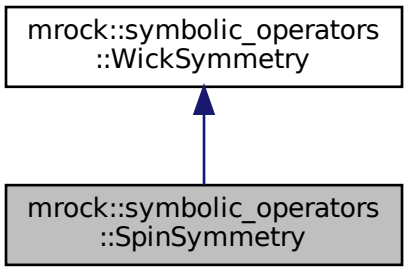
- include/mrock/symbolic_operators/WickOperatorTemplate.hpp

7.14 mrock::symbolic_operators::SpinSymmetry Class Reference

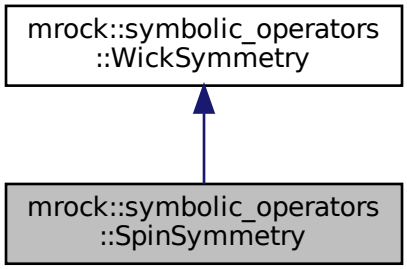
A symmetry where expectation values for spin up and down are the same.

```
#include <WickSymmetry.hpp>
```

Inheritance diagram for mrock::symbolic_operators::SpinSymmetry:



Collaboration diagram for mrock::symbolic_operators::SpinSymmetry:



Public Member Functions

- void apply_to (WickTerm &term) const override

  Applies the spin symmetry to a Wick term.

7.14.1 Detailed Description

A symmetry where expectation values for spin up and down are the same.

Definition at line 57 of file WickSymmetry.hpp.

7.14.2 Member Function Documentation

7.14.2.1 apply_to()

```
void mrock::symbolic_operators::SpinSymmetry::apply_to (
            WickTerm & term ) const  [override], [virtual]
```

Applies the spin symmetry to a Wick term.

Parameters

term	The Wick term to apply the symmetry to.

Implements mrock::symbolic_operators::WickSymmetry.

Definition at line 5 of file WickSymmetry.cpp.

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

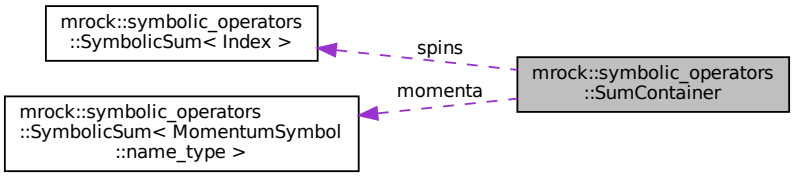
- include/mrock/symbolic_operators/WickSymmetry.hpp
- sources/WickSymmetry.cpp

7.15 mrock::symbolic_operators::SumContainer Struct Reference

A container for holding symbolic sums of momenta and spins.

```
#include <SumContainer.hpp>
```

Collaboration diagram for mrock::symbolic_operators::SumContainer:



Public Member Functions

- template<class Archive >
  void serialize (Archive &ar, const unsigned int version)

  Serializes the SumContainer object.
- SumContainer & append (const SumContainer &other)

  Appends another SumContainer to this one.
- SumContainer & append (const MomentumSum &other)

  Appends a MomentumSum to this SumContainer.
- SumContainer & append (const IndexSum &other)

  Appends an IndexSum to this SumContainer.
- void push_back (const MomentumSymbol::name_type momentum)

  Pushes back a momentum into the momenta container.
- void push_back (const Index spin)

  Pushes back a spin into the spins container.
- bool has_momentum () const noexcept

  Checks if the container has any momenta.
- bool has_spins () const noexcept

  Checks if the container has any spins.

Public Attributes

- MomentumSum momenta

  Container for momentum sums.
- IndexSum spins

  Container for spin sums.

7.15.1 Detailed Description

A container for holding symbolic sums of momenta and spins.

Sums are contained within the `SumContainer` class. It hosts both sums of momenta and sums of spins, each one is accessible via the appropriate class member and its `operator`[], e.g., `container.momenta`[i].

See also

Index, MomentumSymbol, MomentumSymbol::name_type

Definition at line 36 of file SumContainer.hpp.

7.15.2 Member Function Documentation

7.15.2.1 append() [1/3]

```
SumContainer & mrock::symbolic_operators::SumContainer::append (
            const IndexSum & other )
```

Appends an IndexSum to this SumContainer.

Parameters

other	The IndexSum to append.

Returns

Reference to this SumContainer.

Definition at line 17 of file SumContainer.cpp.

7.15.2.2 append() [2/3]

```
SumContainer & mrock::symbolic_operators::SumContainer::append (
            const MomentumSum & other )
```

Appends a MomentumSum to this SumContainer.

Parameters

other	The MomentumSum to append.

Returns

Reference to this SumContainer.

Definition at line 11 of file SumContainer.cpp.

7.15.2.3 append() [3/3]

```
SumContainer & mrock::symbolic_operators::SumContainer::append (
            const SumContainer & other )
```

Appends another SumContainer to this one.

Parameters

other	The other SumContainer to append.

Returns

Reference to this SumContainer.

Definition at line 4 of file SumContainer.cpp.

7.15.2.4 has_momentum()

```
bool mrock::symbolic_operators::SumContainer::has_momentum ( ) const  [inline], [noexcept]
```

Checks if the container has any momenta.

Returns

True if the container has momenta, false otherwise.

Definition at line 136 of file SumContainer.hpp.

7.15.2.5 has_spins()

```
bool mrock::symbolic_operators::SumContainer::has_spins ( ) const  [inline], [noexcept]
```

Checks if the container has any spins.

Returns

True if the container has spins, false otherwise.

Definition at line 139 of file SumContainer.hpp.

7.15.2.6 push_back() [1/2]

```
void mrock::symbolic_operators::SumContainer::push_back (
            const Index spin )  [inline]
```

Pushes back a spin into the spins container.

Parameters

spin	The spin to push back.

Definition at line 133 of file SumContainer.hpp.

7.15.2.7 push_back() [2/2]

```
void mrock::symbolic_operators::SumContainer::push_back (
            const MomentumSymbol::name_type momentum )  [inline]
```

Pushes back a momentum into the momenta container.

Parameters

momentum	The momentum to push back.

Definition at line 130 of file SumContainer.hpp.

7.15.2.8 serialize()

```
template<class Archive >
void mrock::symbolic_operators::SumContainer::serialize (
            Archive & ar,
            const unsigned int version )  [inline]
```

Serializes the SumContainer object.

Template Parameters

Archive	The type of the archive.

Parameters

ar	The archive to serialize to.
version	The version of the serialization.

Definition at line 47 of file SumContainer.hpp.

7.15.3 Member Data Documentation

7.15.3.1 momenta

`MomentumSum mrock::symbolic_operators::SumContainer::momenta`

Container for momentum sums.

Definition at line 37 of file SumContainer.hpp.

7.15.3.2 spins

`IndexSum mrock::symbolic_operators::SumContainer::spins`

Container for spin sums.

Definition at line 38 of file SumContainer.hpp.

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

- include/mrock/symbolic_operators/SumContainer.hpp
- sources/SumContainer.cpp

7.16 mrock::symbolic_operators::SymbolicSum< SumIndex > Struct Template Reference

A struct representing a symbolic summation operation.

`#include <SymbolicSum.hpp>`

Public Member Functions

- template<class Archive >
  void serialize (Archive &ar, const unsigned int version)
  Serializes the SymbolicSum object.
- SymbolicSum ()=default
  Default constructor.
- SymbolicSum (SumIndex sum_index)
  Constructs a SymbolicSum with a single summation index.
- SymbolicSum (const std::vector< SumIndex > &_indizes)
  Constructs a SymbolicSum with a vector of summation indices.
- SymbolicSum (std::vector< SumIndex > &&_indizes)
  Constructs a SymbolicSum with a moved vector of summation indices.
- SymbolicSum (std::initializer_list< SumIndex > init)
  Constructs a SymbolicSum with an initializer list of summation indices.
- bool is_summed_over (SumIndex what) const
  Checks if a given index is part of the summation indices.
- VECTOR_WRAPPER_FILL_MEMBERS (SumIndex, summations)
- auto operator<=> (const SymbolicSum< SumIndex > &rhs) const =default
  Compares two SymbolicSum objects.

Public Attributes

- std::vector< SumIndex > summations

  The vector of summation indices.

7.16.1 Detailed Description

template<class SumIndex>
struct mrock::symbolic_operators::SymbolicSum< SumIndex >

A struct representing a symbolic summation operation.

Template Parameters

SumIndex	The type of the summation index.

Definition at line 22 of file SymbolicSum.hpp.

7.16.2 Constructor & Destructor Documentation

7.16.2.1 SymbolicSum() [1/5]

```
template<class SumIndex >
mrock::symbolic_operators::SymbolicSum< SumIndex >::SymbolicSum ( )  [default]
```

Default constructor.

7.16.2.2 SymbolicSum() [2/5]

```
template<class SumIndex >
mrock::symbolic_operators::SymbolicSum< SumIndex >::SymbolicSum (
            SumIndex sum_index )  [inline]
```

Constructs a SymbolicSum with a single summation index.

Parameters

sum_index	The summation index.

Definition at line 47 of file SymbolicSum.hpp.

7.16.2.3 SymbolicSum() [3/5]

```
template<class SumIndex >
mrock::symbolic_operators::SymbolicSum< SumIndex >::SymbolicSum (
            const std::vector< SumIndex > & _indizes )  [inline]
```

Constructs a SymbolicSum with a vector of summation indices.

Parameters

_indizes	The vector of summation indices.

Definition at line 55 of file SymbolicSum.hpp.

7.16.2.4 SymbolicSum() [4/5]

```
template<class SumIndex >
mrock::symbolic_operators::SymbolicSum< SumIndex >::SymbolicSum (
            std::vector< SumIndex > && _indizes )  [inline]
```

Constructs a SymbolicSum with a moved vector of summation indices.

Parameters

_indizes	The vector of summation indices to move.

Definition at line 63 of file SymbolicSum.hpp.

7.16.2.5 SymbolicSum() [5/5]

```
template<class SumIndex >
mrock::symbolic_operators::SymbolicSum< SumIndex >::SymbolicSum (
            std::initializer_list< SumIndex > init )  [inline]
```

Constructs a SymbolicSum with an initializer list of summation indices.

Parameters

init	The initializer list of summation indices.

Definition at line 71 of file SymbolicSum.hpp.

7.16.3 Member Function Documentation

7.16.3.1 is_summed_over()

```
template<class SumIndex >
bool mrock::symbolic_operators::SymbolicSum< SumIndex >::is_summed_over (
            SumIndex what ) const  [inline]
```

Checks if a given index is part of the summation indices.

Parameters

what	The index to check.

Returns

True if the index is part of the summation indices, false otherwise.

Definition at line 80 of file SymbolicSum.hpp.

7.16.3.2 operator<=>()

```
template<class SumIndex >
auto mrock::symbolic_operators::SymbolicSum< SumIndex >::operator<=> (
            const SymbolicSum< SumIndex > & rhs ) const  [inline], [default]
```

Compares two SymbolicSum objects.

Parameters

rhs	The other SymbolicSum to compare with.

Returns

The result of the comparison.

7.16.3.3 serialize()

```
template<class SumIndex >
template<class Archive >
void mrock::symbolic_operators::SymbolicSum< SumIndex >::serialize (
            Archive & ar,
            const unsigned int version )  [inline]
```

Serializes the SymbolicSum object.

Template Parameters

Archive	The type of the archive.

Parameters

ar	The archive to serialize to.
version	The version of the serialization format.

Definition at line 33 of file SymbolicSum.hpp.

7.16.3.4 VECTOR_WRAPPER_FILL_MEMBERS()

```
template<class SumIndex >
mrock::symbolic_operators::SymbolicSum< SumIndex >::VECTOR_WRAPPER_FILL_MEMBERS (
            SumIndex ,
            summations  )
```

7.16.4 Member Data Documentation

7.16.4.1 summations

```
template<class SumIndex >
std::vector<SumIndex> mrock::symbolic_operators::SymbolicSum< SumIndex >::summations
```

The vector of summation indices.

Definition at line 23 of file SymbolicSum.hpp.

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

- include/mrock/symbolic_operators/SymbolicSum.hpp

7.17 sym_op_test::SymOpTest Struct Reference

```
#include <compare_test.hpp>
```

Public Member Functions

- SymOpTest (const std::string _compare_dir)
- template<class TestClass >
  bool perform_comparison (const TestClass &A, const TestClass &B)
- template<class TestClass >
  bool load_and_test (const std::string &name, const TestClass &computed)
- template<class TestClass >
  void save_as_comparison (const std::string &name, const TestClass &correct)
- int perform_test (const std::vector< Term > &H, const std::vector< Term > &base_term, const std::vector< WickOperatorTemplate > &templates, const std::vector< std::unique_ptr< WickSymmetry >> &symmetries, const bool is_baseline)

Public Attributes

- const std::string COMPARE_DIR

7.17.1 Detailed Description

Definition at line 13 of file compare_test.hpp.

7.17.2 Constructor & Destructor Documentation

7.17.2.1 SymOpTest()

```
sym_op_test::SymOpTest::SymOpTest (
            const std::string _compare_dir )  [inline]
```

Definition at line 16 of file compare_test.hpp.

7.17.3 Member Function Documentation

7.17.3.1 load_and_test()

```
template<class TestClass >
bool sym_op_test::SymOpTest::load_and_test (
            const std::string & name,
            const TestClass & computed )  [inline]
```

Definition at line 35 of file compare_test.hpp.

7.17.3.2 perform_comparison()

```
template<class TestClass >
bool sym_op_test::SymOpTest::perform_comparison (
            const TestClass & A,
            const TestClass & B )  [inline]
```

Definition at line 19 of file compare_test.hpp.

7.17.3.3 perform_test()

```
int sym_op_test::SymOpTest::perform_test (
            const std::vector< Term > & H,
            const std::vector< Term > & base_term,
            const std::vector< WickOperatorTemplate > & templates,
            const std::vector< std::unique_ptr< WickSymmetry >> & symmetries,
            const bool is_baseline )  [inline]
```

Definition at line 64 of file compare_test.hpp.

7.17.3.4 save_as_comparison()

```
template<class TestClass >
void sym_op_test::SymOpTest::save_as_comparison (
            const std::string & name,
            const TestClass & correct )  [inline]
```

Definition at line 55 of file compare_test.hpp.

7.17.4 Member Data Documentation

7.17.4.1 COMPARE_DIR

```
const std::string sym_op_test::SymOpTest::COMPARE_DIR
```

Definition at line 14 of file compare_test.hpp.

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

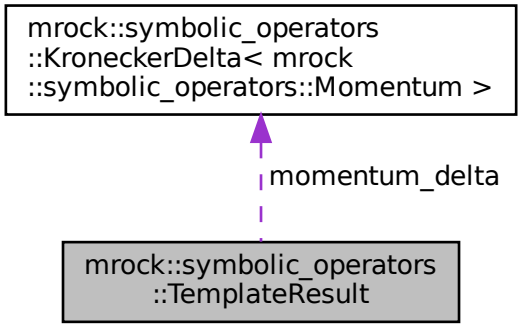
- tests/compare_test.hpp

7.18 mrock::symbolic_operators::TemplateResult Struct Reference

A structure for storing the result of a template operation.

`#include <WickOperatorTemplate.hpp>`

Collaboration diagram for mrock::symbolic_operators::TemplateResult:



Classes

- struct SingleResult

  A structure for storing a single result.

Public Member Functions

- TemplateResult ()=default

  Default constructor for TemplateResult.
- TemplateResult (size_t initial_size, OperatorType operator_type, const Momentum &base_momentum)

  Constructs a TemplateResult object.
- template<class UnaryOperation >
  void operation_on_range (const UnaryOperation &operation, size_t begin, size_t n)

  Applies an operation on a range of results.
- template<class UnaryOperation >
  void operation_on_each (const UnaryOperation &operation)

  Applies an operation on each result.
- void add_index_delta_range (const KroneckerDelta< Index > &index, size_t begin, size_t n)

  Adds an index delta to a range of results.
- void add_index_delta (const KroneckerDelta< Index > &index)

  Adds an index delta to each result.
- size_t create_branch ()

  Creates a branch in the results vector.
- void clear_impossible ()

  Clears impossible results.
- void clean_up ()

  Cleans up the results by clearing deltas that are one and removing impossible results.
- operator bool () const

  Checks if the TemplateResult is valid.

Static Public Member Functions

- static TemplateResult null_result ()

  Creates a null TemplateResult.

Public Attributes

- std::vector< SingleResult > results

  The vector of single results.
- KroneckerDelta< Momentum > momentum_delta

  The momentum delta.

7.18.1 Detailed Description

A structure for storing the result of a template operation.

Definition at line 31 of file WickOperatorTemplate.hpp.

7.18.2 Constructor & Destructor Documentation

7.18.2.1 TemplateResult() [1/2]

```
mrock::symbolic_operators::TemplateResult::TemplateResult ( )  [default]
```

Default constructor for TemplateResult.

7.18.2.2 TemplateResult() [2/2]

```
mrock::symbolic_operators::TemplateResult::TemplateResult (
            size_t initial_size,
            OperatorType operator_type,
            const Momentum & base_momentum )
```

Constructs a TemplateResult object.

Parameters

initial_size	The initial size of the results vector.
operator_type	The type of the operator.
base_momentum	The base momentum.

Definition at line 9 of file WickOperatorTemplate.cpp.

7.18.3 Member Function Documentation

7.18.3.1 add_index_delta()

```
void mrock::symbolic_operators::TemplateResult::add_index_delta (
            const KroneckerDelta< Index > & index )  [inline]
```

Adds an index delta to each result.

Parameters

index	The index delta to add.

Definition at line 205 of file WickOperatorTemplate.hpp.

7.18.3.2 add_index_delta_range()

```
void mrock::symbolic_operators::TemplateResult::add_index_delta_range (
            const KroneckerDelta< Index > & index,
            size_t begin,
            size_t n )  [inline]
```

Adds an index delta to a range of results.

Parameters

index	The index delta to add.
begin	The beginning of the range.
n	The number of elements in the range.

