

**Mule**

1.0.513

Generated by Doxygen 1.7.6.1

Wed Apr 15 2015 15:15:24



# Contents

<b>1 Multiscale</b>	<b>1</b>
1.1 Brief description . . . . .	1
1.2 Contact . . . . .	1
<b>2 Namespace Index</b>	<b>3</b>
2.1 Namespace List . . . . .	3
<b>3 Class Index</b>	<b>5</b>
3.1 Class Hierarchy . . . . .	5
<b>4 Class Index</b>	<b>11</b>
4.1 Class List . . . . .	11
<b>5 Namespace Documentation</b>	<b>23</b>
5.1 multiscale Namespace Reference . . . . .	23
5.1.1 Enumeration Type Documentation . . . . .	25
5.1.1.1 ColourCode . . . . .	25
5.1.1.2 NumberIteratorType . . . . .	26
5.1.1.3 UnixColourCode . . . . .	26
5.1.1.4 WindowsColourCode . . . . .	26
5.1.2 Variable Documentation . . . . .	27
5.1.2.1 ERR_INDEX_OUT_OF_BOUNDS_BEGIN . . . . .	27
5.1.2.2 ERR_INDEX_OUT_OF_BOUNDS_END . . . . .	27
5.1.2.3 ERR_MSG . . . . .	27
5.1.2.4 ERR_UNDEFINED_ENUM_VALUE . . . . .	27
5.1.2.5 ERR_UNIMPLEMENTED_METHOD . . . . .	28
5.1.2.6 EXEC_ERR_CODE . . . . .	28

5.1.2.7	EXEC_SUCCESS_CODE . . . . .	28
5.2	multiscale::analysis Namespace Reference . . . . .	28
5.2.1	Typedef Documentation . . . . .	29
5.2.1.1	Polygon . . . . .	29
5.2.2	Enumeration Type Documentation . . . . .	30
5.2.2.1	DBSCANPointClusteringTag . . . . .	30
5.2.2.2	Shape2D . . . . .	30
5.2.2.3	SpatialEntityPseudo3DType . . . . .	30
5.3	multiscale::verification Namespace Reference . . . . .	30
5.3.1	Typedef Documentation . . . . .	43
5.3.1.1	ConstraintAttributeType . . . . .	43
5.3.1.2	FilterNumericMeasureAttributeType . . . . .	43
5.3.1.3	LogicPropertyAttributeType . . . . .	43
5.3.1.4	NumericMeasureCollectionType . . . . .	44
5.3.1.5	NumericMeasureType . . . . .	44
5.3.1.6	NumericSpatialMeasureType . . . . .	44
5.3.1.7	NumericStatisticalMeasureType . . . . .	44
5.3.1.8	PrimaryConstraintAttributeType . . . . .	44
5.3.1.9	PrimaryLogicPropertyAttributeType . . . . .	45
5.3.1.10	PrimaryNumericMeasureAttributeType . . . . .	45
5.3.1.11	SpatialMeasureCollectionType . . . . .	45
5.3.1.12	SubsetAttributeType . . . . .	45
5.3.1.13	TemporalNumericCollectionType . . . . .	45
5.3.1.14	TemporalNumericMeasureType . . . . .	46
5.3.1.15	TimeseriesComponentType . . . . .	46
5.3.2	Enumeration Type Documentation . . . . .	46
5.3.2.1	ApproximateBayesianModelCheckingResult . . . . .	46
5.3.2.2	BayesianModelCheckingResult . . . . .	46
5.3.2.3	BinaryNumericMeasureType . . . . .	47
5.3.2.4	BinaryStatisticalMeasureType . . . . .	47
5.3.2.5	BinaryStatisticalQuantileMeasureType . . . . .	47
5.3.2.6	ChangeMeasureType . . . . .	47
5.3.2.7	ComparatorType . . . . .	48
5.3.2.8	HeterogeneousTimeseriesComponentType . . . . .	48

5.3.2.9	HomogeneousTimeseriesComponentType . . . . .	48
5.3.2.10	HomogeneousTimeseriesMeasureType . . . . .	48
5.3.2.11	SimilarityMeasureType . . . . .	49
5.3.2.12	SpatialMeasureType . . . . .	49
5.3.2.13	StatisticalModelCheckingResult . . . . .	49
5.3.2.14	SubsetOperationType . . . . .	50
5.3.2.15	SubsetSpecificType . . . . .	50
5.3.2.16	TimeseriesMeasureType . . . . .	50
5.3.2.17	UnaryNumericMeasureType . . . . .	50
5.3.2.18	UnaryStatisticalMeasureType . . . . .	51
5.3.3	Function Documentation . . . . .	51
5.3.3.1	operator<< . . . . .	51
5.3.3.2	operator<< . . . . .	52
5.3.3.3	operator<< . . . . .	52
5.3.3.4	operator<< . . . . .	52
5.3.3.5	operator<< . . . . .	53
5.3.3.6	operator<< . . . . .	53
5.3.3.7	operator<< . . . . .	53
5.3.3.8	operator<< . . . . .	54
5.3.3.9	operator<< . . . . .	54
5.3.3.10	operator<< . . . . .	54
5.3.3.11	operator<< . . . . .	55
5.3.3.12	operator<< . . . . .	55
5.3.3.13	operator<< . . . . .	55
5.3.3.14	operator<< . . . . .	56
5.3.3.15	operator<< . . . . .	56
5.3.4	Variable Documentation . . . . .	56
5.3.4.1	handleProbabilityError . . . . .	56
5.3.4.2	handleUnexpectedTokenError . . . . .	57
5.3.4.3	NR_SPATIAL_MEASURE_TYPES . . . . .	57
5.3.4.4	NR_SUBSET_SPECIFIC_TYPES . . . . .	57
5.3.4.5	WRN_LOGIC_PROPERTY_EVAL_FALSE . . . . .	58
5.3.4.6	WRN_OUTPUT_SEPARATOR . . . . .	58
5.4	multiscale::verification::spatialmeasure Namespace Reference . . . . .	58