Definition at line 202 of file WickOperatorTemplate.hpp.

7.18.3.3 clean_up()

```
void mrock::symbolic_operators::TemplateResult::clean_up ( )
```

Cleans up the results by clearing deltas that are one and removing impossible results.

Definition at line 32 of file WickOperatorTemplate.cpp.

7.18.3.4 clear_impossible()

```
void mrock::symbolic_operators::TemplateResult::clear_impossible ( )
```

Clears impossible results.

Definition at line 26 of file WickOperatorTemplate.cpp.

7.18.3.5 create_branch()

```
size_t mrock::symbolic_operators::TemplateResult::create_branch ( )
```

Creates a branch in the results vector.

Returns

size_t The size of the current results vector.

Definition at line 20 of file WickOperatorTemplate.cpp.

7.18.3.6 null_result()

```
static TemplateResult mrock::symbolic_operators::TemplateResult::null_result ( )  [inline],
[static]
```

Creates a null TemplateResult.

Returns

TemplateResult A null TemplateResult.

Definition at line 77 of file WickOperatorTemplate.hpp.

7.18.3.7 operation_on_each()

```
template<class UnaryOperation >
void mrock::symbolic_operators::TemplateResult::operation_on_each (
            const UnaryOperation & operation )  [inline]
```

Applies an operation on each result.

Template Parameters

UnaryOperation	The type of the operation.

Parameters

operation	The operation to apply.

Definition at line 102 of file WickOperatorTemplate.hpp.

7.18.3.8 operation_on_range()

```
template<class UnaryOperation >
void mrock::symbolic_operators::TemplateResult::operation_on_range (
            const UnaryOperation & operation,
            size_t begin,
            size_t n )  [inline]
```

Applies an operation on a range of results.

Template Parameters

UnaryOperation	The type of the operation.

Parameters

operation	The operation to apply.
begin	The beginning of the range.
n	The number of elements in the range.

Definition at line 88 of file WickOperatorTemplate.hpp.

7.18.3.9 operator bool()

```
mrock::symbolic_operators::TemplateResult::operator bool ( ) const  [inline], [explicit]
```

Checks if the TemplateResult is valid.

Returns

true if the TemplateResult is valid and false otherwise.

Definition at line 147 of file WickOperatorTemplate.hpp.

7.18.4 Member Data Documentation

7.18.4.1 momentum_delta

```
KroneckerDelta<Momentum> mrock::symbolic_operators::TemplateResult::momentum_delta
```

The momentum delta.

Definition at line 56 of file WickOperatorTemplate.hpp.

7.18.4.2 results

```
std::vector<SingleResult> mrock::symbolic_operators::TemplateResult::results
```

The vector of single results.

Definition at line 55 of file WickOperatorTemplate.hpp.

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

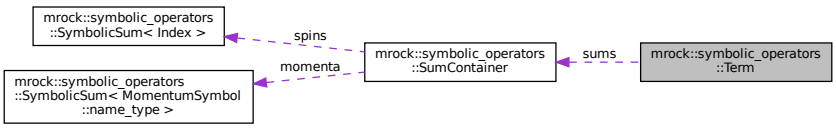
- include/mrock/symbolic_operators/WickOperatorTemplate.hpp
- sources/WickOperatorTemplate.cpp

7.19 mrock::symbolic_operators::Term Class Reference

Represents a term in symbolic operator expressions.

```
#include <Term.hpp>
```

Collaboration diagram for mrock::symbolic_operators::Term:



Public Member Functions

- template<class Archive >
  void serialize (Archive &ar, const unsigned int version)

  Serializes the term.
- Term (IntFractional _multiplicity, std::vector< Coefficient > _coefficients, const SumContainer &_sums, const std::vector< Operator > &_operators=std::vector< Operator >())

  Constructs a Term with a summation over momenta and spins and multiple coefficients.
- Term (IntFractional _multiplicity, Coefficient _coefficient, const SumContainer &_sums, const std::vector< Operator > &_operators=std::vector< Operator >())

  Constructs a Term with a summation over momenta and spins and a coefficient.
- Term (IntFractional _multiplicity, Coefficient _coefficient, const MomentumSum &_sum_momenta, const std::vector< Operator > &_operators=std::vector< Operator >())

  Constructs a Term with a summation over momenta and a coefficient.
- Term (IntFractional _multiplicity, Coefficient _coefficient, const IndexSum &_sum_spins, const std::vector< Operator > &_operators=std::vector< Operator >())

  Constructs a Term with a summation over spins (or other indizes) and a coefficient.
- Term (IntFractional _multiplicity, Coefficient _coefficient, const std::vector< Operator > &_operators=std::vector< Operator >())

  Constructs a Term with a coefficient.
- Term (IntFractional _multiplicity, const SumContainer &_sums, const std::vector< Operator > &_operators=std::vector< Operator >())

  Constructs a Term with a summation over momenta and indizes.
- Term (IntFractional _multiplicity, const MomentumSum &_sum_momenta, const std::vector< Operator > &_operators=std::vector< Operator >())

  Constructs a Term with a summation over momenta.
- Term (IntFractional _multiplicity, const IndexSum &_sum_spins, const std::vector< Operator > &_operators=std::vector< Operator >())

  Constructs a Term with a summation over spins (or other indizes)
- Term (IntFractional _multiplicity, const std::vector< Operator > &_operators=std::vector< Operator >())

  Constructs a Term with only a multiplicity.
- Term ()=default

  Default constructor.
- bool is_identity () const

  Checks if the term is an identity.
- bool contains_boson () const

  Checks if the term contains a boson.
- bool contains_fermion () const

  Checks if the term contains a fermion.
- int count_bosons () const

  Counts the number of bosons in the term.
- int count_fermions () const

  Counts the number of fermions in the term.
- void print () const

  Prints the term.
- void flip_sign ()

  Flips the sign of the term.
- void perform_operator_swap (Operator &lhs, Operator &rhs)

  Swaps two operators in the term. Does NOT consider possible additional terms spawned by this operation due to non-commutivity!
- const std::vector< Operator > & get_operators () const

  Gets the operators in the term.

- bool set_deltas ()

  Sets the Kronecker deltas in the term.
- bool compute_sums ()

  Computes the sums in the term.
- void discard_zero_momenta ()

  Discards zero momenta in the term.
- void sort ()

  Sorts the term.
- void rename_sums ()

  Renames the sum indices in the term.
- bool is_equal (const Term &other) const

  Checks if the term is equal to another term (excluding multiplicity).
- bool is_normal_ordered () const

  Checks if the term is in normal order.
- std::string to_string_without_prefactor () const

  Converts the term to a string without the prefactor.
- Term & hermitian_conjugate_inplace ()

  Applies the Hermitian conjugate to the term.
- Term hermitian_conjugate () const

  Creates hermitian conjugate of this as a new object.
- void rename_indizes (const Index what, const Index to)

  Renames indices in the term.
- void rename_momenta (const MomentumSymbol::name_type what, const MomentumSymbol::name_type to)

  Renames momenta in the term.
- void swap_momenta (const MomentumSymbol::name_type a, const MomentumSymbol::name_type b)

  Swaps two momenta in the term.
- void transform_momentum_sum (const MomentumSymbol::name_type what, const Momentum to, const MomentumSymbol::name_type new_sum_index)

  Transforms a momentum sum in the term.
- void invert_momentum (const MomentumSymbol::name_type what)

  Inverts a momentum in the term.
- void invert_momentum_sum (const MomentumSymbol::name_type what)

  Inverts a momentum sum in the term.
- void remove_momentum_contribution (const MomentumSymbol::name_type value)

  Removes a momentum contribution from the term.

Public Attributes

- std::vector< Coefficient > coefficients

  Coefficients of the term.
- SumContainer sums

  Sum container for the term. Contains e.g. \sum_{k,l} \sum_{sigma}.
- std::vector< Operator > operators

  Operators in the term, if empty the term is considered to contain the identiy operator.
- std::vector< KroneckerDelta< Momentum > > delta_momenta

  Kronecker delta for momenta.
- std::vector< KroneckerDelta< Index > > delta_indizes

  Kronecker delta for indices.
- IntFractional multiplicity

  Multiplicity of the term.
- _TERM_TRACKER_ATTRIBUTE

  Attribute for tracking terms (if enabled).

Friends

- struct WickTerm
- void normal_order (std::vector< Term > &terms)

  Normal orders the terms by using the canoncical (anti-)commutation relations The result is stored in the input vector. A simple example is $bb^\dagger = 1 \pm b^\dagger b$, where the + applies to bosons and the minus to fermions.
- std::vector< Term > commutator (const Term &left, const Term &right)

  Computes the commutator of two terms: $[A, B] = AB - BA$.
- std::ostream & operator<< (std::ostream &os, const Term &term)

  Overloads the stream insertion operator for the Term class.

7.19.1 Detailed Description

Represents a term in symbolic operator expressions.

This class represents a Term. It has various kind of constructors that allow setting coefficient(s), sums, operators and deltas. Using IntFractional, the term can have rational prefactors, e.g., 1/2.

A Hamiltonian (or any other summation of operators) is characterized as `std::vector<Term>`. It can consist of any number of individual terms. For a few practical examples, see the files in the tests folder. See bosons.cpp, continuum.cpp, and compare_test.hpp. My own projects using this library are, e.g., https://github.com/majesticrock/FermionCommute and https://github.com/majesticrock/FlowCommutators.

After creating atleast two `Terms` (or `std::vector<Term>`), you may commute them by calling

```
std::vector<Term> result = commutator(A, B);
clean_up(result);
```

After calling the commutator, you should pretty much always call mrock::symbolic_operators::clean_up(std::vector<Term>) because commutator performs the normal ordering procedure, however, does not attempt to beautify the result. clean_up then sorts the terms, adds identical ones together and removes those that are equal to 0.

Similarly, a double commutator $[C, [A, B]]$ can be evaluated by

```
std::vector<Term> inner_result = commutator(A, B);
clean_up(inner_result);
std::vector<Term> result = commutator(C, inner_result);
clean_up(result);
```

To output the results, an overload of `operator<<` is provided for both `Term` and `std::vector<Term>`. The out put is formatted so that it can be used within an align-environment within LaTeX.

See also

Coefficient, SumContainer, Operator, KroneckerDelta

Definition at line 69 of file Term.hpp.

7.19.2 Constructor & Destructor Documentation

7.19.2.1 Term() [1/10]

```
mrock::symbolic_operators::Term::Term (
            IntFractional _multiplicity,
            std::vector< Coefficient > _coefficients,
            const SumContainer & _sums,
            const std::vector< Operator > & _operators = std::vector<Operator>() )
```

Constructs a Term with a summation over momenta and spins and multiple coefficients.

Parameters

_multiplicity	The _multiplicity of the term
_coefficients	The coefficients
_sums	The sums
_operators	The operators of the term

Definition at line 7 of file Term.cpp.

7.19.2.2 Term() [2/10]

```
mrock::symbolic_operators::Term::Term (
            IntFractional _multiplicity,
            Coefficient _coefficient,
            const SumContainer & _sums,
            const std::vector< Operator > & _operators = std::vector<Operator>() )
```

Constructs a Term with a summation over momenta and spins and a coefficient.

Parameters

_multiplicity	The _multiplicity of the term
_coefficient	The coefficient
_sums	The sums
_operators	The operators of the term

Definition at line 9 of file Term.cpp.

7.19.2.3 Term() [3/10]

```
mrock::symbolic_operators::Term::Term (
            IntFractional _multiplicity,
            Coefficient _coefficient,
            const MomentumSum & _sum_momenta,
            const std::vector< Operator > & _operators = std::vector<Operator>() )
```

Constructs a Term with a summation over momenta and a coefficient.

Parameters

_multiplicity	The _multiplicity of the term
_coefficient	The coefficient
_sum_momenta	Sum over momenta
_operators	The operators of the term

Definition at line 11 of file Term.cpp.

7.19.2.4 Term() [4/10]

```
mrock::symbolic_operators::Term::Term (
            IntFractional _multiplicity,
            Coefficient _coefficient,
            const IndexSum & _sum_spins,
            const std::vector< Operator > & _operators = std::vector<Operator>() )
```

Constructs a Term with a summation over spins (or other indizes) and a coefficient.

Parameters

_multiplicity	The _multiplicity of the term
_coefficient	The coefficient
_sum_spins	Sum over spins (or other indizes)
_operators	The operators of the term

Definition at line 13 of file Term.cpp.

7.19.2.5 Term() [5/10]

```
mrock::symbolic_operators::Term::Term (
            IntFractional _multiplicity,
            Coefficient _coefficient,
            const std::vector< Operator > & _operators = std::vector<Operator>() )
```

Constructs a Term with a coefficient.

Parameters

_multiplicity	The _multiplicity of the term
_coefficient	The coefficient
_operators	The operators of the term

Definition at line 15 of file Term.cpp.

7.19.2.6 Term() [6/10]

```
mrock::symbolic_operators::Term::Term (
            IntFractional _multiplicity,
```

```
const SumContainer & _sums,
const std::vector< Operator > & _operators = std::vector<Operator>() )
```

Constructs a Term with a summation over momenta and indizes.

Parameters

_multiplicity	The _multiplicity of the term
_sums	Sums
_operators	The operators of the term

Definition at line 17 of file Term.cpp.

7.19.2.7 Term() [7/10]

```
mrock::symbolic_operators::Term::Term (
            IntFractional _multiplicity,
            const MomentumSum & _sum_momenta,
            const std::vector< Operator > & _operators = std::vector<Operator>() )
```

Constructs a Term with a summation over momenta.

Parameters

_multiplicity	The _multiplicity of the term
_sum_momenta	Sum over momenta
_operators	The operators of the term

Definition at line 19 of file Term.cpp.

7.19.2.8 Term() [8/10]

```
mrock::symbolic_operators::Term::Term (
            IntFractional _multiplicity,
            const IndexSum & _sum_spins,
            const std::vector< Operator > & _operators = std::vector<Operator>() )
```

Constructs a Term with a summation over spins (or other indizes)

Parameters

_multiplicity	The _multiplicity of the term
_sum_spins	Sum over spins (or other indizes)
_operators	The operators of the term

Definition at line 21 of file Term.cpp.

7.19.2.9 Term() [9/10]

```
mrock::symbolic_operators::Term::Term (
            IntFractional _multiplicity,
            const std::vector< Operator > & _operators = std::vector<Operator>() )  [explicit]
```

Constructs a Term with only a multiplicity.

Parameters

_multiplicity	The _multiplicity of the term
_operators	The operators of the term

Definition at line 23 of file Term.cpp.

7.19.2.10 Term() [10/10]

```
mrock::symbolic_operators::Term::Term ( )  [default]
```

Default constructor.

7.19.3 Member Function Documentation

7.19.3.1 compute_sums()

```
bool mrock::symbolic_operators::Term::compute_sums ( )
```

Computes the sums in the term.

Returns

True if successful, false otherwise.

Definition at line 186 of file Term.cpp.

7.19.3.2 contains_boson()

```
bool mrock::symbolic_operators::Term::contains_boson ( ) const  [inline]
```

Checks if the term contains a boson.

Returns

True if the term contains a boson, false otherwise.

Definition at line 475 of file Term.hpp.

7.19.3.3 contains_fermion()

```
bool mrock::symbolic_operators::Term::contains_fermion ( ) const  [inline]
```

Checks if the term contains a fermion.

Returns

True if the term contains a fermion, false otherwise.

Definition at line 478 of file Term.hpp.

7.19.3.4 count_bosons()

```
int mrock::symbolic_operators::Term::count_bosons ( ) const  [inline]
```

Counts the number of bosons in the term.

Returns

The number of bosons.

Definition at line 481 of file Term.hpp.

7.19.3.5 count_fermions()

```
int mrock::symbolic_operators::Term::count_fermions ( ) const  [inline]
```

Counts the number of fermions in the term.

Returns

The number of fermions.

Definition at line 484 of file Term.hpp.

7.19.3.6 discard_zero_momenta()

```
void mrock::symbolic_operators::Term::discard_zero_momenta ( )
```

Discards zero momenta in the term.

Definition at line 273 of file Term.cpp.

7.19.3.7 flip_sign()

```
void mrock::symbolic_operators::Term::flip_sign ( )  [inline]
```

Flips the sign of the term.

Definition at line 487 of file Term.hpp.

7.19.3.8 get_operators()

```
const std::vector< Operator > & mrock::symbolic_operators::Term::get_operators ( ) const
[inline]
```

Gets the operators in the term.

Returns

The operators.

Definition at line 496 of file Term.hpp.

7.19.3.9 hermitian_conjugate()

```
Term mrock::symbolic_operators::Term::hermitian_conjugate ( ) const
```

Creates hermitian conjugate of this as a new object.

Returns

Returns the new object.

Definition at line 500 of file Term.cpp.

7.19.3.10 hermitian_conjugate_inplace()

```
Term & mrock::symbolic_operators::Term::hermitian_conjugate_inplace ( )
```

Applies the Hermitian conjugate to the term.

Returns

A reference to ∗this

Definition at line 489 of file Term.cpp.

7.19.3.11 invert_momentum()

```
void mrock::symbolic_operators::Term::invert_momentum (
            const MomentumSymbol::name_type what )
```

Inverts a momentum in the term.

Parameters

what	The momentum to invert.

Definition at line 576 of file Term.cpp.

7.19.3.12 invert_momentum_sum()

```
void mrock::symbolic_operators::Term::invert_momentum_sum (
            const MomentumSymbol::name_type what )
```

Inverts a momentum sum in the term.

Parameters

what	The momentum to invert.

Definition at line 585 of file Term.cpp.

7.19.3.13 is_equal()

```
bool mrock::symbolic_operators::Term::is_equal (
            const Term & other ) const
```

Checks if the term is equal to another term (excluding multiplicity).

Parameters

other	The other term.

Returns

True if equal, false otherwise.

Definition at line 434 of file Term.cpp.