5.4.1	Function Documentation . . . . .	59
5.4.1.1	computeSpatialMeasureType . . . . .	59
5.4.1.2	computeSpatialMeasureTypeIndex . . . . .	59
5.4.1.3	getMaxValidSpatialMeasureValue . . . . .	59
5.4.1.4	getMinValidSpatialMeasureValue . . . . .	60
5.4.1.5	toString . . . . .	60
5.4.1.6	toString . . . . .	60
5.4.1.7	validateSpatialMeasureType . . . . .	61
5.4.1.8	validateSpatialMeasureTypeIndex . . . . .	61
5.5	multiscale::verification::subsetspecific Namespace Reference . . . . .	61
5.5.1	Function Documentation . . . . .	62
5.5.1.1	computeSubsetSpecificType . . . . .	62
5.5.1.2	computeSubsetSpecificTypeIndex . . . . .	62
5.5.1.3	toString . . . . .	63
5.5.1.4	toString . . . . .	63
5.5.1.5	validateSubsetSpecificType . . . . .	63
5.5.1.6	validateSubsetSpecificTypeIndex . . . . .	64
5.6	multiscale::visualisation Namespace Reference . . . . .	64
5.7	multiscaletest Namespace Reference . . . . .	65
5.8	multiscaletest::verification Namespace Reference . . . . .	66
5.8.1	Function Documentation . . . . .	66
5.8.1.1	parseInputString . . . . .	66
<b>6</b>	<b>Class Documentation</b> . . . . .	<b>67</b>
6.1	multiscale::verification::AbstractSyntaxTree Class Reference . . . . .	67
6.1.1	Detailed Description . . . . .	69
6.1.2	Constructor & Destructor Documentation . . . . .	69
6.1.2.1	AbstractSyntaxTree . . . . .	69
6.1.2.2	~AbstractSyntaxTree . . . . .	69
6.1.3	Member Function Documentation . . . . .	69
6.1.3.1	evaluate . . . . .	69
6.1.3.2	getComparator . . . . .	70
6.1.3.3	getProbability . . . . .	70
6.1.3.4	initialiseTree . . . . .	70

6.1.4	Member Data Documentation . . . . .	71
6.1.4.1	ERR_ABSTRACT_SYNTAX_TREE_NOT_INITIALISED . . . . .	71
6.1.4.2	isInitialised . . . . .	71
6.1.4.3	probabilisticLogicProperty . . . . .	71
6.2	multiscale::AdditionOperation Class Reference . . . . .	71
6.2.1	Detailed Description . . . . .	72
6.2.2	Member Function Documentation . . . . .	72
6.2.2.1	operator() . . . . .	72
6.3	multiscale::AlgorithmException Class Reference . . . . .	72
6.3.1	Detailed Description . . . . .	75
6.3.2	Constructor & Destructor Documentation . . . . .	75
6.3.2.1	AlgorithmException . . . . .	75
6.3.2.2	AlgorithmException . . . . .	75
6.3.2.3	AlgorithmException . . . . .	75
6.4	multiscale::verification::AndConstraintAttribute Class Reference . . . . .	75
6.4.1	Detailed Description . . . . .	76
6.4.2	Member Data Documentation . . . . .	76
6.4.2.1	constraint . . . . .	76
6.5	multiscale::verification::AndLogicPropertyAttribute Class Reference . . . . .	76
6.5.1	Detailed Description . . . . .	76
6.5.2	Member Data Documentation . . . . .	76
6.5.2.1	logicProperty . . . . .	76
6.6	multiscale::visualisation::AnnularSector Class Reference . . . . .	77
6.6.1	Detailed Description . . . . .	77
6.6.2	Constructor & Destructor Documentation . . . . .	78
6.6.2.1	AnnularSector . . . . .	78
6.6.2.2	~AnnularSector . . . . .	78
6.6.3	Member Function Documentation . . . . .	78
6.6.3.1	getConcentration . . . . .	78
6.6.3.2	getEndingAngle . . . . .	78
6.6.3.3	getEndingRadius . . . . .	78
6.6.3.4	getStartingAngle . . . . .	78
6.6.3.5	getStartingRadius . . . . .	79

6.6.3.6	initialise . . . . .	79
6.6.3.7	toString . . . . .	79
6.6.4	Member Data Documentation . . . . .	79
6.6.4.1	concentration . . . . .	79
6.6.4.2	endingAngle . . . . .	80
6.6.4.3	endingRadius . . . . .	80
6.6.4.4	startingAngle . . . . .	80
6.6.4.5	startingRadius . . . . .	80
6.7	multiscale::verification::ApproximateBayesianModelChecker Class Reference . . . . .	80
6.7.1	Detailed Description . . . . .	85
6.7.2	Constructor & Destructor Documentation . . . . .	85
6.7.2.1	ApproximateBayesianModelChecker . . . . .	85
6.7.2.2	~ApproximateBayesianModelChecker . . . . .	85
6.7.3	Member Function Documentation . . . . .	85
6.7.3.1	acceptsMoreTraces . . . . .	85
6.7.3.2	doesPropertyHold . . . . .	86
6.7.3.3	doesPropertyHoldConsideringResult . . . . .	86
6.7.3.4	getDetailedResults . . . . .	86
6.7.3.5	getDetailedUpdatedResults . . . . .	86
6.7.3.6	initialise . . . . .	87
6.7.3.7	isModelCheckingResultTrueConsideringComparator . . . . .	87
6.7.3.8	isValidShapeParameter . . . . .	87
6.7.3.9	requiresMoreTraces . . . . .	88
6.7.3.10	updateDerivedModelCheckerForFalseEvaluation . . . . .	88
6.7.3.11	updateDerivedModelCheckerForTrueEvaluation . . . . .	88
6.7.3.12	updateMean . . . . .	88
6.7.3.13	updateMeanAndVariance . . . . .	88
6.7.3.14	updateModelCheckingResult . . . . .	89
6.7.3.15	updateModelCheckingResult . . . . .	89
6.7.3.16	updateModelCheckingResultEnoughTraces . . . . .	89
6.7.3.17	updateModelCheckingResultNotEnoughTraces . . . . .	90
6.7.3.18	updateVariance . . . . .	90
6.7.3.19	validateInput . . . . .	90

6.7.3.20	validateShapeParameters . . . . .	90
6.7.3.21	validateVarianceThreshold . . . . .	91
6.7.4	Member Data Documentation . . . . .	91
6.7.4.1	alpha . . . . .	91
6.7.4.2	beta . . . . .	91
6.7.4.3	ERR_SHAPE_PARAMETERS_BEGIN . . . . .	92
6.7.4.4	ERR_SHAPE_PARAMETERS_END . . . . .	92
6.7.4.5	ERR_SHAPE_PARAMETERS_MIDDLE . . . . .	92
6.7.4.6	ERR_UNEXPECTED_MODEL_CHECKING_RESULT . . . . .	92
6.7.4.7	ERR_VARIANCE_THRESHOLD_BEGIN . . . . .	92
6.7.4.8	ERR_VARIANCE_THRESHOLD_END . . . . .	92
6.7.4.9	mean . . . . .	93
6.7.4.10	modelCheckingResult . . . . .	93
6.7.4.11	MSG_OUTPUT_MORE_TRACES_REQUIRED . . . . .	93
6.7.4.12	MSG_OUTPUT_RESULT_BEGIN . . . . .	93
6.7.4.13	MSG_OUTPUT_RESULT_END . . . . .	93
6.7.4.14	MSG_OUTPUT_RESULT_MIDDLE1 . . . . .	94
6.7.4.15	MSG_OUTPUT_RESULT_MIDDLE2 . . . . .	94
6.7.4.16	MSG_OUTPUT_SEPARATOR . . . . .	94
6.7.4.17	probability . . . . .	94
6.7.4.18	variance . . . . .	94
6.7.4.19	varianceThreshold . . . . .	94
6.8	multiscale::verification::ApproximateBayesianModelCheckerFactory - Class Reference . . . . .	95
6.8.1	Detailed Description . . . . .	96
6.8.2	Constructor & Destructor Documentation . . . . .	97
6.8.2.1	ApproximateBayesianModelCheckerFactory . . . . .	97
6.8.2.2	~ApproximateBayesianModelCheckerFactory . . . . .	97
6.8.3	Member Function Documentation . . . . .	97
6.8.3.1	createInstance . . . . .	97
6.8.4	Member Data Documentation . . . . .	97
6.8.4.1	alpha . . . . .	97
6.8.4.2	beta . . . . .	98

6.8.4.3	varianceThreshold . . . . .	98
6.9	multiscaletest::ApproximateBayesianModelCheckerTest Class Reference	98
6.9.1	Detailed Description . . . . .	101
6.9.2	Constructor & Destructor Documentation . . . . .	101
6.9.2.1	ApproximateBayesianModelCheckerTest . . . . .	101
6.9.3	Member Function Documentation . . . . .	101
6.9.3.1	InitialiseModelChecker . . . . .	101
6.9.3.2	SetAlphaParamForBetaPrior . . . . .	101
6.9.3.3	SetBetaParamForBetaPrior . . . . .	102
6.9.3.4	SetVarianceThreshold . . . . .	102
6.9.4	Member Data Documentation . . . . .	102
6.9.4.1	alphaParamForBetaPrior . . . . .	102
6.9.4.2	betaParamForBetaPrior . . . . .	102
6.9.4.3	varianceThreshold . . . . .	103
6.10	multiscale::verification::ApproximateProbabilisticModelChecker Class - Reference . . . . .	103
6.10.1	Detailed Description . . . . .	107
6.10.2	Constructor & Destructor Documentation . . . . .	107
6.10.2.1	ApproximateProbabilisticModelChecker . . . . .	107
6.10.2.2	~ApproximateProbabilisticModelChecker . . . . .	107
6.10.3	Member Function Documentation . . . . .	107
6.10.3.1	acceptsMoreTraces . . . . .	108
6.10.3.2	doesPropertyHold . . . . .	108
6.10.3.3	doesPropertyHoldConsideringProbabilityComparator .	108
6.10.3.4	getDetailedResults . . . . .	108
6.10.3.5	initialise . . . . .	109
6.10.3.6	initialiseNumberOfRequiredTraces . . . . .	109
6.10.3.7	isBetweenZeroAndOne . . . . .	109
6.10.3.8	requiresMoreTraces . . . . .	109
6.10.3.9	updateDerivedModelCheckerForFalseEvaluation . .	110
6.10.3.10	updateDerivedModelCheckerForTrueEvaluation . .	110
6.10.3.11	validateInput . . . . .	110
6.10.4	Member Data Documentation . . . . .	111
6.10.4.1	delta . . . . .	111

6.10.4.2	epsilon . . . . .	111
6.10.4.3	ERR_INVALID_INPUT_BEGIN . . . . .	111
6.10.4.4	ERR_INVALID_INPUT_END . . . . .	111
6.10.4.5	ERR_INVALID_INPUT_MIDDLE . . . . .	111
6.10.4.6	MSG_OUTPUT_MORE_TRACES_REQUIRED . . . . .	112
6.10.4.7	MSG_OUTPUT_RESULT_BEGIN . . . . .	112
6.10.4.8	MSG_OUTPUT_RESULT_END . . . . .	112
6.10.4.9	MSG_OUTPUT_RESULT_MIDDLE1 . . . . .	112
6.10.4.10	MSG_OUTPUT_RESULT_MIDDLE2 . . . . .	112
6.10.4.11	MSG_OUTPUT_SEPARATOR . . . . .	112
6.10.4.12	nrOfRequiredTraces . . . . .	113
6.10.4.13	probability . . . . .	113
6.11	multiscale::verification::ApproximateProbabilisticModelCheckerFactory - Class Reference . . . . .	113
6.11.1	Detailed Description . . . . .	115
6.11.2	Constructor & Destructor Documentation . . . . .	116
6.11.2.1	ApproximateProbabilisticModelCheckerFactory . . . . .	116
6.11.2.2	~ApproximateProbabilisticModelCheckerFactory . . . . .	116
6.11.3	Member Function Documentation . . . . .	116
6.11.3.1	createInstance . . . . .	116
6.11.4	Member Data Documentation . . . . .	116
6.11.4.1	delta . . . . .	116
6.11.4.2	epsilon . . . . .	117
6.12	multiscaletest::ApproximateProbabilisticModelCheckerTest Class - Reference . . . . .	117
6.12.1	Detailed Description . . . . .	120
6.12.2	Constructor & Destructor Documentation . . . . .	120
6.12.2.1	ApproximateProbabilisticModelCheckerTest . . . . .	120
6.12.3	Member Function Documentation . . . . .	120
6.12.3.1	InitialiseModelChecker . . . . .	120
6.12.3.2	SetDelta . . . . .	120
6.12.3.3	SetEpsilon . . . . .	121
6.12.4	Member Data Documentation . . . . .	121
6.12.4.1	delta . . . . .	121