7.19.3.14 is_identity()

```
bool mrock::symbolic_operators::Term::is_identity ( ) const  [inline]
```

Checks if the term is an identity.

Returns

True if the term is an identity, false otherwise.

Definition at line 472 of file Term.hpp.

7.19.3.15 is_normal_ordered()

```
bool mrock::symbolic_operators::Term::is_normal_ordered ( ) const
```

Checks if the term is in normal order.

Returns

True if in normal order, false otherwise.

Definition at line 443 of file Term.cpp.

7.19.3.16 perform_operator_swap()

```
void mrock::symbolic_operators::Term::perform_operator_swap (
            Operator & lhs,
            Operator & rhs ) [inline]
```

Swaps two operators in the term. Does NOT consider possible additional terms spawned by this operation due to non-commutivity!

Parameters

lhs	The first operator.
rhs	The second operator.

Definition at line 490 of file Term.hpp.

7.19.3.17 print()

```
void mrock::symbolic_operators::Term::print ( ) const
```

Prints the term.

Definition at line 26 of file Term.cpp.

7.19.3.18 remove_momentum_contribution()

```
void mrock::symbolic_operators::Term::remove_momentum_contribution (
            const MomentumSymbol::name_type value )
```

Removes a momentum contribution from the term.

Parameters

value	The momentum to remove.

Definition at line 592 of file Term.cpp.

7.19.3.19 rename_indizes()

```
void mrock::symbolic_operators::Term::rename_indizes (
            const Index what,
            const Index to )
```

Renames indices in the term.

Parameters

what	The index to rename.
to	The new index.

Definition at line 506 of file Term.cpp.

7.19.3.20 rename_momenta()

```
void mrock::symbolic_operators::Term::rename_momenta (
            const MomentumSymbol::name_type what,
            const MomentumSymbol::name_type to )
```

Renames momenta in the term.

Parameters

what	The momentum to rename.
to	The new momentum.

Definition at line 532 of file Term.cpp.

7.19.3.21 rename_sums()

```
void mrock::symbolic_operators::Term::rename_sums ( )
```

Renames the sum indices in the term.

Definition at line 387 of file Term.cpp.

7.19.3.22 serialize()

```
template<class Archive >
void mrock::symbolic_operators::Term::serialize (
            Archive & ar,
            const unsigned int version )  [inline]
```

Serializes the term.

Template Parameters

Archive	The archive type.

Parameters

ar	The archive.
version	The version.

Definition at line 86 of file Term.hpp.

7.19.3.23 set_deltas()

```
bool mrock::symbolic_operators::Term::set_deltas ( )
```

Sets the Kronecker deltas in the term.

Returns

True if successful, false otherwise.

Definition at line 30 of file Term.cpp.

7.19.3.24 sort()

```
void mrock::symbolic_operators::Term::sort ( )
```

Sorts the term.

Definition at line 282 of file Term.cpp.

7.19.3.25 swap_momenta()

```
void mrock::symbolic_operators::Term::swap_momenta (
            const MomentumSymbol::name_type a,
            const MomentumSymbol::name_type b )
```

Swaps two momenta in the term.

Parameters

a	The first momentum.
b	The second momentum.

Definition at line 552 of file Term.cpp.

7.19.3.26 to_string_without_prefactor()

```
std::string mrock::symbolic_operators::Term::to_string_without_prefactor ( ) const
```

Converts the term to a string without the prefactor.

Returns

The string representation.

Definition at line 454 of file Term.cpp.

7.19.3.27 transform_momentum_sum()

```
void mrock::symbolic_operators::Term::transform_momentum_sum (
            const MomentumSymbol::name_type what,
            const Momentum to,
            const MomentumSymbol::name_type new_sum_index )
```

Transforms a momentum sum in the term.

Parameters

what	The momentum to transform.
to	The new momentum.
new_sum_index	The new sum index.

Definition at line 558 of file Term.cpp.

7.19.4 Friends And Related Function Documentation

7.19.4.1 commutator

```
std::vector<Term> commutator (
            const Term & left,
            const Term & right )  [friend]
```

Computes the commutator of two terms: $[A, B] = AB - BA$.

Parameters

left	The left term.
right	The right term.

Returns

The commutation result.

Definition at line 694 of file Term.cpp.

7.19.4.2 normal_order

```
void normal_order (
            std::vector< Term > & terms )  [friend]
```

Normal orders the terms by using the canoncical (anti-)commutation relations The result is stored in the input vector. A simple example is $bb^\dagger = 1 \pm b^\dagger b$, where the + applies to bosons and the minus to fermions.

Parameters

terms	The terms to normal order.

Definition at line 606 of file Term.cpp.

7.19.4.3 operator<<

```
std::ostream& operator<< (
            std::ostream & os,
            const Term & term )  [friend]
```

Overloads the stream insertion operator for the Term class.

Parameters

os	The output stream.
term	The Term object to insert into the stream.

Returns

The output stream.

Definition at line 731 of file Term.cpp.

7.19.4.4 WickTerm

```
friend struct WickTerm  [friend]
```

Definition at line 95 of file Term.hpp.

7.19.5 Member Data Documentation

7.19.5.1 _TERM_TRACKER_ATTRIBUTE

```
mrock::symbolic_operators::Term::_TERM_TRACKER_ATTRIBUTE
```

Attribute for tracking terms (if enabled).

Definition at line 77 of file Term.hpp.

7.19.5.2 coefficients

```
std::vector<Coefficient> mrock::symbolic_operators::Term::coefficients
```

Coefficients of the term.

Definition at line 71 of file Term.hpp.

7.19.5.3 delta_indizes

`std::vector<KroneckerDelta<Index> > mrock::symbolic_operators::Term::delta_indizes`

Kronecker delta for indices.

Definition at line 75 of file Term.hpp.

7.19.5.4 delta_momenta

`std::vector<KroneckerDelta<Momentum> > mrock::symbolic_operators::Term::delta_momenta`

Kronecker delta for momenta.

Definition at line 74 of file Term.hpp.

7.19.5.5 multiplicity

`IntFractional mrock::symbolic_operators::Term::multiplicity`

Multiplicity of the term.

Definition at line 76 of file Term.hpp.

7.19.5.6 operators

`std::vector<Operator> mrock::symbolic_operators::Term::operators`

Operators in the term, if empty the term is considered to contain the identiy operator.

Definition at line 73 of file Term.hpp.

7.19.5.7 sums

`SumContainer mrock::symbolic_operators::Term::sums`

Sum container for the term. Contains e.g. \sum_{k,l} \sum_{sigma}.

Definition at line 72 of file Term.hpp.

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

- include/mrock/symbolic_operators/Term.hpp
- sources/Term.cpp

7.20 mrock::symbolic_operators::TermLoader Struct Reference

A structure to load and manage Wick terms.

`#include <TermLoader.hpp>`

Public Member Functions

- void load (std::string const &folder, bool use_XP, int n_terms, int start_at=0)

  Loads Wick terms from a specified folder.

Public Attributes

- std::vector< WickTermCollector > M

  Vector to store Wick terms of the dynamical matrix M.
- std::vector< WickTermCollector > N

  Vector to store Wick terms of the norm matrix N.

7.20.1 Detailed Description

A structure to load and manage Wick terms.

Definition at line 17 of file TermLoader.hpp.

7.20.2 Member Function Documentation

7.20.2.1 load()

```
void mrock::symbolic_operators::TermLoader::load (
            std::string const & folder,
            bool use_XP,
            int n_terms,
            int start_at = 0 )
```

Loads Wick terms from a specified folder.

Parameters

folder	The path to the folder containing the terms.
use_XP	A boolean flag to indicate whether to use the XP basis.
n_terms	The number of terms to load.
start_at	The starting index for loading terms (default is 0).

Definition at line 7 of file TermLoader.cpp.

7.20.3 Member Data Documentation

7.20.3.1 M

```
std::vector<WickTermCollector> mrock::symbolic_operators::TermLoader::M
```

Vector to store Wick terms of the dynamical matrix M.

Definition at line 18 of file TermLoader.hpp.

7.20.3.2 N

```
std::vector<WickTermCollector> mrock::symbolic_operators::TermLoader::N
```

Vector to store Wick terms of the norm matrix N.

Definition at line 19 of file TermLoader.hpp.

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

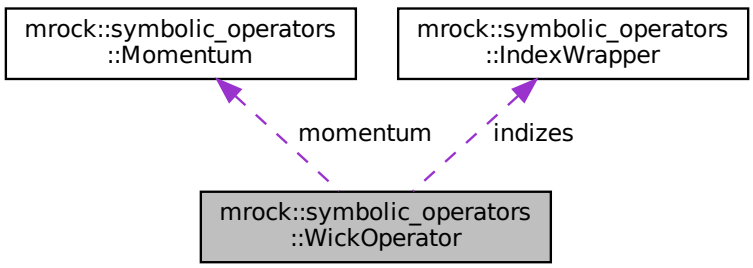
- include/mrock/symbolic_operators/TermLoader.hpp
- sources/TermLoader.cpp

7.21 mrock::symbolic_operators::WickOperator Class Reference

A structure representing a Wick operator.

```
#include <WickOperator.hpp>
```

Collaboration diagram for mrock::symbolic_operators::WickOperator:



Public Member Functions

- template<class Archive >
  void serialize (Archive &ar, const unsigned int version)

  Serializes the WickOperator object.
- WickOperator (const OperatorType &_type, const bool _is_daggered, const Momentum &_momentum, const IndexWrapper &_indizes=IndexWrapper())

  Constructs a WickOperator object.
- WickOperator (const OperatorType &_type, const bool _is_daggered, const Momentum &_momentum, const Index _index)

  Constructs a WickOperator object.
- WickOperator ()=default

  Default constructor for WickOperator.
- WickOperator (const std::string &expression)

  Constructs a WickOperator object from a string expression.
- bool uses_index (const Index index) const noexcept

  Checks if the operator uses a specific index.
- bool depends_on (const MomentumSymbol::name_type momentum) const noexcept

  Checks if the operator depends on a specific momentum.
- void remove_momentum_contribution (const MomentumSymbol::name_type value)

  Removes a momentum contribution from the operator.

Public Attributes

- OperatorType type { OperatorType::Undefined_Type }

  The type of the operator.
- bool is_daggered {}

  Indicates if the operator is daggered.
- Momentum momentum

  The momentum associated with the operator.
- IndexWrapper indizes

  The indices associated with the operator.

7.21.1 Detailed Description

A structure representing a Wick operator.

Definition at line 18 of file WickOperator.hpp.

7.21.2 Constructor & Destructor Documentation

7.21.2.1 WickOperator() [1/4]

```
mrock::symbolic_operators::WickOperator::WickOperator (
            const OperatorType & _type,
            const bool _is_daggered,
            const Momentum & _momentum,
            const IndexWrapper & _indizes = IndexWrapper() )
```

Constructs a WickOperator object.

Parameters

_type	The type of the operator.
_is_daggered	Whether the operator is daggered.
_momentum	The momentum of the operator.
_indizes	The indices of the operator.

Definition at line 6 of file WickOperator.cpp.

7.21.2.2 WickOperator() [2/4]

```
mrock::symbolic_operators::WickOperator::WickOperator (
            const OperatorType & _type,
            const bool _is_daggered,
            const Momentum & _momentum,
            const Index _index )
```

Constructs a WickOperator object.

Parameters

_type	The type of the operator.
_is_daggered	Whether the operator is daggered.
_momentum	The momentum of the operator.
_index	The index of the operator.

Definition at line 8 of file WickOperator.cpp.

7.21.2.3 WickOperator() [3/4]

```
mrock::symbolic_operators::WickOperator::WickOperator ( )  [default]
```

Default constructor for WickOperator.

7.21.2.4 WickOperator() [4/4]

```
mrock::symbolic_operators::WickOperator::WickOperator (
            const std::string & expression )
```

Constructs a WickOperator object from a string expression.

Parameters

expression	The string expression.

Definition at line 11 of file WickOperator.cpp.

7.21.3 Member Function Documentation

7.21.3.1 depends_on()

```
bool mrock::symbolic_operators::WickOperator::depends_on (
            const MomentumSymbol::name_type momentum ) const  [inline], [noexcept]
```

Checks if the operator depends on a specific momentum.

Parameters

momentum	The momentum to check.

Returns

true if the operator depends on the momentum.

false otherwise.

Definition at line 122 of file WickOperator.hpp.

7.21.3.2 remove_momentum_contribution()

```
void mrock::symbolic_operators::WickOperator::remove_momentum_contribution (
            const MomentumSymbol::name_type value )  [inline]
```

Removes a momentum contribution from the operator.

Parameters

value	The momentum value to remove.

Definition at line 125 of file WickOperator.hpp.

7.21.3.3 serialize()

```
template<class Archive >
void mrock::symbolic_operators::WickOperator::serialize (
            Archive & ar,
            const unsigned int version )  [inline]
```

Serializes the WickOperator object.

Template Parameters

Archive	The type of the archive.

Parameters

ar	The archive object.
version	The version of the serialization.

Definition at line 32 of file WickOperator.hpp.

7.21.3.4 uses_index()

```
bool mrock::symbolic_operators::WickOperator::uses_index (
            const Index index ) const  [inline], [noexcept]
```

Checks if the operator uses a specific index.

Parameters

index	The index to check.

Returns

true if the operator uses the index.

false otherwise.

Definition at line 116 of file WickOperator.hpp.

7.21.4 Member Data Documentation

7.21.4.1 indizes

```
IndexWrapper mrock::symbolic_operators::WickOperator::indizes
```

The indices associated with the operator.

Definition at line 22 of file WickOperator.hpp.

7.21.4.2 is_daggered

```
bool mrock::symbolic_operators::WickOperator::is_daggered {}
```

Indicates if the operator is daggered.

Definition at line 20 of file WickOperator.hpp.

7.21.4.3 momentum

```
Momentum mrock::symbolic_operators::WickOperator::momentum
```

The momentum associated with the operator.

Definition at line 21 of file WickOperator.hpp.

7.21.4.4 type

```
OperatorType mrock::symbolic_operators::WickOperator::type { OperatorType::Undefined_Type }
```

The type of the operator.

Definition at line 19 of file WickOperator.hpp.

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

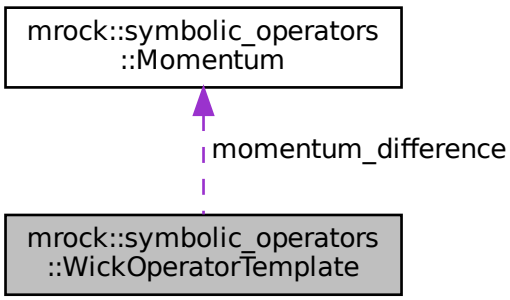
- include/mrock/symbolic_operators/WickOperator.hpp
- sources/WickOperator.cpp

7.22 mrock::symbolic_operators::WickOperatorTemplate Class Reference

A template for creating Wick operators.

```
#include <WickOperatorTemplate.hpp>
```

Collaboration diagram for mrock::symbolic_operators::WickOperatorTemplate:



Public Member Functions

- TemplateResult create_from_operators (const Operator &left, const Operator &right) const

  Creates a WickOperator from two operators if possible.

Public Attributes

- std::vector< IndexComparison > indexComparison

  The vector of index comparisons.
- Momentum momentum_difference

  The momentum difference.
- OperatorType type

  The type of the operator.
- bool is_sc_type {}

  Indicates if the operator is of SC type.

Private Member Functions

- TemplateResult _handle_sc_type (const Operator &left, const Operator &right) const

  Handles the creation of SC type operators.
- TemplateResult _handle_num_type (const Operator &left, const Operator &right) const

  Handles the creation of NUM type operators.

7.22.1 Detailed Description

A template for creating Wick operators.

Definition at line 154 of file WickOperatorTemplate.hpp.

7.22.2 Member Function Documentation

7.22.2.1 _handle_num_type()

```
TemplateResult mrock::symbolic_operators::WickOperatorTemplate::_handle_num_type (
            const Operator & left,
            const Operator & right ) const  [private]
```

Handles the creation of NUM type operators.

Parameters

left	The left operator.
right	The right operator.

Returns

TemplateResult The result of the creation.

Definition at line 80 of file WickOperatorTemplate.cpp.

7.22.2.2 _handle_sc_type()

```
TemplateResult mrock::symbolic_operators::WickOperatorTemplate::_handle_sc_type (
            const Operator & left,
            const Operator & right ) const  [private]
```

Handles the creation of SC type operators.

Parameters

left	The left operator.
right	The right operator.

Returns

TemplateResult The result of the creation.

Definition at line 39 of file WickOperatorTemplate.cpp.

7.22.2.3 create_from_operators()

```
TemplateResult mrock::symbolic_operators::WickOperatorTemplate::create_from_operators (
            const Operator & left,
            const Operator & right ) const
```

Creates a WickOperator from two operators if possible.

Parameters

left	The left operator.
right	The right operator.

Returns

TemplateResult The result of the creation.

Definition at line 113 of file WickOperatorTemplate.cpp.

7.22.3 Member Data Documentation

7.22.3.1 indexComparison

```
std::vector<IndexComparison> mrock::symbolic_operators::WickOperatorTemplate::indexComparison
```

The vector of index comparisons.

Definition at line 155 of file WickOperatorTemplate.hpp.

7.22.3.2 is_sc_type

```
bool mrock::symbolic_operators::WickOperatorTemplate::is_sc_type {}
```

Indicates if the operator is of SC type.

Definition at line 158 of file WickOperatorTemplate.hpp.

7.22.3.3 momentum_difference

```
Momentum mrock::symbolic_operators::WickOperatorTemplate::momentum_difference
```

The momentum difference.

Definition at line 156 of file WickOperatorTemplate.hpp.

7.22.3.4 type

```
OperatorType mrock::symbolic_operators::WickOperatorTemplate::type
```

The type of the operator.

Definition at line 157 of file WickOperatorTemplate.hpp.

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

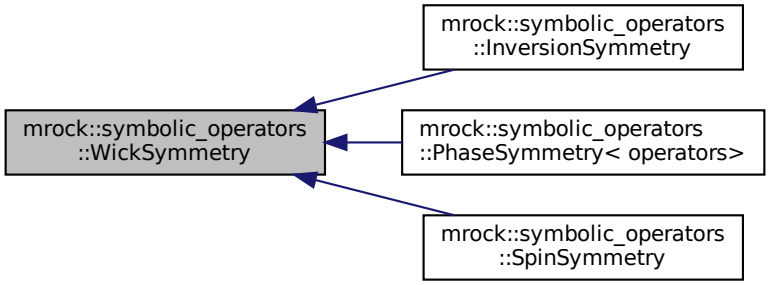
- include/mrock/symbolic_operators/WickOperatorTemplate.hpp
- sources/WickOperatorTemplate.cpp

7.23 mrock::symbolic_operators::WickSymmetry Class Reference

An abstract base class for Wick symmetries.

`#include <WickSymmetry.hpp>`

Inheritance diagram for mrock::symbolic_operators::WickSymmetry:



Public Member Functions

- virtual void apply_to (WickTerm &term) const =0

  Applies the symmetry to a Wick term.
- virtual ~WickSymmetry ()=default

  Virtual destructor for WickSymmetry.

7.23.1 Detailed Description

An abstract base class for Wick symmetries.

There may be some symmetries that simplify your results, e.g., $\langle O^\dagger \rangle = \langle O \rangle$. These symmetries can be implemented by inheriting from the `WickSymmetry` class and defining the member function `virtual void apply_to(WickTerm& term) const`. Then create a `std::vector<std::unique_ptr<WickSymmetry>> symmetries` and make use of polymorphism by calling `clean_wicks(wicks,symmetries)`. There are the following predefined symmetry operations:

SpinSymmetry
Changes all spins of the operators in `term` to $\uparrow$.

InversionSymmetry
Flips the momenta in such a way, that the first momentum in a term is always positive, i.e., $-k + l$ is changed to $k - l$ while $k - l$ would stay unmodified.

PhaseSymmetry
Takes a list of `OperatorType` as template arguments. Removes any dagger from all operators with a type from the list. Example: `PhaseSymmetry<SC_Type,CDW_Type` removes the dagger from `SC_Type` and `CDW_Type` operators.

See also

SpinSymmetry, InversionSymmetry, PhaseSymmetry

Definition at line 39 of file WickSymmetry.hpp.

7.23.2 Constructor & Destructor Documentation

7.23.2.1 ~WickSymmetry()

```
virtual mrock::symbolic_operators::WickSymmetry::~WickSymmetry ( )  [virtual], [default]
```

Virtual destructor for WickSymmetry.

7.23.3 Member Function Documentation

7.23.3.1 apply_to()

```
virtual void mrock::symbolic_operators::WickSymmetry::apply_to (
            WickTerm & term ) const  [pure virtual]
```

Applies the symmetry to a Wick term.

Parameters

term	The Wick term to apply the symmetry to.

Implemented in mrock::symbolic_operators::PhaseSymmetry< operators>, mrock::symbolic_operators::InversionSymmetry, and mrock::symbolic_operators::SpinSymmetry.

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

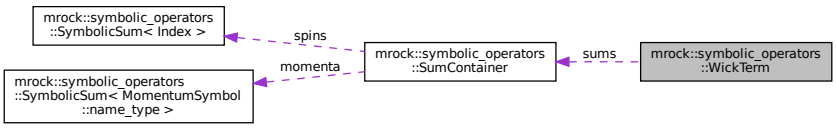
- include/mrock/symbolic_operators/WickSymmetry.hpp

7.24 mrock::symbolic_operators::WickTerm Class Reference

A structure representing a Wick term.

```
#include <WickTerm.hpp>
```

Collaboration diagram for mrock::symbolic_operators::WickTerm:



Public Member Functions

- template<class Archive >
  void serialize (Archive &ar, const unsigned int version)

  Serializes the WickTerm object.
- WickTerm (const Term *base)

  Constructs a WickTerm object from a base Term pointer.
- WickTerm (const Term &base)

  Constructs a WickTerm object from a base Term reference.
- WickTerm ()=default

  Default constructor for WickTerm.
- WickTerm (const WickTerm &base, const TemplateResult::SingleResult &result)

  Constructs a WickTerm object from a base WickTerm and a TemplateResult::SingleResult.
- WickTerm (const std::string &expression)

  Constructs a WickTerm object from a string expression.
- bool includes_type (const OperatorType operator_type) const

  Checks if the term includes a specific operator type.
- bool has_single_coefficient () const noexcept

  Checks if the term has a single coefficient.
- bool uses_index (const Index index) const noexcept

  Checks if the term uses a specific index.
- bool is_identity () const noexcept

  Checks if the term is an identity term.
- bool is_bilinear () const noexcept

  Checks if the term is bilinear.
- bool is_quartic () const noexcept

  Checks if the term is quartic.
- double get_factor () const noexcept

  Returns the multiplicity as a double.
- int which_operator_depends_on (const MomentumSymbol::name_type momentum) const noexcept

  Returns the position of the first operator that depends on a specific momentum.
- const Coefficient & get_first_coefficient () const

  Returns the first coefficient in the term.
- bool handled () const noexcept

  Checks if the term has been handled.
- bool set_deltas ()

  Sets the deltas in the term.
- bool compute_sums ()

  Computes the sums in the term.
- void discard_zero_momenta ()

  Discards zero momenta in the term.
- void rename_sums ()

  Renames the sums in the term.
- void sort ()

  Sorts the elements in the term.
- void include_template_result (const TemplateResult::SingleResult &result)

  Includes a template result in the term.
- void invert_momentum (const MomentumSymbol::name_type what)

  Inverts a momentum in the term.
- void invert_momentum_sum (const MomentumSymbol::name_type what)

  Inverts a momentum sum in the term.
- void remove_momentum_contribution (const MomentumSymbol::name_type value)

  Removes a momentum contribution from the term.

Public Attributes

- IntFractional multiplicity {}

  The multiplicity of the term.
- std::vector< Coefficient > coefficients

  The coefficients of the term.
- SumContainer sums

  The sums in the term.
- std::vector< WickOperator > operators

  The operators in the term.
- std::vector< KroneckerDelta< Momentum > > delta_momenta

  The momentum deltas.
- std::vector< KroneckerDelta< Index > > delta_indizes

  The index deltas.
- std::vector< Operator > temporary_operators

  Temporary operators used in the term.

Private Member Functions

- void string_parser (std::string &&expression)

  Parses a string expression to initialize the WickTerm.

7.24.1 Detailed Description

A structure representing a Wick term.

Prerequisite: The terms you want to apply Wick's theorem on are saved in an `std::vector<Term>`.

Applying Wick's theorem often involves omiting certain expecation values because you know them to be 0 for symmtry reasons. Therefore the class `WickOperatorTemplate` exists. Here, you specify, which kind of expectation values will be finite. In the following, the meaning of the different attributes is listed:

std::vector< IndexComparison > indexComparison}
If `any_identical` is `true`, any two identical indizes are considered valid. An example would be in the number operator $c^\dagger_{k,\sigma} c_{k,\sigma'}$*: No matter what σ is, as long as $sigma = sigma$' the expectation value will be finite. If `any_identical` is `false`, the members `base` and `other` become relevant: They define what the indizes need to be, e.g., for a pair annihlation operator $c_{-kdown} c_{kup}$ one would set `base` to $\downarrow$ and `other` to $\uparrow$.

Note, once one operator is set as a template, it is not necessary to set its Hermitian conjugate.

Momentum momentum_difference
Defines the allowed difference in momentum, e.g., for a number operator, this would be 0. Note, this also applies to a standard pair creation/annihilation operator, because in total, these operators create/annihilate a particle with $-k$ and one with k, resulting in 0 net momentum.

OperatorType type
Specifies what kind of WickOperator will be the result, see `enum OperatorType` in WickOperator.hpp.

bool is_sc_type
Specifies whether the operator is a pair creation/annihilation operator or a standard $c^\dagger c$ type term.

Apply Wick's theorem Create an instance of `WickTermCollector`. Then simply call

```
WickTermCollector wicks;
wicks_theorem(terms, templates, wicks);
```

```
clean_wicks(wicks);
```

Similar to how we worked with the Term class and commutators, it is strongly recommended to call clean_wicks() after applying Wick's theorem.

clean_wicks() will also make use of polymorphism to apply symmetries to the term, e.g., inversion symmetry. For details, see `WickSymmetry`.

You can print the the result to the console or utilize boost's serialization to load it later (or within another program). Serialization can be achieved via this code

```
std::ofstream ofs("path/to/file.txt");
boost::archive::text_oarchive oa(ofs);
oa « wicks;
ofs.close();
```

To later on load the output use

```
std::ifstream ifs("path/to/file.txt");
boost::archive::text_iarchive ia(ifs);
target.clear();
ia » target;
ifs.close();
```

or if you want to use this code of the iEoM, there is the class `TermLoader` for easy use. It loads the terms for the matrices M and N and saves same as class members.

See also

WickTermCollector, Coefficient, SumContainer, WickOperator, KroneckerDelta, Momentum, Index, clean_wicks(), wicks_theorem(), TermLoader

Definition at line 80 of file WickTerm.hpp.

7.24.2 Constructor & Destructor Documentation

7.24.2.1 WickTerm() [1/5]

```
mrock::symbolic_operators::WickTerm::WickTerm (
            const Term * base )  [explicit]
```

Constructs a WickTerm object from a base Term pointer.

Parameters

base	The base Term pointer.

Definition at line 14 of file WickTerm.cpp.

7.24.2.2 WickTerm() [2/5]

```
mrock::symbolic_operators::WickTerm::WickTerm (
            const Term & base )  [explicit]
```

Constructs a WickTerm object from a base Term reference.

Parameters

base	The base Term reference.

Definition at line 19 of file WickTerm.cpp.

7.24.2.3 WickTerm() [3/5]

```
mrock::symbolic_operators::WickTerm::WickTerm ( )  [default]
```

Default constructor for WickTerm.

7.24.2.4 WickTerm() [4/5]

```
mrock::symbolic_operators::WickTerm::WickTerm (
            const WickTerm & base,
            const TemplateResult::SingleResult & result )
```

Constructs a WickTerm object from a base WickTerm and a TemplateResult::SingleResult.

Parameters

base	The base WickTerm.
result	The TemplateResult::SingleResult.

Definition at line 24 of file WickTerm.cpp.

7.24.2.5 WickTerm() [5/5]

```
mrock::symbolic_operators::WickTerm::WickTerm (
            const std::string & expression )  [explicit]
```

Constructs a WickTerm object from a string expression.

Parameters

expression	The string expression.

Definition at line 31 of file WickTerm.cpp.

7.24.3 Member Function Documentation

7.24.3.1 compute_sums()

```
bool mrock::symbolic_operators::WickTerm::compute_sums ( )
```

Computes the sums in the term.

Returns

true if the sums were computed successfully.

false otherwise.

Definition at line 298 of file WickTerm.cpp.

7.24.3.2 discard_zero_momenta()

```
void mrock::symbolic_operators::WickTerm::discard_zero_momenta ( )
```

Discards zero momenta in the term.

Definition at line 401 of file WickTerm.cpp.

7.24.3.3 get_factor()

```
double mrock::symbolic_operators::WickTerm::get_factor ( ) const  [inline], [noexcept]
```

Returns the multiplicity as a double.

Returns

double The multiplicity as a double.

Definition at line 517 of file WickTerm.hpp.

7.24.3.4 get_first_coefficient()

```
const Coefficient & mrock::symbolic_operators::WickTerm::get_first_coefficient ( ) const  [inline]
```

Returns the first coefficient in the term.

Returns

const Coefficient& The first coefficient.

Definition at line 527 of file WickTerm.hpp.

7.24.3.5 handled()

```
bool mrock::symbolic_operators::WickTerm::handled ( ) const  [inline], [noexcept]
```

Checks if the term has been handled.

Returns

true if the term has been handled.

false otherwise.

Definition at line 531 of file WickTerm.hpp.

7.24.3.6 has_single_coefficient()

```
bool mrock::symbolic_operators::WickTerm::has_single_coefficient ( ) const  [inline], [noexcept]
```

Checks if the term has a single coefficient.

Returns

true if the term has a single coefficient.

false otherwise.

Definition at line 496 of file WickTerm.hpp.

7.24.3.7 include_template_result()

```
void mrock::symbolic_operators::WickTerm::include_template_result (
            const TemplateResult::SingleResult & result )
```

Includes a template result in the term.

Parameters

result	The TemplateResult::SingleResult to include.

Definition at line 587 of file WickTerm.cpp.

7.24.3.8 includes_type()

```
bool mrock::symbolic_operators::WickTerm::includes_type (
            const OperatorType operator_type ) const  [inline]
```

Checks if the term includes a specific operator type.

Parameters

operator_type	The operator type to check.

Returns

true if the term includes the operator type.

false otherwise.

Definition at line 492 of file WickTerm.hpp.

7.24.3.9 invert_momentum()

```
void mrock::symbolic_operators::WickTerm::invert_momentum (
            const MomentumSymbol::name_type what )
```

Inverts a momentum in the term.

Parameters

what	The momentum to invert.

Definition at line 593 of file WickTerm.cpp.

7.24.3.10 invert_momentum_sum()

```
void mrock::symbolic_operators::WickTerm::invert_momentum_sum (
            const MomentumSymbol::name_type what )
```

Inverts a momentum sum in the term.

Parameters

what	The momentum sum to invert.

Definition at line 602 of file WickTerm.cpp.

7.24.3.11 is_bilinear()

```
bool mrock::symbolic_operators::WickTerm::is_bilinear ( ) const  [inline], [noexcept]
```

Checks if the term is bilinear.

Returns

true if the term is bilinear.
false otherwise.

Definition at line 511 of file WickTerm.hpp.

7.24.3.12 is_identity()

```
bool mrock::symbolic_operators::WickTerm::is_identity ( ) const  [inline], [noexcept]
```

Checks if the term is an identity term.

Returns

true if the term is an identity term.
false otherwise.

Definition at line 508 of file WickTerm.hpp.

7.24.3.13 is_quartic()

```
bool mrock::symbolic_operators::WickTerm::is_quartic ( ) const  [inline], [noexcept]
```

Checks if the term is quartic.

Returns

true if the term is quartic.
false otherwise.

Definition at line 514 of file WickTerm.hpp.

7.24.3.14 remove_momentum_contribution()

```
void mrock::symbolic_operators::WickTerm::remove_momentum_contribution (
            const MomentumSymbol::name_type value )  [inline]
```

Removes a momentum contribution from the term.

Parameters

value	The momentum value to remove.

Definition at line 535 of file WickTerm.hpp.

7.24.3.15 rename_sums()

```
void mrock::symbolic_operators::WickTerm::rename_sums ( )
```

Renames the sums in the term.

Definition at line 411 of file WickTerm.cpp.

7.24.3.16 serialize()

```
template<class Archive >
void mrock::symbolic_operators::WickTerm::serialize (
            Archive & ar,
            const unsigned int version )  [inline]
```

Serializes the WickTerm object.

Template Parameters

Archive	The type of the archive.

Parameters

ar	The archive object.
version	The version of the serialization.

Definition at line 107 of file WickTerm.hpp.

7.24.3.17 set_deltas()

```
bool mrock::symbolic_operators::WickTerm::set_deltas ( )
```

Sets the deltas in the term.

Returns

true if the deltas were set successfully.

false otherwise.

Definition at line 117 of file WickTerm.cpp.

7.24.3.18 sort()

```
void mrock::symbolic_operators::WickTerm::sort ( )
```

Sorts the elements in the term.

Definition at line 484 of file WickTerm.cpp.

7.24.3.19 string_parser()

```
void mrock::symbolic_operators::WickTerm::string_parser (
            std::string && expression )  [private]
```

Parses a string expression to initialize the WickTerm.

Parameters

expression	The string expression.

Definition at line 59 of file WickTerm.cpp.

7.24.3.20 uses_index()

```
bool mrock::symbolic_operators::WickTerm::uses_index (
            const Index index ) const  [inline], [noexcept]
```

Checks if the term uses a specific index.

Parameters

index	The index to check.

Returns

true if the term uses the index.

false otherwise.

Definition at line 499 of file WickTerm.hpp.

7.24.3.21 which_operator_depends_on()

```
int mrock::symbolic_operators::WickTerm::which_operator_depends_on (
            const MomentumSymbol::name_type momentum ) const  [inline], [noexcept]
```

Returns the position of the first operator that depends on a specific momentum.

Parameters

momentum	The momentum to check.

Returns

int The position of the first operator that depends on the momentum, or -1 if none.

Definition at line 520 of file WickTerm.hpp.

7.24.4 Member Data Documentation

7.24.4.1 coefficients

```
std::vector<Coefficient> mrock::symbolic_operators::WickTerm::coefficients
```

The coefficients of the term.

Definition at line 91 of file WickTerm.hpp.

7.24.4.2 delta_indizes

```
std::vector<KroneckerDelta<Index> > mrock::symbolic_operators::WickTerm::delta_indizes
```

The index deltas.

Definition at line 97 of file WickTerm.hpp.

7.24.4.3 delta_momenta

```
std::vector<KroneckerDelta<Momentum> > mrock::symbolic_operators::WickTerm::delta_momenta
```

The momentum deltas.

Definition at line 96 of file WickTerm.hpp.

7.24.4.4 multiplicity

```
IntFractional mrock::symbolic_operators::WickTerm::multiplicity {}
```

The multiplicity of the term.

Definition at line 90 of file WickTerm.hpp.

7.24.4.5 operators

```
std::vector<WickOperator> mrock::symbolic_operators::WickTerm::operators
```

The operators in the term.

Definition at line 93 of file WickTerm.hpp.

7.24.4.6 sums

```
SumContainer mrock::symbolic_operators::WickTerm::sums
```

The sums in the term.