6.12.4.2	epsilon	121
6.13	multiscale::verification::BayesianModelChecker Class Reference	121
6.13.1	Detailed Description	126
6.13.2	Constructor & Destructor Documentation	126
6.13.2.1	BayesianModelChecker	126
6.13.2.2	~BayesianModelChecker	127
6.13.3	Member Function Documentation	127
6.13.3.1	acceptsMoreTraces	127
6.13.3.2	computeBayesFactorValue	127
6.13.3.3	computeBinomialPDF	127
6.13.3.4	computeMaximumBinomialPDF	128
6.13.3.5	doesPropertyHold	128
6.13.3.6	doesPropertyHoldConsideringProbabilityComparator	128
6.13.3.7	doesPropertyHoldConsideringResult	129
6.13.3.8	getDetailedResults	129
6.13.3.9	getDetailedUpdatedResults	129
6.13.3.10	indicatorFunction	129
6.13.3.11	initialise	130
6.13.3.12	isValidShapeParameter	130
6.13.3.13	requiresMoreTraces	130
6.13.3.14	updateDerivedModelCheckerForFalseEvaluation	131
6.13.3.15	updateDerivedModelCheckerForTrueEvaluation	131
6.13.3.16	updateModelCheckingResult	131
6.13.3.17	updateModelCheckingResult	131
6.13.3.18	updateModelCheckingResultEnoughTraces	132
6.13.3.19	updateModelCheckingResultNotEnoughTraces	132
6.13.3.20	updateTypeIErrorUpperBound	132
6.13.3.21	validateBayesFactorThreshold	132
6.13.3.22	validateInput	133
6.13.3.23	validateShapeParameters	133
6.13.4	Member Data Documentation	133
6.13.4.1	alpha	134
6.13.4.2	bayesFactorThreshold	134
6.13.4.3	bayesFactorThresholdInverse	134

6.13.4.4	beta	134
6.13.4.5	ERR_BAYES_FACTOR_THRESHOLD_BEGIN	134
6.13.4.6	ERR_BAYES_FACTOR_THRESHOLD_END	135
6.13.4.7	ERR_SHAPE_PARAMETERS_BEGIN	135
6.13.4.8	ERR_SHAPE_PARAMETERS_END	135
6.13.4.9	ERR_SHAPE_PARAMETERS_MIDDLE	135
6.13.4.10	ERR_UNEXPECTED_MODEL_CHECKING_RESULT	135
6.13.4.11	modelCheckingResult	135
6.13.4.12	MSG_OUTPUT_MORE_TRACES_REQUIRED	136
6.13.4.13	MSG_OUTPUT_RESULT_BEGIN	136
6.13.4.14	MSG_OUTPUT_RESULT_END	136
6.13.4.15	MSG_OUTPUT_RESULT_MIDDLE1	136
6.13.4.16	MSG_OUTPUT_RESULT_MIDDLE2	136
6.13.4.17	MSG_OUTPUT_RESULT_MIDDLE3	137
6.13.4.18	MSG_OUTPUT_SEPARATOR	137
6.13.4.19	probability	137
6.13.4.20	typeIErrorUpperBound	137
6.14	multiscale::verification::BayesianModelCheckerFactory Class Reference	137
6.14.1	Detailed Description	140
6.14.2	Constructor & Destructor Documentation	140
6.14.2.1	BayesianModelCheckerFactory	140
6.14.2.2	~BayesianModelCheckerFactory	140
6.14.3	Member Function Documentation	140
6.14.3.1	createInstance	140
6.14.4	Member Data Documentation	140
6.14.4.1	alpha	140
6.14.4.2	bayesFactorThreshold	141
6.14.4.3	beta	141
6.15	multiscaletest::BayesianModelCheckerTest Class Reference	141
6.15.1	Detailed Description	144
6.15.2	Constructor & Destructor Documentation	144
6.15.2.1	BayesianModelCheckerTest	144
6.15.3	Member Function Documentation	144

6.15.3.1	InitialiseModelChecker . . . . .	144
6.15.3.2	SetAlphaParamForBetaPrior . . . . .	144
6.15.3.3	SetBayesFactorThreshold . . . . .	145
6.15.3.4	SetBetaParamForBetaPrior . . . . .	145
6.15.4	Member Data Documentation . . . . .	145
6.15.4.1	alphaParamForBetaPrior . . . . .	145
6.15.4.2	bayesFactorThreshold . . . . .	145
6.15.4.3	betaParamForBetaPrior . . . . .	145
6.16	multiscale::BetaDistribution Class Reference . . . . .	146
6.16.1	Detailed Description . . . . .	148
6.16.2	Member Function Documentation . . . . .	148
6.16.2.1	cdf . . . . .	148
6.16.2.2	computeCdf . . . . .	148
6.16.2.3	isValidShapeParameter . . . . .	149
6.16.2.4	validateShapeParameters . . . . .	149
6.16.3	Member Data Documentation . . . . .	149
6.16.3.1	ERR_SHAPE_PARAMETERS_BEGIN . . . . .	150
6.16.3.2	ERR_SHAPE_PARAMETERS_END . . . . .	150
6.16.3.3	ERR_SHAPE_PARAMETERS_MIDDLE . . . . .	150
6.17	multiscale::verification::BinaryNumericFilterAttribute Class Reference . . . . .	150
6.17.1	Detailed Description . . . . .	151
6.17.2	Member Data Documentation . . . . .	151
6.17.2.1	binaryNumericMeasure . . . . .	151
6.17.2.2	firstFilterNumericMeasure . . . . .	152
6.17.2.3	secondFilterNumericMeasure . . . . .	152
6.18	multiscale::verification::BinaryNumericMeasureAttribute Class Reference . . . . .	152
6.18.1	Detailed Description . . . . .	152
6.18.2	Member Data Documentation . . . . .	152
6.18.2.1	binaryNumericMeasureType . . . . .	152
6.19	multiscale::verification::BinaryNumericMeasureGrammar< Iterator > - Class Template Reference . . . . .	153
6.19.1	Detailed Description . . . . .	155
6.19.2	Constructor & Destructor Documentation . . . . .	156
6.19.2.1	BinaryNumericMeasureGrammar . . . . .	156

6.19.3 Member Function Documentation . . . . .	156
6.19.3.1 assignNamesToRules . . . . .	156
6.19.3.2 initialise . . . . .	156
6.19.3.3 initialiseDebugSupport . . . . .	156
6.19.3.4 initialiseGrammar . . . . .	156
6.19.3.5 initialiseRulesDebugging . . . . .	157
6.19.4 Member Data Documentation . . . . .	157
6.19.4.1 binaryNumericMeasureRule . . . . .	157
6.19.4.2 binaryNumericMeasureTypeParser . . . . .	157
6.20 multiscale::verification::BinaryNumericMeasureTypeParser Struct - Reference . . . . .	157
6.20.1 Detailed Description . . . . .	157
6.20.2 Constructor & Destructor Documentation . . . . .	158
6.20.2.1 BinaryNumericMeasureTypeParser . . . . .	158
6.21 multiscale::verification::BinaryNumericNumericAttribute Class Reference . . . . .	158
6.21.1 Detailed Description . . . . .	159
6.21.2 Member Data Documentation . . . . .	159
6.21.2.1 binaryNumericMeasure . . . . .	159
6.21.2.2 firstNumericMeasure . . . . .	160
6.21.2.3 secondNumericMeasure . . . . .	160
6.22 multiscale::verification::BinaryNumericSpatialAttribute Class Reference . . . . .	160
6.22.1 Detailed Description . . . . .	161
6.22.2 Member Data Documentation . . . . .	161
6.22.2.1 binaryNumericMeasure . . . . .	161
6.22.2.2 firstSpatialMeasureCollection . . . . .	161
6.22.2.3 secondSpatialMeasureCollection . . . . .	161
6.23 multiscale::verification::BinaryNumericTemporalAttribute Class Reference . . . . .	161
6.23.1 Detailed Description . . . . .	162
6.23.2 Member Data Documentation . . . . .	162
6.23.2.1 binaryNumericMeasure . . . . .	162
6.23.2.2 firstTemporalNumericMeasure . . . . .	163
6.23.2.3 secondTemporalNumericMeasure . . . . .	163
6.24 multiscale::verification::BinaryStatisticalMeasureAttribute Class - Reference . . . . .	163

6.24.1	Detailed Description . . . . .	163
6.24.2	Member Data Documentation . . . . .	163
6.24.2.1	binaryStatisticalMeasureType . . . . .	164
6.25	multiscale::verification::BinaryStatisticalMeasureGrammar< Iterator > - Class Template Reference . . . . .	164
6.25.1	Detailed Description . . . . .	166
6.25.2	Constructor & Destructor Documentation . . . . .	167
6.25.2.1	BinaryStatisticalMeasureGrammar . . . . .	167
6.25.3	Member Function Documentation . . . . .	167
6.25.3.1	assignNamesToRules . . . . .	167
6.25.3.2	initialise . . . . .	167
6.25.3.3	initialiseDebugSupport . . . . .	167
6.25.3.4	initialiseGrammar . . . . .	167
6.25.3.5	initialiseRulesDebugging . . . . .	168
6.25.4	Member Data Documentation . . . . .	168
6.25.4.1	binaryStatisticalMeasureRule . . . . .	168
6.25.4.2	binaryStatisticalMeasureTypeParser . . . . .	168
6.26	multiscale::verification::BinaryStatisticalMeasureTypeParser Struct - Reference . . . . .	168
6.26.1	Detailed Description . . . . .	168
6.26.2	Constructor & Destructor Documentation . . . . .	169
6.26.2.1	BinaryStatisticalMeasureTypeParser . . . . .	169
6.27	multiscale::verification::BinaryStatisticalNumericAttribute Class Reference	169
6.27.1	Detailed Description . . . . .	170
6.27.2	Member Data Documentation . . . . .	170
6.27.2.1	binaryStatisticalMeasure . . . . .	170
6.27.2.2	firstNumericMeasureCollection . . . . .	170
6.27.2.3	secondNumericMeasureCollection . . . . .	170
6.28	multiscale::verification::BinaryStatisticalQuantileMeasureAttribute Class Reference . . . . .	170
6.28.1	Detailed Description . . . . .	171
6.28.2	Member Data Documentation . . . . .	171
6.28.2.1	binaryStatisticalQuantileMeasureType . . . . .	171
6.29	multiscale::verification::BinaryStatisticalQuantileMeasureGrammar< Iterator > Class Template Reference . . . . .	171

6.29.1	Detailed Description	173
6.29.2	Constructor & Destructor Documentation	173
6.29.2.1	BinaryStatisticalQuantileMeasureGrammar	173
6.29.3	Member Function Documentation	174
6.29.3.1	assignNamesToRules	174
6.29.3.2	initialise	174
6.29.3.3	initialiseDebugSupport	174
6.29.3.4	initialiseGrammar	174
6.29.3.5	initialiseRulesDebugging	174
6.29.4	Member Data Documentation	174
6.29.4.1	binaryStatisticalQuantileMeasureRule	175
6.29.4.2	binaryStatisticalQuantileMeasureTypeParser	175
6.30	multiscale::verification::BinaryStatisticalQuantileMeasureTypeParser Struct Reference	175
6.30.1	Detailed Description	175
6.30.2	Constructor & Destructor Documentation	175
6.30.2.1	BinaryStatisticalQuantileMeasureTypeParser	175
6.31	multiscale::verification::BinaryStatisticalQuantileNumericAttribute Class Reference	176
6.31.1	Detailed Description	176
6.31.2	Member Data Documentation	176
6.31.2.1	binaryStatisticalQuantileMeasure	177
6.31.2.2	numericMeasureCollection	177
6.31.2.3	parameter	177
6.32	multiscale::verification::BinaryStatisticalQuantileSpatialAttribute Class - Reference	177
6.32.1	Detailed Description	178
6.32.2	Member Data Documentation	178
6.32.2.1	binaryStatisticalQuantileMeasure	178
6.32.2.2	parameter	178
6.32.2.3	spatialMeasureCollection	179
6.33	multiscale::verification::BinaryStatisticalSpatialAttribute Class Reference	179
6.33.1	Detailed Description	179
6.33.2	Member Data Documentation	180
6.33.2.1	binaryStatisticalMeasure	180