Definition at line 92 of file WickTerm.hpp.

7.24.4.7 temporary_operators

```
std::vector<Operator> mrock::symbolic_operators::WickTerm::temporary_operators
```

Temporary operators used in the term.

Definition at line 116 of file WickTerm.hpp.

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

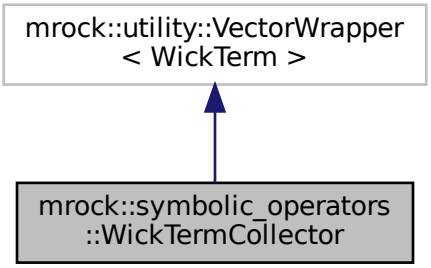
- include/mrock/symbolic_operators/WickTerm.hpp
- sources/WickTerm.cpp

7.25 mrock::symbolic_operators::WickTermCollector Class Reference

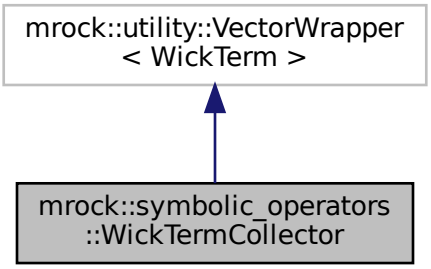
A wrapper for a vector of WickTerm objects.

`#include <WickTerm.hpp>`

Inheritance diagram for mrock::symbolic_operators::WickTermCollector:



Collaboration diagram for mrock::symbolic_operators::WickTermCollector:



Public Member Functions

- template<class Archive >
  void serialize (Archive &ar, const unsigned int version)

  Serializes the WickTermCollector object.

7.25.1 Detailed Description

A wrapper for a vector of WickTerm objects.

Definition at line 336 of file WickTerm.hpp.

7.25.2 Member Function Documentation

7.25.2.1 serialize()

```
template<class Archive >
void mrock::symbolic_operators::WickTermCollector::serialize (
            Archive & ar,
            const unsigned int version )  [inline]
```

Serializes the WickTermCollector object.

Template Parameters

Archive	The type of the archive.

Parameters

ar	The archive object.
version	The version of the serialization.

Definition at line 345 of file WickTerm.hpp.

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

- include/mrock/symbolic_operators/WickTerm.hpp

Chapter 8

File Documentation

8.1 include/mrock/symbolic_operators/Coefficient.hpp File Reference

Defines the Coefficient structure used in symbolic operators.

```
#include "Operator.hpp"
#include "IndexWrapper.hpp"
#include "MomentumList.hpp"
#include <optional>
#include <functional>
```

Classes

- struct mrock::symbolic_operators::Coefficient

  Represents a coefficient. Various symmetries are pre defined (e.g. inversion symmetry) and can be toggled on or off A custom symmetry can also be provided.

Namespaces

- mrock
- mrock::symbolic_operators

Functions

- bool mrock::symbolic_operators::operator== (const Coefficient &lhs, const Coefficient &rhs)

  Equality operator for Coefficient.
- bool mrock::symbolic_operators::operator!= (const Coefficient &lhs, const Coefficient &rhs)

  Inequality operator for Coefficient.

8.1.1 Detailed Description

Defines the Coefficient structure used in symbolic operators.

8.2 include/mrock/symbolic_operators/IndexWrapper.hpp File Reference

Defines the Index enum and the IndexWrapper class for handling indizes.

```
#include <iostream>
#include <mrock/utility/VectorWrapper.hpp>
#include <string>
#include <map>
```

Classes

- struct mrock::symbolic_operators::IndexWrapper

  A wrapper for a vector of Index values.

Namespaces

- mrock
- mrock::symbolic_operators

Typedefs

- typedef unsigned char mrock::symbolic_operators::index_base

  Defines the base type for the Index enum as unsigned char.

Enumerations

- enum class mrock::symbolic_operators::Index : index_base { mrock::symbolic_operators::SpinUp = 0 , mrock::symbolic_operators::SpinDown , mrock::symbolic_operators::Sigma , mrock::symbolic_operators::SigmaPrime , mrock::symbolic_operators::GeneralSpin_S , mrock::symbolic_operators::GeneralSpin_SPrime , mrock::symbolic_operators::Ty , mrock::symbolic_operators::TypeB , mrock::symbolic_operators::TypeC , mrock::symbolic_operators::char_a = 97 , mrock::symbolic_operators::UndefinedIndex = 254 , mrock::symbolic_operators::NoIndex = 255 }

  Enumeration representing various symbolic indices.

Functions

- constexpr Index mrock::symbolic_operators::char_to_index (unsigned char c)

  Converts a character to an Index.
- constexpr bool mrock::symbolic_operators::is_mutable (const Index idx)

  Checks if the given index represents a variable (mutable). 'Mutable' means that it is associated with a sum or similar. An example is sigma; it is commonly summed over as a representation of spins. Then expressions like delta_↩ {sigma,up} can be evaluated to be one if sigma=up. An Index like SpinUp is set to be non-mutable. This allows us to evaluate delta_{up,down}=0.
- std::ostream & mrock::symbolic_operators::operator<< (std::ostream &os, const Index index)

  Overloads the stream insertion operator for the Index enum.
- std::ostream & mrock::symbolic_operators::operator<< (std::ostream &os, const IndexWrapper &indizes)

  Overloads the stream insertion operator for the IndexWrapper struct.

Variables

- const std::map< std::string, Index > mrock::symbolic_operators::string_to_index

  A map that associates string representations with their corresponding Index values.

8.2.1 Detailed Description

Defines the Index enum and the IndexWrapper class for handling indizes.

8.3 include/mrock/symbolic_operators/KroneckerDelta.hpp File Reference

Defines the KroneckerDelta structure used in symbolic operators.

```
#include <utility>
#include <iostream>
#include <mrock/utility/defines_arithmetic_operators.hpp>
```

Classes

- class mrock::symbolic_operators::KroneckerDelta< T >

  A structure representing the Kronecker Delta.

Namespaces

- mrock
- mrock::symbolic_operators

Functions

- template<typename T >
  constexpr auto mrock::symbolic_operators::make_delta (const T &first, const T &second)

  Creates a KroneckerDelta object.
- template<typename T >
  constexpr auto mrock::symbolic_operators::make_delta (std::decay_t< T > &&first, std::decay_t< T > &&second)

  Creates a KroneckerDelta object with rvalue references.
- template<typename T >
  bool mrock::symbolic_operators::operator== (const KroneckerDelta< T > &lhs, const KroneckerDelta< T > &rhs)

  Equality operator for KroneckerDelta.
- template<typename T >
  bool mrock::symbolic_operators::operator!= (const KroneckerDelta< T > &lhs, const KroneckerDelta< T > &rhs)

  Inequality operator for KroneckerDelta.

- template<typename T >
  requires mrock::utility::defines_plus< T >::value KroneckerDelta< T > & mrock::symbolic_operators::operator+= (KroneckerDelta< T > &lhs, T &rhs)

  Addition assignment operator for KroneckerDelta.
- template<typename T >
  requires mrock::utility::defines_minus< T >::value KroneckerDelta< T > & mrock::symbolic_operators::operator-= (KroneckerDelta< T > &lhs, const T &rhs)

  Subtraction assignment operator for KroneckerDelta.
- template<typename T >
  requires mrock::utility::defines_plus< T >::value KroneckerDelta< T > mrock::symbolic_operators::operator+ (KroneckerDelta< T > lhs, T const &rhs)

  Addition operator for KroneckerDelta.
- template<typename T >
  requires mrock::utility::defines_minus< T >::value KroneckerDelta< T > mrock::symbolic_operators::operator- (KroneckerDelta< T > lhs, T const &rhs)

  Subtraction operator for KroneckerDelta.
- template<typename T >
  std::ostream & mrock::symbolic_operators::operator<< (std::ostream &os, const KroneckerDelta< T > &delta)

  Stream insertion operator for KroneckerDelta.

8.3.1 Detailed Description

Defines the KroneckerDelta structure used in symbolic operators.

8.4 include/mrock/symbolic_operators/KroneckerDeltaUtility.hpp File Reference

Utility functions for manipulating KroneckerDelta objects.

```
#include "KroneckerDelta.hpp"
#include "Momentum.hpp"
#include "IndexWrapper.hpp"
#include <mrock/utility/defines_arithmetic_operators.hpp>
```

Namespaces

- mrock
- mrock::symbolic_operators

Functions

- template<class T >
  void mrock::symbolic_operators::remove_delta_squared (std::vector< KroneckerDelta< T >> &deltas)

  Removes squared KroneckerDelta objects from the vector. Note that delta_{a,b}^N = delta_{a,b}.
- template<class T >
  void mrock::symbolic_operators::remove_delta_is_one (std::vector< KroneckerDelta< T >> &deltas)

  Removes KroneckerDelta objects that are one from the vector. Note that delta_{a,a} = 1.
- bool mrock::symbolic_operators::is_always_zero (const std::vector< KroneckerDelta< Index >> &deltas)

  Checks if the vector of KroneckerDelta<Index> objects is always zero.
- bool mrock::symbolic_operators::is_always_zero (const std::vector< KroneckerDelta< Momentum >> &deltas)

  Checks if the vector of KroneckerDelta<Momentum> objects is always zero.
- void mrock::symbolic_operators::remove_double_occurances (KroneckerDelta< Momentum > &delta)

  Removes double occurrences in a KroneckerDelta<Momentum> object.

8.4.1 Detailed Description

Utility functions for manipulating KroneckerDelta objects.

8.5 include/mrock/symbolic_operators/Momentum.hpp File Reference

Defines the Momentum structure and related operations for symbolic manipulation of momentum symbols.

```
#include <boost/archive/text_oarchive.hpp>
#include <boost/archive/text_iarchive.hpp>
#include <boost/serialization/vector.hpp>
#include <boost/serialization/utility.hpp>
#include <boost/serialization/string.hpp>
#include <algorithm>
#include <vector>
#include <utility>
#include <mrock/utility/VectorWrapper.hpp>
#include "MomentumSymbol.hpp"
```

Classes

- struct mrock::symbolic_operators::Momentum

  Represents a collection of momentum symbols with associated operations.

Namespaces

- mrock
- mrock::symbolic_operators

Typedefs

- typedef std::vector< MomentumSymbol > mrock::symbolic_operators::momentum_symbols

  Alias for a vector of MomentumSymbol.

Functions

- bool mrock::symbolic_operators::momentum_order (const Momentum &lhs, const Momentum &rhs)

  Compares two Momentum objects for ordering.
- Momentum mrock::symbolic_operators::operator+ (Momentum lhs, const Momentum &rhs)

  Adds two Momentum objects.
- Momentum mrock::symbolic_operators::operator- (Momentum lhs, const Momentum &rhs)

  Subtracts one Momentum from another.
- Momentum mrock::symbolic_operators::operator* (Momentum lhs, const int rhs)

  Multiplies a Momentum by an integer factor.
- Momentum mrock::symbolic_operators::operator* (const int lhs, Momentum rhs)

  Multiplies an integer factor by a Momentum.
- Momentum mrock::symbolic_operators::operator- (Momentum rhs)

  Negates a Momentum.
- bool mrock::symbolic_operators::operator> (const Momentum &lhs, const Momentum &rhs)

  Compares two Momentum objects for greater-than ordering.
- bool mrock::symbolic_operators::operator< (const Momentum &lhs, const Momentum &rhs)

  Compares two Momentum objects for less-than ordering.
- std::ostream & mrock::symbolic_operators::operator<< (std::ostream &os, const Momentum &momentum)

  Outputs a Momentum to an output stream.

8.5.1 Detailed Description

Defines the Momentum structure and related operations for symbolic manipulation of momentum symbols.

8.6 include/mrock/symbolic_operators/MomentumList.hpp File Reference

Defines the MomentumList class for handling a list of Momentum objects.

```
#include "Momentum.hpp"
#include <mrock/utility/VectorWrapper.hpp>
#include <algorithm>
```

Classes

- class mrock::symbolic_operators::MomentumList

  A wrapper class for a vector of Momentum objects with additional functionalities.

Namespaces

- mrock
- mrock::symbolic_operators

Functions

- std::ostream & mrock::symbolic_operators::operator<< (std::ostream &os, const MomentumList &momenta)

  Outputs the MomentumList to an output stream.

8.6.1 Detailed Description

Defines the MomentumList class for handling a list of Momentum objects.

8.7 include/mrock/symbolic_operators/MomentumSymbol.hpp File Reference

Defines the MomentumSymbol structure and related operators for symbolic operations.

```
#include <iostream>
#include <string>
```

Classes

- struct mrock::symbolic_operators::MomentumSymbol

  Represents a symbolic momentum with a factor and a name.
- struct mrock::symbolic_operators::MomentumSymbol::name_type

  Represents a name as a single character with comparison and serialization capabilities, but without arithmetic operations (it does not make sense to add or multiply names).

Namespaces

- mrock
- mrock::symbolic_operators

Functions

- std::ostream & mrock::symbolic_operators::operator<< (std::ostream &os, const MomentumSymbol↩ ::name_type name)

  Outputs the name_type to an output stream.
- std::istream & mrock::symbolic_operators::operator>> (std::istream &is, MomentumSymbol::name_type &name)

  Inputs a name_type from an input stream.
- std::string mrock::symbolic_operators::operator+ (const std::string &str, const MomentumSymbol::name_type sym)

  Concatenates a string and a name_type.
- std::string mrock::symbolic_operators::operator+ (const MomentumSymbol::name_type sym, const std↩ ::string &str)

  Concatenates a name_type and a string.

8.7.1 Detailed Description

Defines the MomentumSymbol structure and related operators for symbolic operations.

8.8 include/mrock/symbolic_operators/Operator.hpp File Reference

Defines the Operator struct and related functions for symbolic operators.

```
#include "Momentum.hpp"
#include "IndexWrapper.hpp"
```

Classes

- struct mrock::symbolic_operators::Operator

  Represents a symbolic operator with momentum, indices, and properties.

Namespaces

- mrock
- mrock::symbolic_operators

Functions

- bool mrock::symbolic_operators::operator== (const Operator &lhs, const Operator &rhs)

  Equality operator for Operator.
- bool mrock::symbolic_operators::operator!= (const Operator &lhs, const Operator &rhs)

  Inequality operator for Operator.
- std::ostream & mrock::symbolic_operators::operator<< (std::ostream &os, const Operator &op)

  Stream insertion operator for Operator.
- std::ostream & mrock::symbolic_operators::operator<< (std::ostream &os, const std::vector< Operator > &ops)

  Stream insertion operator for a vector of Operators.

8.8.1 Detailed Description

Defines the Operator struct and related functions for symbolic operators.

8.9 include/mrock/symbolic_operators/OperatorType.hpp File Reference

Defines the OperatorType enum and related functions for symbolic operators.

```
#include <iostream>
#include <map>
```

Namespaces

- mrock
- mrock::symbolic_operators

Enumerations

- enum mrock::symbolic_operators::OperatorType {
  mrock::symbolic_operators::Number_Type = 0 , mrock::symbolic_operators::CDW_Type , mrock::symbolic_operators::SC_Type
  , mrock::symbolic_operators::Eta_Type ,
  mrock::symbolic_operators::Undefined_Type = 255 }

Functions

- std::ostream & mrock::symbolic_operators::operator<< (std::ostream &os, const OperatorType op)

  Overloads the stream insertion operator for the Index enum.

Variables

- const std::map< std::string, OperatorType > mrock::symbolic_operators::string_to_wick

  A map that associates string representations with their corresponding Wick operators values.

8.9.1 Detailed Description

Defines the OperatorType enum and related functions for symbolic operators.

8.10 include/mrock/symbolic_operators/SumContainer.hpp File Reference

Defines the SumContainer structure and related operators for symbolic operations.

```
#include "SymbolicSum.hpp"
#include <mrock/utility/RangeUtility.hpp>
#include "IndexWrapper.hpp"
#include "MomentumSymbol.hpp"
```

Classes

- struct mrock::symbolic_operators::SumContainer

  A container for holding symbolic sums of momenta and spins.

Namespaces

- mrock
- mrock::symbolic_operators

Typedefs

- typedef SymbolicSum< Index > mrock::symbolic_operators::IndexSum

  Typedef for SymbolicSum with Index type.
- typedef SymbolicSum< MomentumSymbol::name_type > mrock::symbolic_operators::MomentumSum

  Typedef for SymbolicSum with MomentumSymbol::name_type type.

Functions

- bool mrock::symbolic_operators::operator== (const SumContainer &lhs, const SumContainer &rhs)

  Equality operator for SumContainer.
- bool mrock::symbolic_operators::operator!= (const SumContainer &lhs, const SumContainer &rhs)

  Inequality operator for SumContainer.
- std::ostream & mrock::symbolic_operators::operator<< (std::ostream &os, const SumContainer &sums)

  Stream insertion operator for SumContainer.

8.10.1 Detailed Description

Defines the SumContainer structure and related operators for symbolic operations.

8.11 include/mrock/symbolic_operators/SymbolicSum.hpp File Reference

Defines the SymbolicSum template struct for symbolic summation operations.

```
#include <mrock/utility/VectorWrapper.hpp>
#include <ostream>
```

Classes

- struct mrock::symbolic_operators::SymbolicSum< SumIndex >

  A struct representing a symbolic summation operation.

Namespaces

- mrock
- mrock::symbolic_operators

Functions

- template<class SumIndex >
  std::ostream & mrock::symbolic_operators::operator<< (std::ostream &os, SymbolicSum< SumIndex > const &sum)

  Outputs the SymbolicSum object to an output stream.

8.11.1 Detailed Description

Defines the SymbolicSum template struct for symbolic summation operations.

This file contains the definition of the SymbolicSum template struct, which is used to represent and manipulate symbolic summation operations. It provides various constructors, serialization support, and comparison operators.