6.33.2.2	firstSpatialMeasureCollection . . . . .	180
6.33.2.3	secondSpatialMeasureCollection . . . . .	180
6.34	multiscale::BinomialDistribution Class Reference . . . . .	180
6.34.1	Detailed Description . . . . .	183
6.34.2	Member Function Documentation . . . . .	183
6.34.2.1	cdf . . . . .	183
6.34.2.2	computeCdf . . . . .	184
6.34.2.3	computePdf . . . . .	184
6.34.2.4	pdf . . . . .	184
6.34.2.5	validateInput . . . . .	185
6.34.2.6	validateNrOfSuccesses . . . . .	185
6.34.3	Member Data Documentation . . . . .	186
6.34.3.1	ERR_NR_OF_SUCCESSES_BEGIN . . . . .	186
6.34.3.2	ERR_NR_OF_SUCCESSES_END . . . . .	186
6.34.3.3	ERR_NR_OF_SUCCESSES_MIDDLE . . . . .	186
6.35	multiscale::visualisation::CartesianToConcentrationsConverter Class Reference . . . . .	186
6.35.1	Detailed Description . . . . .	189
6.35.2	Constructor & Destructor Documentation . . . . .	189
6.35.2.1	CartesianToConcentrationsConverter . . . . .	189
6.35.2.2	~CartesianToConcentrationsConverter . . . . .	189
6.35.3	Member Function Documentation . . . . .	189
6.35.3.1	convert . . . . .	189
6.35.3.2	outputResults . . . . .	189
6.35.3.3	readConcentrations . . . . .	189
6.35.3.4	readHeaderLine . . . . .	190
6.35.3.5	readInputData . . . . .	190
6.35.4	Member Data Documentation . . . . .	190
6.35.4.1	concentrations . . . . .	190
6.35.4.2	ERR_CONC . . . . .	191
6.35.4.3	ERR_IN_EXTRA_DATA . . . . .	191
6.35.4.4	ERR_INPUT_OPEN . . . . .	191
6.35.4.5	ERR_NEG_SIM_TIME . . . . .	191
6.35.4.6	ERR_NONPOS_DIMENSION . . . . .	191

6.35.4.7 height . . . . .	191
6.35.4.8 inputfilepath . . . . .	192
6.35.4.9 OUTPUT_FILE_EXTENSION . . . . .	192
6.35.4.10 outputfilepath . . . . .	192
6.35.4.11 RADIUS_MAX . . . . .	192
6.35.4.12 RADIUS_MIN . . . . .	192
6.35.4.13 simulationTime . . . . .	192
6.35.4.14 width . . . . .	192
6.36 multiscale::visualisation::CartesianToPolarConverter Class Reference . . . . .	193
6.36.1 Detailed Description . . . . .	196
6.36.2 Constructor & Destructor Documentation . . . . .	196
6.36.2.1 CartesianToPolarConverter . . . . .	196
6.36.2.2 ~CartesianToPolarConverter . . . . .	196
6.36.3 Member Function Documentation . . . . .	196
6.36.3.1 convert . . . . .	196
6.36.3.2 outputResultsAsFile . . . . .	196
6.36.3.3 outputResultsAsScript . . . . .	197
6.36.3.4 readConcentrations . . . . .	197
6.36.3.5 readHeaderLine . . . . .	197
6.36.3.6 readInputData . . . . .	197
6.36.3.7 transformToAnnularSectors . . . . .	198
6.36.4 Member Data Documentation . . . . .	198
6.36.4.1 annularSectors . . . . .	198
6.36.4.2 concentrations . . . . .	198
6.36.4.3 ERR_CONC . . . . .	198
6.36.4.4 ERR_IN_EXTRA_DATA . . . . .	198
6.36.4.5 ERR_INPUT_OPEN . . . . .	199
6.36.4.6 ERR_NEG_SIM_TIME . . . . .	199
6.36.4.7 ERR_NONPOS_DIMENSION . . . . .	199
6.36.4.8 inputfilepath . . . . .	199
6.36.4.9 nrOfConcentricCircles . . . . .	199
6.36.4.10 nrOfSectors . . . . .	199
6.36.4.11 OUTPUT_FILE_EXTENSION . . . . .	200
6.36.4.12 outputfilepath . . . . .	200

6.36.4.13 RADIUS_MAX . . . . .	200
6.36.4.14 RADIUS_MIN . . . . .	200
6.36.4.15 simulationTime . . . . .	200
6.37 multiscale::verification::ChangeMeasureAttribute Class Reference . . . . .	200
6.37.1 Detailed Description . . . . .	201
6.37.2 Member Data Documentation . . . . .	201
6.37.2.1 changeMeasureType . . . . .	201
6.38 multiscale::verification::ChangeMeasureEvaluator Class Reference . . . . .	201
6.38.1 Detailed Description . . . . .	202
6.38.2 Member Function Documentation . . . . .	202
6.38.2.1 computeNumericMeasureValueChange . . . . .	202
6.38.2.2 computeTimeValueDifference . . . . .	203
6.38.2.3 evaluate . . . . .	203
6.38.2.4 evaluate . . . . .	204
6.39 multiscale::verification::ChangeMeasureGrammar< Iterator > Class - Template Reference . . . . .	205
6.39.1 Detailed Description . . . . .	206
6.39.2 Constructor & Destructor Documentation . . . . .	207
6.39.2.1 ChangeMeasureGrammar . . . . .	207
6.39.3 Member Function Documentation . . . . .	207
6.39.3.1 assignNamesToRules . . . . .	207
6.39.3.2 initialise . . . . .	207
6.39.3.3 initialiseDebugSupport . . . . .	207
6.39.3.4 initialiseGrammar . . . . .	208
6.39.3.5 initialiseRulesDebugging . . . . .	208
6.39.4 Member Data Documentation . . . . .	208
6.39.4.1 changeMeasureRule . . . . .	208
6.39.4.2 changeMeasureTypeParser . . . . .	208
6.40 multiscale::verification::ChangeMeasureTypeParser Struct Reference . . . . .	208
6.40.1 Detailed Description . . . . .	209
6.40.2 Constructor & Destructor Documentation . . . . .	209
6.40.2.1 ChangeMeasureTypeParser . . . . .	209
6.41 multiscale::verification::ChangeTemporalNumericCollectionAttribute - Class Reference . . . . .	209

6.41.1	Detailed Description	210
6.41.2	Member Data Documentation	210
6.41.2.1	changeMeasure	210
6.41.2.2	temporalNumericCollection	210
6.42	multiscale::verification::ChangeTemporalNumericMeasureAttribute - Class Reference	210
6.42.1	Detailed Description	211
6.42.2	Member Data Documentation	211
6.42.2.1	changeMeasure	211
6.42.2.2	comparator	212
6.42.2.3	IhsTemporalNumericMeasure	212
6.42.2.4	rhsTemporalNumericMeasure	212
6.43	multiscale::analysis::CircularityMeasure< PointType > Class Template Reference	212
6.43.1	Detailed Description	212
6.43.2	Member Function Documentation	213
6.43.2.1	compute	213
6.44	multiscale::analysis::CircularMatFactory Class Reference	213
6.44.1	Detailed Description	216
6.44.2	Constructor & Destructor Documentation	216
6.44.2.1	CircularMatFactory	216
6.44.2.2	~CircularMatFactory	216
6.44.3	Member Function Documentation	217
6.44.3.1	createCircularMaskFromCentreToEdge	217
6.44.3.2	createFromImageFile	217
6.44.3.3	readValuesFromFile	217
6.44.4	Member Data Documentation	218
6.44.4.1	ERR_UNIMPLEMENTED_METHOD	218
6.44.4.2	INTENSITY_MAX	218
6.45	multiscale::analysis::Cluster Class Reference	218
6.45.1	Detailed Description	222
6.45.2	Constructor & Destructor Documentation	222
6.45.2.1	Cluster	222
6.45.2.2	~Cluster	222