8.12 include/mrock/symbolic_operators/Term.hpp File Reference

Defines the Term class and related functions for symbolic operators.

```
#include "KroneckerDelta.hpp"
#include "Coefficient.hpp"
#include "SumContainer.hpp"
#include <mrock/utility/Fractional.hpp>
#include <algorithm>
#include <vector>
```

Classes

- class mrock::symbolic_operators::Term

  Represents a term in symbolic operator expressions.

Namespaces

- mrock
- mrock::symbolic_operators

Macros

- #define _TERM_TRACKER_PARAMETER
- #define _TERM_TRACKER_ATTRIBUTE
- #define IF_IS_TERM_TRACKED(statement)
- #define CLEAR_TRACKED(terms)

Typedefs

- using mrock::symbolic_operators::IntFractional = mrock::utility::Fractional< int >

Functions

- std::vector< Term > mrock::symbolic_operators::commutator (const std::vector< Term > &left, const std::vector< Term > &right)

  Computes the commutator of two sets of terms: $[A, B] = AB - BA$.
- std::vector< Term > mrock::symbolic_operators::commutator (const Term &left, const std::vector< Term > &right)

  Computes the commutator of a term and a set of terms: $[A, B] = AB - BA$.
- std::vector< Term > mrock::symbolic_operators::commutator (const std::vector< Term > &left, const Term &right)

  Computes the commutator of a set of terms and a term: $[A, B] = AB - BA$.
- bool mrock::symbolic_operators::operator== (const Term &lhs, const Term &rhs)

  Checks if two terms are equal.
- bool mrock::symbolic_operators::operator!= (const Term &lhs, const Term &rhs)

  Checks if two terms are not equal.
- std::ostream & mrock::symbolic_operators::operator<< (std::ostream &os, const Coefficient &coeff)

  Outputs a coefficient to a stream.
- std::ostream & mrock::symbolic_operators::operator<< (std::ostream &os, const std::vector< Coefficient > &coeffs)

  Outputs a vector of coefficients to a stream.
- std::ostream & mrock::symbolic_operators::operator<< (std::ostream &os, const std::vector< Term > &terms)

  Outputs a vector of terms to a stream.
- void mrock::symbolic_operators::clear_duplicates (std::vector< Term > &terms)

  Clears duplicate terms from a vector.
- void mrock::symbolic_operators::clean_up (std::vector< Term > &terms)

  Sorts the terms, adds identical ones together and removes those that are equal to 0.
- void mrock::symbolic_operators::hermitian_conjugate (std::vector< Term > &terms)

  Applies the Hermitian conjugate to a vector of terms.
- void mrock::symbolic_operators::rename_momenta (std::vector< Term > &terms, const MomentumSymbol::name_type what, const MomentumSymbol::name_type to)

  Renames momenta in a vector of terms.
- std::string mrock::symbolic_operators::to_string_without_prefactor (const std::vector< Term > &terms)

  Converts a vector of terms to a string without the prefactor.

8.12.1 Detailed Description

Defines the Term class and related functions for symbolic operators.

8.12.2 Macro Definition Documentation

8.12.2.1 _TERM_TRACKER_ATTRIBUTE

```
#define _TERM_TRACKER_ATTRIBUTE
```

Definition at line 25 of file Term.hpp.

8.12.2.2 _TERM_TRACKER_PARAMETER

```
#define _TERM_TRACKER_PARAMETER
```

Definition at line 24 of file Term.hpp.

8.12.2.3 CLEAR_TRACKED

```
#define CLEAR_TRACKED(
              terms )
```

Definition at line 27 of file Term.hpp.

8.12.2.4 IF_IS_TERM_TRACKED

```
#define IF_IS_TERM_TRACKED(
              statement )
```

Definition at line 26 of file Term.hpp.

8.13 include/mrock/symbolic_operators/TermLoader.hpp File Reference

Header file for the TermLoader structure in the symbolic_operators namespace.

```
#include "WickTerm.hpp"
#include <vector>
#include <string>
```

Classes

- struct mrock::symbolic_operators::TermLoader

  A structure to load and manage Wick terms.

Namespaces

- mrock
- mrock::symbolic_operators

8.13.1 Detailed Description

Header file for the TermLoader structure in the symbolic_operators namespace.

8.14 include/mrock/symbolic_operators/Wick.hpp File Reference

Functions for applying Wick's theorem and manipulating Wick terms.

```
#include "WickTerm.hpp"
#include "WickSymmetry.hpp"
#include <vector>
#include <memory>
```

Namespaces

- mrock
- mrock::symbolic_operators

Functions

- WickTermCollector mrock::symbolic_operators::identify_wick_operators (const WickTerm &source, const std::vector< WickOperatorTemplate > &operator_templates)

  Identifies Wick operators in a given Wick term.
- void mrock::symbolic_operators::wicks_theorem (const std::vector< Term > &terms, const std::vector< WickOperatorTemplate > &operator_templates, WickTermCollector &reciever)

  Applies Wick's theorem to a set of terms.
- void mrock::symbolic_operators::clear_etas (WickTermCollector &terms)

  Clears eta terms from the WickTermCollector. Intended for use if <eta>=0.
- void mrock::symbolic_operators::clean_wicks (WickTermCollector &terms, const std::vector< std::unique_ptr< WickSymmetry >> &symmetries=std::vector< std::unique_ptr< WickSymmetry >>{})

  Cleans Wick terms using the provided symmetries.

8.14.1 Detailed Description

Functions for applying Wick's theorem and manipulating Wick terms.

8.15 include/mrock/symbolic_operators/WickOperator.hpp File Reference

Defines the WickOperator structure used in symbolic operators.

```
#include <iostream>
#include "Momentum.hpp"
#include "IndexWrapper.hpp"
#include "OperatorType.hpp"
```

Classes

- class mrock::symbolic_operators::WickOperator

  A structure representing a Wick operator.

Namespaces

- mrock
- mrock::symbolic_operators

Functions

- std::ostream & mrock::symbolic_operators::operator<< (std::ostream &os, const WickOperator &op)

  Stream insertion operator for WickOperator.
- std::ostream & mrock::symbolic_operators::operator<< (std::ostream &os, const std::vector< WickOperator > &ops)

  Stream insertion operator for a vector of WickOperator objects.

8.15.1 Detailed Description

Defines the WickOperator structure used in symbolic operators.

8.16 include/mrock/symbolic_operators/WickOperatorTemplate.hpp File Reference

Defines templates for creating Wick operators from a set of normal operators.

```
#include "Operator.hpp"
#include "WickOperator.hpp"
#include "KroneckerDelta.hpp"
#include <optional>
#include <algorithm>
```

Classes

- struct mrock::symbolic_operators::IndexComparison

  A structure for comparing indices. E.g. <n_k> merely requires that the spin indizes of the composing operators are identical, but <f_k> requires the first index to be spin down.
- struct mrock::symbolic_operators::TemplateResult

  A structure for storing the result of a template operation.
- struct mrock::symbolic_operators::TemplateResult::SingleResult

  A structure for storing a single result.
- class mrock::symbolic_operators::WickOperatorTemplate

  A template for creating Wick operators.

Namespaces

- mrock
- mrock::symbolic_operators

8.16.1 Detailed Description

Defines templates for creating Wick operators from a set of normal operators.

8.17 include/mrock/symbolic_operators/WickSymmetry.hpp File Reference

Defines symmetries for Wick terms.

```
#include "OperatorType.hpp"
#include "WickTerm.hpp"
#include <type_traits>
```

Classes

- class mrock::symbolic_operators::WickSymmetry

  An abstract base class for Wick symmetries.
- class mrock::symbolic_operators::SpinSymmetry

  A symmetry where expectation values for spin up and down are the same.
- class mrock::symbolic_operators::InversionSymmetry

  A symmetry where expectation values for k and -k are the same.
- class mrock::symbolic_operators::PhaseSymmetry< operators>

  A symmetry where <operator^+> = <operator>.

Namespaces

- mrock
- mrock::symbolic_operators

8.17.1 Detailed Description

Defines symmetries for Wick terms.

8.18 include/mrock/symbolic_operators/WickTerm.hpp File Reference

Defines the WickTerm structure and related functions.

```
#include "Term.hpp"
#include "WickOperator.hpp"
#include "WickOperatorTemplate.hpp"
#include <algorithm>
#include <mrock/utility/Fractional.hpp>
```

Classes

- class mrock::symbolic_operators::WickTerm

  A structure representing a Wick term.
- class mrock::symbolic_operators::WickTermCollector

  A wrapper for a vector of WickTerm objects.
- class mrock::symbolic_operators::bad_term_exception

  An exception class for bad terms.

Namespaces

- mrock
- mrock::symbolic_operators

Functions

- bool mrock::symbolic_operators::operator== (const WickOperator &lhs, const WickOperator &rhs)

  Equality operator for WickOperator.
- bool mrock::symbolic_operators::operator!= (const WickOperator &lhs, const WickOperator &rhs)

  Inequality operator for WickOperator.
- bool mrock::symbolic_operators::operator== (const WickTerm &lhs, const WickTerm &rhs)

  Equality operator for WickTerm.
- bool mrock::symbolic_operators::operator!= (const WickTerm &lhs, const WickTerm &rhs)

  Inequality operator for WickTerm.
- WickTermCollector & mrock::symbolic_operators::operator+= (WickTermCollector &lhs, const WickTerm &rhs)

  Addition assignment operator for WickTermCollector and WickTerm.
- WickTermCollector & mrock::symbolic_operators::operator-= (WickTermCollector &lhs, const WickTerm &rhs)

  Subtraction assignment operator for WickTermCollector and WickTerm.
- WickTermCollector & mrock::symbolic_operators::operator+= (WickTermCollector &lhs, const WickTerm↩ Collector &rhs)

  Addition assignment operator for two WickTermCollector objects.
- WickTermCollector & mrock::symbolic_operators::operator-= (WickTermCollector &lhs, const WickTerm↩ Collector &rhs)

  Subtraction assignment operator for two WickTermCollector objects.
- WickTermCollector mrock::symbolic_operators::operator+ (WickTermCollector lhs, const WickTerm &rhs)

  Addition operator for WickTermCollector and WickTerm.
- WickTermCollector mrock::symbolic_operators::operator- (WickTermCollector lhs, const WickTerm &rhs)

  Subtraction operator for WickTermCollector and WickTerm.
- WickTermCollector mrock::symbolic_operators::operator+ (const WickTerm &lhs, WickTermCollector rhs)

  Addition operator for WickTerm and WickTermCollector.
- WickTermCollector mrock::symbolic_operators::operator- (const WickTerm &lhs, WickTermCollector rhs)

  Subtraction operator for WickTerm and WickTermCollector.
- WickTermCollector mrock::symbolic_operators::operator+ (WickTermCollector lhs, const WickTermCollector &rhs)

  Addition operator for two WickTermCollector objects.
- WickTermCollector mrock::symbolic_operators::operator- (WickTermCollector lhs, const WickTermCollector &rhs)

  Subtraction operator for two WickTermCollector objects.
- std::ostream & mrock::symbolic_operators::operator<< (std::ostream &os, const WickTerm &term)

  Stream insertion operator for WickTerm.
- std::ostream & mrock::symbolic_operators::operator<< (std::ostream &os, const WickTermCollector &terms)

  Stream insertion operator for WickTermCollector.

8.18.1 Detailed Description

Defines the WickTerm structure and related functions.

8.19 mainpage.dox File Reference

8.20 sources/Coefficient.cpp File Reference

```
#include <mrock/symbolic_operators/Coefficient.hpp>
#include <mrock/utility/StringUtility.hpp>
```

Namespaces

- mrock
- mrock::symbolic_operators

Functions

- std::ostream & mrock::symbolic_operators::operator<< (std::ostream &os, const Coefficient &coeff)

  Outputs a coefficient to a stream.
- std::ostream & mrock::symbolic_operators::operator<< (std::ostream &os, const std::vector< Coefficient > &coeffs)

  Outputs a vector of coefficients to a stream.

8.21 sources/IndexWrapper.cpp File Reference

```
#include <mrock/symbolic_operators/IndexWrapper.hpp>
#include <cassert>
```

Namespaces

- mrock
- mrock::symbolic_operators

Functions

- std::ostream & mrock::symbolic_operators::operator<< (std::ostream &os, const Index index)

  Overloads the stream insertion operator for the Index enum.
- std::ostream & mrock::symbolic_operators::operator<< (std::ostream &os, const IndexWrapper &indizes)

  Overloads the stream insertion operator for the IndexWrapper struct.

8.22 sources/Momentum.cpp File Reference

```
#include <mrock/symbolic_operators/Momentum.hpp>
#include <cctype>
#include <sstream>
#include <string>
```

Namespaces

- mrock
- mrock::symbolic_operators

Functions

- momentum_symbols::value_type mrock::symbolic_operators::identify_subexpression (const std::string &sub)
- bool mrock::symbolic_operators::momentum_order (const Momentum &lhs, const Momentum &rhs)

  Compares two Momentum objects for ordering.
- std::ostream & mrock::symbolic_operators::operator<< (std::ostream &os, const Momentum &momentum)

  Outputs a Momentum to an output stream.
- bool mrock::symbolic_operators::operator> (const Momentum &lhs, const Momentum &rhs)

  Compares two Momentum objects for greater-than ordering.
- bool mrock::symbolic_operators::operator< (const Momentum &lhs, const Momentum &rhs)

  Compares two Momentum objects for less-than ordering.

8.23 sources/MomentumList.cpp File Reference

```
#include <mrock/symbolic_operators/MomentumList.hpp>
```

Namespaces

- mrock
- mrock::symbolic_operators

Functions

- std::ostream & mrock::symbolic_operators::operator<< (std::ostream &os, const MomentumList &momenta)

  Outputs the MomentumList to an output stream.

8.24 sources/Operator.cpp File Reference

```
#include <mrock/symbolic_operators/Operator.hpp>
```

Namespaces

- mrock
- mrock::symbolic_operators

Functions

- std::ostream & mrock::symbolic_operators::operator<< (std::ostream &os, const Operator &op)

  Stream insertion operator for Operator.
- std::ostream & mrock::symbolic_operators::operator<< (std::ostream &os, const std::vector< Operator > &ops)

  Stream insertion operator for a vector of Operators.

8.25 sources/OperatorType.cpp File Reference

```
#include <mrock/symbolic_operators/OperatorType.hpp>
```

Namespaces

- mrock
- mrock::symbolic_operators

Functions

- std::ostream & mrock::symbolic_operators::operator<< (std::ostream &os, const OperatorType op)

  Overloads the stream insertion operator for the Index enum.

8.26 sources/SumContainer.cpp File Reference

```
#include <mrock/symbolic_operators/SumContainer.hpp>
```

Namespaces

- mrock
- mrock::symbolic_operators

8.27 sources/Term.cpp File Reference

```
#include <mrock/symbolic_operators/Term.hpp>
#include <mrock/symbolic_operators/KroneckerDeltaUtility.hpp>
#include <mrock/utility/RangeUtility.hpp>
#include <sstream>
```

Namespaces

- mrock
- mrock::symbolic_operators

Macros

- #define fill_reciever(x) reciever[0].x = left.x; mrock::utility::append_vector(reciever[0].x, right.x); reciever[1].x = left.x; mrock::utility::append_vector(reciever[1].x, right.x);

Functions

- void mrock::symbolic_operators::normal_order (std::vector< Term > &terms)
- std::vector< Term > mrock::symbolic_operators::commutator (const Term &left, const Term &right)
- std::vector< Term > mrock::symbolic_operators::commutator (const std::vector< Term > &left, const std::vector< Term > &right)

  Computes the commutator of two sets of terms: $[A, B] = AB - BA$.
- std::ostream & mrock::symbolic_operators::operator<< (std::ostream &os, const Term &term)
- std::ostream & mrock::symbolic_operators::operator<< (std::ostream &os, const std::vector< Term > &terms)

  Outputs a vector of terms to a stream.
- void mrock::symbolic_operators::clean_up (std::vector< Term > &terms)

  Sorts the terms, adds identical ones together and removes those that are equal to 0.
- void mrock::symbolic_operators::clear_duplicates (std::vector< Term > &terms)

  Clears duplicate terms from a vector.
- std::string mrock::symbolic_operators::to_string_without_prefactor (const std::vector< Term > &terms)

  Converts a vector of terms to a string without the prefactor.

8.27.1 Macro Definition Documentation

8.27.1.1 fill_reciever

```
#define fill_reciever(
             x ) reciever[0].x = left.x; mrock::utility::append_vector(reciever[0].x, right.↩
x); reciever[1].x = left.x; mrock::utility::append_vector(reciever[1].x, right.x);
```

Definition at line 693 of file Term.cpp.

8.28 sources/TermLoader.cpp File Reference

```
#include <mrock/symbolic_operators/TermLoader.hpp>
#include <fstream>
#include <boost/archive/binary_iarchive.hpp>
#include <filesystem>
```

Namespaces

- mrock
- mrock::symbolic_operators

8.29 sources/Wick.cpp File Reference

```
#include <mrock/symbolic_operators/Wick.hpp>
#include <mrock/symbolic_operators/KroneckerDeltaUtility.hpp>
#include <mrock/utility/Numerics/MathFunctions.hpp>
#include <mrock/utility/RangeUtility.hpp>
#include <variant>
#include <numeric>
```

Namespaces

- mrock
- mrock::symbolic_operators

Functions

- void mrock::symbolic_operators::wick_processor (const std::vector< Operator > &remaining, WickTerm↩ Collector &reciever_list, std::variant< WickTerm, Term > buffer)
- WickTermCollector mrock::symbolic_operators::prepare_wick (const std::vector< Term > &terms)
- WickTermCollector mrock::symbolic_operators::identify_wick_operators (const WickTerm &source, const std::vector< WickOperatorTemplate > &operator_templates)

  Identifies Wick operators in a given Wick term.
- void mrock::symbolic_operators::wicks_theorem (const std::vector< Term > &terms, const std::vector< WickOperatorTemplate > &operator_templates, WickTermCollector &reciever)

  Applies Wick's theorem to a set of terms.
- void mrock::symbolic_operators::clear_etas (WickTermCollector &terms)

  Clears eta terms from the WickTermCollector. Intended for use if <eta>=0.
- void mrock::symbolic_operators::clean_wicks (WickTermCollector &terms, const std::vector< std::unique_↩ ptr< WickSymmetry >> &symmetries=std::vector< std::unique_ptr< WickSymmetry >>{})

  Cleans Wick terms using the provided symmetries.

8.30 sources/WickOperator.cpp File Reference

```
#include <mrock/symbolic_operators/WickOperator.hpp>
#include <mrock/utility/StringUtility.hpp>
#include <cassert>
```

Namespaces

- mrock
- mrock::symbolic_operators

Functions

- std::ostream & mrock::symbolic_operators::operator<< (std::ostream &os, const WickOperator &op)

  Stream insertion operator for WickOperator.
- std::ostream & mrock::symbolic_operators::operator<< (std::ostream &os, const std::vector< WickOperator > &ops)

  Stream insertion operator for a vector of WickOperator objects.

8.31 sources/WickOperatorTemplate.cpp File Reference

```
#include <mrock/symbolic_operators/WickOperatorTemplate.hpp>
#include <mrock/symbolic_operators/Momentum.hpp>
#include <mrock/symbolic_operators/KroneckerDelta.hpp>
#include <mrock/symbolic_operators/KroneckerDeltaUtility.hpp>
#include <algorithm>
#include <iterator>
```

Namespaces

- mrock
- mrock::symbolic_operators

8.32 sources/WickSymmetry.cpp File Reference

```
#include <mrock/symbolic_operators/WickSymmetry.hpp>
#include <mrock/symbolic_operators/WickTerm.hpp>
```

Namespaces

- mrock
- mrock::symbolic_operators

8.33 sources/WickTerm.cpp File Reference

```
#include <mrock/symbolic_operators/WickTerm.hpp>
#include <mrock/symbolic_operators/KroneckerDeltaUtility.hpp>
#include <mrock/utility/StringUtility.hpp>
#include <cctype>
#include <cassert>
```

Namespaces

- mrock
- mrock::symbolic_operators

Macros

- #define LEFT temporary_operators[i]
- #define RIGHT temporary_operators[i + 1]
- #define L_SPIN temporary_operators[i].first_index()
- #define R_SPIN temporary_operators[i + 1].first_index()

Functions

- std::ostream & mrock::symbolic_operators::operator<< (std::ostream &os, const WickTerm &term)

  Stream insertion operator for WickTerm.
- std::ostream & mrock::symbolic_operators::operator<< (std::ostream &os, const WickTermCollector &terms)

  Stream insertion operator for WickTermCollector.
- WickTermCollector & mrock::symbolic_operators::operator+= (WickTermCollector &lhs, const WickTerm &rhs)

  Addition assignment operator for WickTermCollector and WickTerm.
- WickTermCollector & mrock::symbolic_operators::operator-= (WickTermCollector &lhs, const WickTerm &rhs)

  Subtraction assignment operator for WickTermCollector and WickTerm.
- WickTermCollector & mrock::symbolic_operators::operator+= (WickTermCollector &lhs, const WickTerm↩ Collector &rhs)

  Addition assignment operator for two WickTermCollector objects.
- WickTermCollector & mrock::symbolic_operators::operator-= (WickTermCollector &lhs, const WickTerm↩ Collector &rhs)

  Subtraction assignment operator for two WickTermCollector objects.

8.33.1 Macro Definition Documentation

8.33.1.1 L_SPIN

```
#define L_SPIN temporary_operators[i].first_index()
```

Definition at line 9 of file WickTerm.cpp.

8.33.1.2 LEFT

```
#define LEFT temporary_operators[i]
```

Definition at line 7 of file WickTerm.cpp.

8.33.1.3 R_SPIN

```
#define R_SPIN temporary_operators[i + 1].first_index()
```

Definition at line 10 of file WickTerm.cpp.

8.33.1.4 RIGHT

```
#define RIGHT temporary_operators[i + 1]
```

Definition at line 8 of file WickTerm.cpp.

8.34 tests/bosons.cpp File Reference

Example code for defining and using bosonic operators.

```
#include "compare_test.hpp"
#include <string>
#include <iostream>
#include <filesystem>
```

Functions

- int main (int argc, char **argv)

Variables

- const std::string begin_align = "\\begin{align*}\n\t"
- const std::string end_align = "\\end{align*}\n"
- const std::string file_names [2] = { "first_commutation.bin", "second_commutation.bin" }
- const std::string COMPARE_DIR = "../../../symbolic_operators/tests/correct_bosons/"

8.34.1 Detailed Description

Example code for defining and using bosonic operators.

This file demonstrates how to define a Hamiltonian using bosonic operators and how to perform commutation operations on them.

8.34.2 Function Documentation

8.34.2.1 main()

```
int main (
             int argc,
             char ** argv )
```

Definition at line 21 of file bosons.cpp.

8.34.3 Variable Documentation

8.34.3.1 begin_align

```
const std::string begin_align = "\\begin{align*}\n\t"
```

Definition at line 16 of file bosons.cpp.

8.34.3.2 COMPARE_DIR

```
const std::string COMPARE_DIR = "../../../symbolic_operators/tests/correct_bosons/"
```

Definition at line 19 of file bosons.cpp.

8.34.3.3 end_align

```
const std::string end_align = "\\end{align*}\n"
```

Definition at line 17 of file bosons.cpp.

8.34.3.4 file_names

```
const std::string file_names[2] = { "first_commutation.bin", "second_commutation.bin" }
```

Definition at line 18 of file bosons.cpp.

8.35 tests/compare_test.hpp File Reference

```
#include "../include/mrock/symbolic_operators/Term.hpp"
#include "../include/mrock/symbolic_operators/Wick.hpp"
#include <boost/archive/binary_oarchive.hpp>
#include <boost/archive/binary_iarchive.hpp>
#include <fstream>
#include <filesystem>
#include <string>
```

Classes

- struct sym_op_test::SymOpTest

Namespaces

- sym_op_test

8.36 tests/continuum.cpp File Reference

Example code for defining and using continuum operators.

```
#include "compare_test.hpp"
#include <string>
#include <iostream>
#include <filesystem>
```

Functions

- int main (int argc, char **argv)

8.36.1 Detailed Description

Example code for defining and using continuum operators.

This file demonstrates how to define a Hamiltonian using an interacting electron gas. The commutations are performed, Wick's theorem applied and then the terms are reduced using symmetries.

8.36.2 Function Documentation

8.36.2.1 main()

```
int main (
            int argc,
            char ** argv )
```

Definition at line 16 of file continuum.cpp.

Index

_TERM_TRACKER_ATTRIBUTE
    mrock::symbolic_operators::Term, 157
    Term.hpp, 196
_TERM_TRACKER_PARAMETER
    Term.hpp, 196
_handle_num_type
    mrock::symbolic_operators::WickOperatorTemplate, 166
_handle_sc_type
    mrock::symbolic_operators::WickOperatorTemplate, 167
_n
    mrock::symbolic_operators::MomentumSymbol::name_type, 108
_parent
    mrock::symbolic_operators::MomentumList, 96
_term
    mrock::symbolic_operators::bad_term_exception, 61
~WickSymmetry
    mrock::symbolic_operators::WickSymmetry, 170

add_in_place
    mrock::symbolic_operators::Momentum, 86
add_index_delta
    mrock::symbolic_operators::TemplateResult, 137
add_index_delta_range
    mrock::symbolic_operators::TemplateResult, 137
add_momentum
    mrock::symbolic_operators::Operator, 112, 113
add_Q
    mrock::symbolic_operators::Momentum, 94
any_identical
    mrock::symbolic_operators::IndexComparison, 75
append
    mrock::symbolic_operators::SumContainer, 125
apply_custom_symmetry
    mrock::symbolic_operators::Coefficient, 66
apply_to
    mrock::symbolic_operators::InversionSymmetry, 80
    mrock::symbolic_operators::PhaseSymmetry< operators >, 119
    mrock::symbolic_operators::SpinSymmetry, 123
    mrock::symbolic_operators::WickSymmetry, 170

bad_term_exception
    mrock::symbolic_operators::bad_term_exception, 60
base
    mrock::symbolic_operators::IndexComparison, 75
begin_align
    bosons.cpp, 210
Boson
    mrock::symbolic_operators::Operator, 113
bosons.cpp
    begin_align, 210
    COMPARE_DIR, 210
    end_align, 210
    file_names, 210
    main, 209

CDW_Type
    mrock::symbolic_operators, 21
char_a
    mrock::symbolic_operators, 20
char_to_index
    mrock::symbolic_operators, 21
clean_up
    mrock::symbolic_operators, 21
    mrock::symbolic_operators::TemplateResult, 137
clean_wicks
    mrock::symbolic_operators, 22
clear_delta_equals_one
    mrock::symbolic_operators::TemplateResult::SingleResult, 120
clear_duplicates
    mrock::symbolic_operators, 22
clear_etas
    mrock::symbolic_operators, 22
clear_impossible
    mrock::symbolic_operators::TemplateResult, 137
CLEAR_TRACKED
    Term.hpp, 197
Coefficient
    mrock::symbolic_operators::Coefficient, 64, 65
coefficients
    mrock::symbolic_operators::Term, 157
    mrock::symbolic_operators::WickTerm, 181
commutator
    mrock::symbolic_operators, 23, 24
    mrock::symbolic_operators::Term, 156
COMPARE_DIR
    bosons.cpp, 210
    sym_op_test::SymOpTest, 134
compute_sums
    mrock::symbolic_operators::Term, 148
    mrock::symbolic_operators::WickTerm, 175
Constant
    mrock::symbolic_operators::Coefficient, 66

contains_boson
mrock::symbolic_operators::Term, 148
contains_fermion
mrock::symbolic_operators::Term, 148
contains_impossible_delta
mrock::symbolic_operators::TemplateResult::SingleResult, 121
continuum.cpp
main, 211
count_bosons
mrock::symbolic_operators::Term, 149
count_fermions
mrock::symbolic_operators::Term, 149
create_branch
mrock::symbolic_operators::TemplateResult, 138
create_from_operators
mrock::symbolic_operators::WickOperatorTemplate, 167
custom_symmetry
mrock::symbolic_operators::Coefficient, 72

delta_indizes
mrock::symbolic_operators::Term, 157
mrock::symbolic_operators::WickTerm, 181
delta_momenta
mrock::symbolic_operators::Term, 158
mrock::symbolic_operators::WickTerm, 181
depends_on
mrock::symbolic_operators::Coefficient, 67
mrock::symbolic_operators::WickOperator, 163
depends_on_momentum
mrock::symbolic_operators::Coefficient, 67
depends_on_two_momenta
mrock::symbolic_operators::Coefficient, 67
differs_only_in_Q
mrock::symbolic_operators::Momentum, 87
discard_zero_momenta
mrock::symbolic_operators::Term, 149
mrock::symbolic_operators::WickTerm, 175

end_align
bosons.cpp, 210
Eta_Type
mrock::symbolic_operators, 21

factor
mrock::symbolic_operators::MomentumSymbol, 104
mrock::symbolic_operators::TemplateResult::SingleResult, 121
file_names
bosons.cpp, 210
fill_reciever
Term.cpp, 205
first
mrock::symbolic_operators::KroneckerDelta< T >, 82
first_index
mrock::symbolic_operators::Operator, 114
first_momentum_is
mrock::symbolic_operators::Momentum, 87
first_momentum_is_negative
mrock::symbolic_operators::Momentum, 87
flip_momentum
mrock::symbolic_operators::Momentum, 88
mrock::symbolic_operators::MomentumList, 98
flip_sign
mrock::symbolic_operators::Term, 149
flip_single
mrock::symbolic_operators::Momentum, 88
mrock::symbolic_operators::MomentumList, 98

GeneralSpin_S
mrock::symbolic_operators, 20
GeneralSpin_SPrime
mrock::symbolic_operators, 20
get_factor
mrock::symbolic_operators::WickTerm, 175
get_first_coefficient
mrock::symbolic_operators::WickTerm, 175
get_operators
mrock::symbolic_operators::Term, 150

handled
mrock::symbolic_operators::WickTerm, 176
has_momentum
mrock::symbolic_operators::SumContainer, 126
has_single_coefficient
mrock::symbolic_operators::WickTerm, 176
has_spins
mrock::symbolic_operators::SumContainer, 126
hermitian_conjugate
mrock::symbolic_operators, 24
mrock::symbolic_operators::Coefficient, 67
mrock::symbolic_operators::Operator, 114
mrock::symbolic_operators::Term, 150
hermitian_conjugate_inplace
mrock::symbolic_operators::Coefficient, 68
mrock::symbolic_operators::Operator, 114
mrock::symbolic_operators::Term, 150
HoneyComb
mrock::symbolic_operators::Coefficient, 68

identify_subexpression
mrock::symbolic_operators, 25
identify_wick_operators
mrock::symbolic_operators, 25
IS_TERM_TRACKED
Term.hpp, 197
include/mrock/symbolic_operators/Coefficient.hpp, 185
include/mrock/symbolic_operators/IndexWrapper.hpp, 186
include/mrock/symbolic_operators/KroneckerDelta.hpp, 187
include/mrock/symbolic_operators/KroneckerDeltaUtility.hpp, 188
include/mrock/symbolic_operators/Momentum.hpp, 189

include/mrock/symbolic_operators/MomentumList.hpp, 190
include/mrock/symbolic_operators/MomentumSymbol.hpp, 191
include/mrock/symbolic_operators/Operator.hpp, 192
include/mrock/symbolic_operators/OperatorType.hpp, 192
include/mrock/symbolic_operators/SumContainer.hpp, 193
include/mrock/symbolic_operators/SymbolicSum.hpp, 194
include/mrock/symbolic_operators/Term.hpp, 195
include/mrock/symbolic_operators/TermLoader.hpp, 197
include/mrock/symbolic_operators/Wick.hpp, 198
include/mrock/symbolic_operators/WickOperator.hpp, 198
include/mrock/symbolic_operators/WickOperatorTemplate.hpp, 199
include/mrock/symbolic_operators/WickSymmetry.hpp, 200
include/mrock/symbolic_operators/WickTerm.hpp, 200
include_template_result
mrock::symbolic_operators::WickTerm, 176
includes_type
mrock::symbolic_operators::WickTerm, 177
Index
mrock::symbolic_operators, 19
index_base
mrock::symbolic_operators, 18
index_deltas
mrock::symbolic_operators::TemplateResult::SingleResult, 121
indexComparison
mrock::symbolic_operators::WickOperatorTemplate, 167
IndexSum
mrock::symbolic_operators, 19
IndexWrapper
mrock::symbolic_operators::IndexWrapper, 76, 77
indizes
mrock::symbolic_operators::Coefficient, 72
mrock::symbolic_operators::IndexWrapper, 78
mrock::symbolic_operators::Operator, 116
mrock::symbolic_operators::WickOperator, 164
IntFractional
mrock::symbolic_operators, 19
inversion_symmetry
mrock::symbolic_operators::Coefficient, 73
invert_momentum
mrock::symbolic_operators::Coefficient, 69
mrock::symbolic_operators::Term, 150
mrock::symbolic_operators::WickTerm, 177
invert_momentum_sum
mrock::symbolic_operators::Term, 151
mrock::symbolic_operators::WickTerm, 177
is_always_zero
mrock::symbolic_operators, 25, 26
is_bilinear
mrock::symbolic_operators::WickTerm, 178
is_daggered
mrock::symbolic_operators::Coefficient, 73
mrock::symbolic_operators::Operator, 116
mrock::symbolic_operators::WickOperator, 164
is_equal
mrock::symbolic_operators::Term, 151
is_fermion
mrock::symbolic_operators::Operator, 117
is_identity
mrock::symbolic_operators::Term, 151
mrock::symbolic_operators::WickTerm, 178
is_mutable
mrock::symbolic_operators, 26
is_normal_ordered
mrock::symbolic_operators::Term, 152
is_quartic
mrock::symbolic_operators::WickTerm, 178
is_real
mrock::symbolic_operators::Coefficient, 73
is_sc_type
mrock::symbolic_operators::WickOperatorTemplate, 168
is_summed_over
mrock::symbolic_operators::SymbolicSum< SumIndex >, 130
is_symmetrized_interaction
mrock::symbolic_operators::Coefficient, 73
is_used_at
mrock::symbolic_operators::Momentum, 88
is_zero
mrock::symbolic_operators::Momentum, 89
isOne
mrock::symbolic_operators::KroneckerDelta< T >, 81

L_SPIN
WickTerm.cpp, 208
last_momentum_is
mrock::symbolic_operators::Momentum, 89
last_momentum_is_negative
mrock::symbolic_operators::Momentum, 89
LEFT
WickTerm.cpp, 208
load
mrock::symbolic_operators::TermLoader, 159
load_and_test
sym_op_test::SymOpTest, 133