6.45.3 Member Function Documentation . . . . .	223
6.45.3.1 addEntity . . . . .	223
6.45.3.2 areValidOriginDependentValues . . . . .	223
6.45.3.3 getEntities . . . . .	223
6.45.3.4 getEntitiesCentrePoints . . . . .	223
6.45.3.5 getEntitiesContourPoints . . . . .	224
6.45.3.6 getEntitiesConvexHull . . . . .	224
6.45.3.7 getMinAreaEnclosingCircleCentre . . . . .	224
6.45.3.8 getMinAreaEnclosingCircleRadius . . . . .	224
6.45.3.9 getMinAreaEnclosingRect . . . . .	224
6.45.3.10 getMinAreaEnclosingTriangle . . . . .	225
6.45.3.11 initialise . . . . .	225
6.45.3.12 isCircularMeasure . . . . .	225
6.45.3.13 isRectangularMeasure . . . . .	226
6.45.3.14 isTriangularMeasure . . . . .	226
6.45.3.15 setOriginDependentMembers . . . . .	226
6.45.3.16 type . . . . .	226
6.45.3.17 updateArea . . . . .	227
6.45.3.18 updateCentrePoint . . . . .	227
6.45.3.19 updateClusterednessDegree . . . . .	227
6.45.3.20 updateDensity . . . . .	227
6.45.3.21 updatePerimeter . . . . .	228
6.45.3.22 updateSpatialEntityShapeArea . . . . .	228
6.45.3.23 validateOriginDependentValues . . . . .	228
6.45.4 Member Data Documentation . . . . .	228
6.45.4.1 entities . . . . .	228
6.45.4.2 ERR_ORIGIN_DEPENDENT_VALUES . . . . .	229
6.45.4.3 ERR_UNDEFINED_SHAPE . . . . .	229
6.46 multiscale::verification::Cluster Class Reference . . . . .	229
6.46.1 Detailed Description . . . . .	231
6.47 multiscale::analysis::ClusterDetector Class Reference . . . . .	232
6.47.1 Detailed Description . . . . .	237
6.47.2 Constructor & Destructor Documentation . . . . .	237
6.47.2.1 ClusterDetector . . . . .	237

6.47.2.2	<code>~ClusterDetector</code>	237
6.47.3	Member Function Documentation	237
6.47.3.1	<code>addEntitiesToClusters</code>	237
6.47.3.2	<code>addPiledUpEntitiesToCollection</code>	237
6.47.3.3	<code>analyseClusters</code>	238
6.47.3.4	<code>analyseClustersOriginDependentValues</code>	238
6.47.3.5	<code>clearPreviousDetectionResults</code>	238
6.47.3.6	<code>computeAveragePileUpDegree</code>	239
6.47.3.7	<code>computeClusterednessIndex</code>	239
6.47.3.8	<code>convertEpsValue</code>	239
6.47.3.9	<code>createDetectorSpecificTrackbars</code>	240
6.47.3.10	<code>detectAndAnalyseClusters</code>	240
6.47.3.11	<code>detectClusters</code>	240
6.47.3.12	<code>detectEntitiesInImage</code>	241
6.47.3.13	<code>flattenEntitiesCollection</code>	241
6.47.3.14	<code>getClusters</code>	241
6.47.3.15	<code>getCollectionOfSpatialEntityPseudo3D</code>	241
6.47.3.16	<code>getDetectorTypeAsString</code>	242
6.47.3.17	<code>getEps</code>	242
6.47.3.18	<code>getMinPoints</code>	242
6.47.3.19	<code>getValidMinPointsValue</code>	242
6.47.3.20	<code>initialiseDetectorSpecificFields</code>	242
6.47.3.21	<code>processImageAndDetect</code>	242
6.47.3.22	<code>setEps</code>	243
6.47.3.23	<code>setMinPoints</code>	243
6.47.3.24	<code>updateClusterOriginDependentValues</code>	243
6.47.4	Member Data Documentation	244
6.47.4.1	<code>clusters</code>	244
6.47.4.2	<code>DETECTOR_TYPE</code>	244
6.47.4.3	<code>eps</code>	244
6.47.4.4	<code>EPS_MAX</code>	244
6.47.4.5	<code>EPS_MIN</code>	244
6.47.4.6	<code>EPS_REAL_MAX</code>	244
6.47.4.7	<code>EPS_REAL_MIN</code>	245

6.47.4.8	maxPileupNumber . . . . .	245
6.47.4.9	MIN_POINTS_MAX . . . . .	245
6.47.4.10	MIN_POINTS_MIN . . . . .	245
6.47.4.11	minPoints . . . . .	245
6.47.4.12	singleEntityIntensity . . . . .	245
6.47.4.13	TRACKBAR_EPS . . . . .	246
6.47.4.14	TRACKBAR_MINPOINTS . . . . .	246
6.48	multiscale::verification::CommandLineModelChecking Class Reference .	246
6.48.1	Detailed Description . . . . .	255
6.48.2	Constructor & Destructor Documentation . . . . .	255
6.48.2.1	CommandLineModelChecking . . . . .	255
6.48.2.2	~CommandLineModelChecking . . . . .	255
6.48.3	Member Function Documentation . . . . .	255
6.48.3.1	areApproximateBayesianModelCheckingArgumentsPresent . . . . .	255
6.48.3.2	areApproximateProbabilisticModelCheckingArgumentsPresent . . . . .	256
6.48.3.3	areBayesianModelCheckingArgumentsPresent . . . . .	256
6.48.3.4	areInvalidExecutionArguments . . . . .	257
6.48.3.5	areInvalidModelCheckingArguments . . . . .	257
6.48.3.6	areInvalidModelCheckingArgumentsPresent . . . . .	257
6.48.3.7	areInvalidModelCheckingTypeSpecificArguments . . . . .	257
6.48.3.8	areModelCheckingTypeSpecificArgumentsPresent . . . . .	258
6.48.3.9	areStatisticalModelCheckingArgumentsPresent . . . . .	258
6.48.3.10	areUnrecognizedArgumentsPresent . . . . .	259
6.48.3.11	areValidArguments . . . . .	259
6.48.3.12	areValidArgumentsConsideringConfiguration . . . . .	259
6.48.3.13	execute . . . . .	260
6.48.3.14	handleHelpRequest . . . . .	260
6.48.3.15	initialise . . . . .	260
6.48.3.16	initialiseAllowedArgumentsConfiguration . . . . .	260
6.48.3.17	initialiseApproximateBayesianModelChecker . . . . .	261
6.48.3.18	initialiseApproximateBayesianModelCheckerArgumentsConfiguration . . . . .	261
6.48.3.19	initialiseApproximateProbabilisticModelChecker . . . . .	261