M
mrock::symbolic_operators::TermLoader, 160
main
bosons.cpp, 209
continuum.cpp, 211
mainpage.dox, 202
make_delta
mrock::symbolic_operators, 26, 27
momenta

mrock::symbolic_operators::Coefficient, 73
mrock::symbolic_operators::SumContainer, 127
Momentum
mrock::symbolic_operators::Momentum, 84–86
momentum
mrock::symbolic_operators::Operator, 117
mrock::symbolic_operators::WickOperator, 165
momentum_delta
mrock::symbolic_operators::TemplateResult, 139
momentum_difference
mrock::symbolic_operators::WickOperatorTemplate, 168
momentum_list
mrock::symbolic_operators::Momentum, 94
momentum_order
mrock::symbolic_operators, 27
momentum_symbols
mrock::symbolic_operators, 19
MomentumList
mrock::symbolic_operators::MomentumList, 96–98
MomentumSum
mrock::symbolic_operators, 19
MomentumSymbol
mrock::symbolic_operators::MomentumSymbol, 103
mrock, 13
mrock::symbolic_operators, 13
CDW_Type, 21
char_a, 20
char_to_index, 21
clean_up, 21
clean_wicks, 22
clear_duplicates, 22
clear_etas, 22
commutator, 23, 24
Eta_Type, 21
GeneralSpin_S, 20
GeneralSpin_SPrime, 20
hermitian_conjugate, 24
identify_subexpression, 25
identify_wick_operators, 25
Index, 19
index_base, 18
IndexSum, 19
IntFractional, 19
is_always_zero, 25, 26
is_mutable, 26
make_delta, 26, 27
momentum_order, 27
momentum_symbols, 19
MomentumSum, 19
NoIndex, 20
normal_order, 28
Number_Type, 21
operator!=, 28–31
operator<, 41
operator<<, 41–49
operator>, 52
operator>>, 53
operator*, 31, 32
operator+, 32–35
operator+=, 35, 36
operator-, 37–39
operator-=, 39, 40
operator==, 49–52
OperatorType, 21
prepare_wick, 53
remove_delta_is_one, 53
remove_delta_squared, 54
remove_double_occurances, 54
rename_momenta, 55
SC_Type, 21
Sigma, 20
SigmaPrime, 20
SpinDown, 20
SpinUp, 20
string_to_index, 56
string_to_wick, 56
to_string_without_prefactor, 55
TypeA, 20
TypeB, 20
TypeC, 20
Undefined_Type, 21
UndefinedIndex, 20
wick_processor, 55
wicks_theorem, 56
mrock::symbolic_operators::bad_term_exception, 59
_term, 61
bad_term_exception, 60
which_term, 61
mrock::symbolic_operators::Coefficient, 62
apply_custom_symmetry, 66
Coefficient, 64, 65
Constant, 66
custom_symmetry, 72
depends_on, 67
depends_on_momentum, 67
depends_on_two_momenta, 67
hermitian_conjugate, 67
hermitian_conjugate_inplace, 68
HoneyComb, 68
indizes, 72
inversion_symmetry, 73
invert_momentum, 69
is_daggered, 73
is_real, 73
is_symmetrized_interaction, 73
momenta, 73
name, 74
parse_interaction_string, 69
parse_string, 69
Q_changes_sign, 74
RealInteraction, 70
RealInversionSymmetric, 70
remove_momentum_contribution, 71
serialize, 71

use_symmetric_interaction_exchange, 72
use_symmetric_interaction_inversion, 72
uses_index, 72
mrock::symbolic_operators::IndexComparison, 74
any_identical, 75
base, 75
other, 75
mrock::symbolic_operators::IndexWrapper, 75
IndexWrapper, 76, 77
indizes, 78
operator<=>, 77
serialize, 78
VECTOR_WRAPPER_FILL_MEMBERS, 78
mrock::symbolic_operators::InversionSymmetry, 79
apply_to, 80
mrock::symbolic_operators::KroneckerDelta< T >, 80
first, 82
isOne, 81
second, 82
serialize, 81
mrock::symbolic_operators::Momentum, 82
add_in_place, 86
add_Q, 94
differs_only_in_Q, 87
first_momentum_is, 87
first_momentum_is_negative, 87
flip_momentum, 88
flip_single, 88
is_used_at, 88
is_zero, 89
last_momentum_is, 89
last_momentum_is_negative, 89
Momentum, 84–86
momentum_list, 94
multiply_by, 89
operator!=, 90
operator*=, 90
operator+=, 91
operator-=, 91
operator==, 91
remove_contribution, 92
remove_zeros, 92
replace_occurances, 92
serialize, 93
sort, 93
to_string, 93
uses, 93
VECTOR_WRAPPER_FILL_MEMBERS, 94
mrock::symbolic_operators::MomentumList, 95
_parent, 96
flip_momentum, 98
flip_single, 98
MomentumList, 96–98
multiply_by, 99
operator*=, 99
remove_zeros, 99
replace_occurances, 99
serialize, 101
sort, 101
mrock::symbolic_operators::MomentumSymbol, 102
factor, 104
MomentumSymbol, 103
name, 105
operator<=>, 104
serialize, 104
mrock::symbolic_operators::MomentumSymbol::name_type, 105
_n, 108
name_type, 106
operator char, 106
operator<=>, 106, 107
serialize, 107
mrock::symbolic_operators::Operator, 108
add_momentum, 112, 113
Boson, 113
first_index, 114
hermitian_conjugate, 114
hermitian_conjugate_inplace, 114
indizes, 116
is_daggered, 116
is_fermion, 117
momentum, 117
Operator, 110–112
remove_momentum_contribution, 114
serialize, 115
set_first_index, 115
with_momentum, 115, 116
mrock::symbolic_operators::PhaseSymmetry< operators>, 117
apply_to, 119
mrock::symbolic_operators::SpinSymmetry, 122
apply_to, 123
mrock::symbolic_operators::SumContainer, 123
append, 125
has_momentum, 126
has_spins, 126
momenta, 127
push_back, 126, 127
serialize, 127
spins, 128
mrock::symbolic_operators::SymbolicSum< SumIndex >, 128
is_summed_over, 130
operator<=>, 131
serialize, 131
summations, 132
SymbolicSum, 129, 130
VECTOR_WRAPPER_FILL_MEMBERS, 132
mrock::symbolic_operators::TemplateResult, 135
add_index_delta, 137
add_index_delta_range, 137
clean_up, 137
clear_impossible, 137
create_branch, 138
momentum_delta, 139
null_result, 138

operation_on_each, 138
operation_on_range, 139
operator bool, 139
results, 140
TemplateResult, 136
mrock::symbolic_operators::TemplateResult::SingleResult, 119
clear_delta_equals_one, 120
contains_impossible_delta, 121
factor, 121
index_deltas, 121
op, 121
mrock::symbolic_operators::Term, 140
_TERM_TRACKER_ATTRIBUTE, 157
coefficients, 157
commutator, 156
compute_sums, 148
contains_boson, 148
contains_fermion, 148
count_bosons, 149
count_fermions, 149
delta_indizes, 157
delta_momenta, 158
discard_zero_momenta, 149
flip_sign, 149
get_operators, 150
hermitian_conjugate, 150
hermitian_conjugate_inplace, 150
invert_momentum, 150
invert_momentum_sum, 151
is_equal, 151
is_identity, 151
is_normal_ordered, 152
multiplicity, 158
normal_order, 156
operator<<, 156
operators, 158
perform_operator_swap, 152
print, 152
remove_momentum_contribution, 152
rename_indizes, 153
rename_momenta, 153
rename_sums, 153
serialize, 154
set_deltas, 154
sort, 154
sums, 158
swap_momenta, 154
Term, 143–145, 147, 148
to_string_without_prefactor, 155
transform_momentum_sum, 155
WickTerm, 157
mrock::symbolic_operators::TermLoader, 159
load, 159
M, 160
N, 160
mrock::symbolic_operators::WickOperator, 160
depends_on, 163
indizes, 164
is_daggered, 164
momentum, 165
remove_momentum_contribution, 163
serialize, 163
type, 165
uses_index, 164
WickOperator, 161, 162
mrock::symbolic_operators::WickOperatorTemplate, 165
_handle_num_type, 166
_handle_sc_type, 167
create_from_operators, 167
indexComparison, 167
is_sc_type, 168
momentum_difference, 168
type, 168
mrock::symbolic_operators::WickSymmetry, 169
~WickSymmetry, 170
apply_to, 170
mrock::symbolic_operators::WickTerm, 170
coefficients, 181
compute_sums, 175
delta_indizes, 181
delta_momenta, 181
discard_zero_momenta, 175
get_factor, 175
get_first_coefficient, 175
handled, 176
has_single_coefficient, 176
include_template_result, 176
includes_type, 177
invert_momentum, 177
invert_momentum_sum, 177
is_bilinear, 178
is_identity, 178
is_quartic, 178
multiplicity, 182
operators, 182
remove_momentum_contribution, 178
rename_sums, 179
serialize, 179
set_deltas, 179
sort, 180
string_parser, 180
sums, 182
temporary_operators, 182
uses_index, 180
which_operator_depends_on, 181
WickTerm, 173, 174
mrock::symbolic_operators::WickTermCollector, 183
serialize, 184
multiplicity
mrock::symbolic_operators::Term, 158
mrock::symbolic_operators::WickTerm, 182
multiply_by
mrock::symbolic_operators::Momentum, 89
mrock::symbolic_operators::MomentumList, 99

N
mrock::symbolic_operators::TermLoader, 160
name
mrock::symbolic_operators::Coefficient, 74
mrock::symbolic_operators::MomentumSymbol, 105
name_type
mrock::symbolic_operators::MomentumSymbol::name_type, 106
NoIndex
mrock::symbolic_operators, 20
normal_order
mrock::symbolic_operators, 28
mrock::symbolic_operators::Term, 156
null_result
mrock::symbolic_operators::TemplateResult, 138
Number_Type
mrock::symbolic_operators, 21

op
mrock::symbolic_operators::TemplateResult::SingleResult, 121
operation_on_each
mrock::symbolic_operators::TemplateResult, 138
operation_on_range
mrock::symbolic_operators::TemplateResult, 139
Operator
mrock::symbolic_operators::Operator, 110–112
operator bool
mrock::symbolic_operators::TemplateResult, 139
operator char
mrock::symbolic_operators::MomentumSymbol::name_type, 106
operator!=
mrock::symbolic_operators, 28–31
mrock::symbolic_operators::Momentum, 90
operator<
mrock::symbolic_operators, 41
operator<<
mrock::symbolic_operators, 41–49
mrock::symbolic_operators::Term, 156
operator<=>
mrock::symbolic_operators::IndexWrapper, 77
mrock::symbolic_operators::MomentumSymbol, 104
mrock::symbolic_operators::MomentumSymbol::name_type, 106, 107
mrock::symbolic_operators::SymbolicSum< SumIndex >, 131
operator>
mrock::symbolic_operators, 52
operator>>
mrock::symbolic_operators, 53
operator*
mrock::symbolic_operators, 31, 32
operator*=
mrock::symbolic_operators::Momentum, 90
mrock::symbolic_operators::MomentumList, 99
operator+
mrock::symbolic_operators, 32–35
operator+=
mrock::symbolic_operators, 35, 36
mrock::symbolic_operators::Momentum, 91
operator-
mrock::symbolic_operators, 37–39
operator-=
mrock::symbolic_operators, 39, 40
mrock::symbolic_operators::Momentum, 91
operator==
mrock::symbolic_operators, 49–52
mrock::symbolic_operators::Momentum, 91
operators
mrock::symbolic_operators::Term, 158
mrock::symbolic_operators::WickTerm, 182
OperatorType
mrock::symbolic_operators, 21
other
mrock::symbolic_operators::IndexComparison, 75

parse_interaction_string
mrock::symbolic_operators::Coefficient, 69
parse_string
mrock::symbolic_operators::Coefficient, 69
perform_comparison
sym_op_test::SymOpTest, 133
perform_operator_swap
mrock::symbolic_operators::Term, 152
perform_test
sym_op_test::SymOpTest, 134
prepare_wick
mrock::symbolic_operators, 53
print
mrock::symbolic_operators::Term, 152
push_back
mrock::symbolic_operators::SumContainer, 126, 127

Q_changes_sign
mrock::symbolic_operators::Coefficient, 74

R_SPIN
WickTerm.cpp, 208
RealInteraction
mrock::symbolic_operators::Coefficient, 70
RealInversionSymmetric
mrock::symbolic_operators::Coefficient, 70
remove_contribution
mrock::symbolic_operators::Momentum, 92
remove_delta_is_one
mrock::symbolic_operators, 53
remove_delta_squared
mrock::symbolic_operators, 54
remove_double_occurances
mrock::symbolic_operators, 54
remove_momentum_contribution
mrock::symbolic_operators::Coefficient, 71
mrock::symbolic_operators::Operator, 114
mrock::symbolic_operators::Term, 152

mrock::symbolic_operators::WickOperator, 163
mrock::symbolic_operators::WickTerm, 178
remove_zeros
mrock::symbolic_operators::Momentum, 92
mrock::symbolic_operators::MomentumList, 99
rename_indizes
mrock::symbolic_operators::Term, 153
rename_momenta
mrock::symbolic_operators, 55
mrock::symbolic_operators::Term, 153
rename_sums
mrock::symbolic_operators::Term, 153
mrock::symbolic_operators::WickTerm, 179
replace_occurances
mrock::symbolic_operators::Momentum, 92
mrock::symbolic_operators::MomentumList, 99
results
mrock::symbolic_operators::TemplateResult, 140
RIGHT
WickTerm.cpp, 209

save_as_comparison
sym_op_test::SymOpTest, 134
SC_Type
mrock::symbolic_operators, 21
second
mrock::symbolic_operators::KroneckerDelta< T >, 82
serialize
mrock::symbolic_operators::Coefficient, 71
mrock::symbolic_operators::IndexWrapper, 78
mrock::symbolic_operators::KroneckerDelta< T >, 81
mrock::symbolic_operators::Momentum, 93
mrock::symbolic_operators::MomentumList, 101
mrock::symbolic_operators::MomentumSymbol, 104
mrock::symbolic_operators::MomentumSymbol::name_type, 107
mrock::symbolic_operators::Operator, 115
mrock::symbolic_operators::SumContainer, 127
mrock::symbolic_operators::SymbolicSum< SumIndex >, 131
mrock::symbolic_operators::Term, 154
mrock::symbolic_operators::WickOperator, 163
mrock::symbolic_operators::WickTerm, 179
mrock::symbolic_operators::WickTermCollector, 184
set_deltas
mrock::symbolic_operators::Term, 154
mrock::symbolic_operators::WickTerm, 179
set_first_index
mrock::symbolic_operators::Operator, 115
Sigma
mrock::symbolic_operators, 20
SigmaPrime
mrock::symbolic_operators, 20
sort
mrock::symbolic_operators::Momentum, 93
mrock::symbolic_operators::MomentumList, 101
mrock::symbolic_operators::Term, 154
mrock::symbolic_operators::WickTerm, 180
sources/Coefficient.cpp, 202
sources/IndexWrapper.cpp, 202
sources/Momentum.cpp, 203
sources/MomentumList.cpp, 203
sources/Operator.cpp, 203
sources/OperatorType.cpp, 204
sources/SumContainer.cpp, 204
sources/Term.cpp, 204
sources/TermLoader.cpp, 205
sources/Wick.cpp, 206
sources/WickOperator.cpp, 206
sources/WickOperatorTemplate.cpp, 207
sources/WickSymmetry.cpp, 207
sources/WickTerm.cpp, 207
SpinDown
mrock::symbolic_operators, 20
spins
mrock::symbolic_operators::SumContainer, 128
SpinUp
mrock::symbolic_operators, 20
string_parser
mrock::symbolic_operators::WickTerm, 180
string_to_index
mrock::symbolic_operators, 56
string_to_wick
mrock::symbolic_operators, 56
summations
mrock::symbolic_operators::SymbolicSum< SumIndex >, 132
sums
mrock::symbolic_operators::Term, 158
mrock::symbolic_operators::WickTerm, 182
swap_momenta
mrock::symbolic_operators::Term, 154
sym_op_test, 57
sym_op_test::SymOpTest, 132
COMPARE_DIR, 134
load_and_test, 133
perform_comparison, 133
perform_test, 134
save_as_comparison, 134
SymOpTest, 133
SymbolicSum
mrock::symbolic_operators::SymbolicSum< SumIndex >, 129, 130
SymOpTest
sym_op_test::SymOpTest, 133

TemplateResult
mrock::symbolic_operators::TemplateResult, 136
temporary_operators
mrock::symbolic_operators::WickTerm, 182
Term
mrock::symbolic_operators::Term, 143–145, 147, 148
Term.cpp

fill_reciever, 205
Term.hpp
_TERM_TRACKER_ATTRIBUTE, 196
_TERM_TRACKER_PARAMETER, 196
CLEAR_TRACKED, 197
IF_IS_TERM_TRACKED, 197
tests/bosons.cpp, 209
tests/compare_test.hpp, 211
tests/continuum.cpp, 211
to_string
mrock::symbolic_operators::Momentum, 93
to_string_without_prefactor
mrock::symbolic_operators, 55
mrock::symbolic_operators::Term, 155
transform_momentum_sum
mrock::symbolic_operators::Term, 155
type
mrock::symbolic_operators::WickOperator, 165
mrock::symbolic_operators::WickOperatorTemplate, 168
TypeA
mrock::symbolic_operators, 20
TypeB
mrock::symbolic_operators, 20
TypeC
mrock::symbolic_operators, 20

Undefined_Type
mrock::symbolic_operators, 21
UndefinedIndex
mrock::symbolic_operators, 20
use_symmetric_interaction_exchange
mrock::symbolic_operators::Coefficient, 72
use_symmetric_interaction_inversion
mrock::symbolic_operators::Coefficient, 72
uses
mrock::symbolic_operators::Momentum, 93
uses_index
mrock::symbolic_operators::Coefficient, 72
mrock::symbolic_operators::WickOperator, 164
mrock::symbolic_operators::WickTerm, 180

VECTOR_WRAPPER_FILL_MEMBERS
mrock::symbolic_operators::IndexWrapper, 78
mrock::symbolic_operators::Momentum, 94
mrock::symbolic_operators::SymbolicSum< SumIndex >, 132

which_operator_depends_on
mrock::symbolic_operators::WickTerm, 181
which_term
mrock::symbolic_operators::bad_term_exception, 61
wick_processor
mrock::symbolic_operators, 55
WickOperator
mrock::symbolic_operators::WickOperator, 161, 162
wicks_theorem
mrock::symbolic_operators, 56
WickTerm
mrock::symbolic_operators::Term, 157
mrock::symbolic_operators::WickTerm, 173, 174
WickTerm.cpp
L_SPIN, 208
LEFT, 208
R_SPIN, 208
RIGHT, 209
with_momentum
mrock::symbolic_operators::Operator, 115, 116