6.48.3.20 initialiseApproximateProbabilisticModelChecker-ArgumentsConfiguration . . . . .	262
6.48.3.21 initialiseBayesianModelChecker . . . . .	262
6.48.3.22 initialiseBayesianModelCheckerArgumentsConfiguration	262
6.48.3.23 initialiseClassMembers . . . . .	263
6.48.3.24 initialiseModelChecker . . . . .	263
6.48.3.25 initialiseModelCheckerTypeDependentClassMembers	263
6.48.3.26 initialiseModelCheckerTypeSpecificArguments-Configuration . . . . .	263
6.48.3.27 initialiseModelCheckingManager . . . . .	264
6.48.3.28 initialiseOptionalArgumentsConfiguration . . . . .	264
6.48.3.29 initialiseOptionalArgumentsDependentClassMembers	264
6.48.3.30 initialiseProbabilisticBlackBoxModelChecker . . . . .	264
6.48.3.31 initialiseRequiredArgumentsConfiguration . . . . .	265
6.48.3.32 initialiseRequiredArgumentsDependentClassMembers	265
6.48.3.33 initialiseStatisticalModelChecker . . . . .	265
6.48.3.34 initialiseStatisticalModelCheckerArgumentsConfiguration	266
6.48.3.35 isHelpArgumentPresent . . . . .	266
6.48.3.36 parseAndStoreArgumentsValues . . . . .	266
6.48.3.37 printHelpClosingMessage . . . . .	266
6.48.3.38 printHelpContentsMessage . . . . .	267
6.48.3.39 printHelpIntroMessage . . . . .	267
6.48.3.40 printHelpMessage . . . . .	267
6.48.3.41 printModelCheckingInitialisationMessage . . . . .	267
6.48.3.42 removeApproximateBayesianModelCheckingArguments	268
6.48.3.43 removeApproximateProbabilisticModelChecking-Arguments . . . . .	268
6.48.3.44 removeBayesianModelCheckingArguments . . . . .	268
6.48.3.45 removeModelCheckingTypeSpecificArguments . . . . .	269
6.48.3.46 removeOptionalArguments . . . . .	269
6.48.3.47 removeRequiredArguments . . . . .	270
6.48.3.48 removeStatisticalModelCheckingArguments . . . . .	270
6.48.4 Member Data Documentation . . . . .	270
6.48.4.1 allowedArguments . . . . .	270

6.48.4.2 ARG_APPROXIMATE_BAYESIAN_ALPHA_DESCRIPTION . . . . .	271
6.48.4.3 ARG_APPROXIMATE_BAYESIAN_ALPHA_NAME_LONG . . . . .	271
6.48.4.4 ARG_APPROXIMATE_BAYESIAN_BETA_DESCRIPTION . . . . .	271
6.48.4.5 ARG_APPROXIMATE_BAYESIAN_BETA_NAME_LONG . . . . .	271
6.48.4.6 ARG_BAYES_FACTOR_THRESHOLD_DESCRIPTION . . . . .	271
6.48.4.7 ARG_BAYES_FACTOR_THRESHOLD_NAME_LONG . . . . .	271
6.48.4.8 ARG_BAYESIAN_ALPHA_DESCRIPTION . . . . .	272
6.48.4.9 ARG_BAYESIAN_ALPHA_NAME_LONG . . . . .	272
6.48.4.10 ARG_BAYESIAN_BETA_DESCRIPTION . . . . .	272
6.48.4.11 ARG_BAYESIAN_BETA_NAME_LONG . . . . .	272
6.48.4.12 ARG_DELTA_DESCRIPTION . . . . .	272
6.48.4.13 ARG_DELTA_NAME_LONG . . . . .	273
6.48.4.14 ARG_EPSILON_DESCRIPTION . . . . .	273
6.48.4.15 ARG_EPSILON_NAME_LONG . . . . .	273
6.48.4.16 ARG_EXTRA_EVALUATION_PROGRAM_DESCRIPTION . . . . .	273
6.48.4.17 ARG_EXTRA_EVALUATION_PROGRAM_NAME_BOTH . . . . .	273
6.48.4.18 ARG_EXTRA_EVALUATION_PROGRAM_NAME_LONG . . . . .	274
6.48.4.19 ARG_EXTRA_EVALUATION_TIME_DESCRIPTION . . . . .	274
6.48.4.20 ARG_EXTRA_EVALUATION_TIME_NAME_BOTH . . . . .	274
6.48.4.21 ARG_EXTRA_EVALUATION_TIME_NAME_LONG . . . . .	274
6.48.4.22 ARG_HELP_DESCRIPTION . . . . .	274
6.48.4.23 ARG_HELP_NAME_BOTH . . . . .	274
6.48.4.24 ARG_HELP_NAME_LONG . . . . .	275
6.48.4.25 ARG_LOGIC_QUERIES_DESCRIPTION . . . . .	275
6.48.4.26 ARG_LOGIC_QUERIES_NAME_BOTH . . . . .	275
6.48.4.27 ARG_LOGIC_QUERIES_NAME_LONG . . . . .	275
6.48.4.28 ARG_MODEL_CHECKER_TYPE_DESCRIPTION . . . . .	275
6.48.4.29 ARG_MODEL_CHECKER_TYPE_NAME_BOTH . . . . .	275

6.48.4.30 ARG_MODEL_CHECKER_TYPE_NAME_LONG . . . . .	276
6.48.4.31 ARG_MULTISCALE_ARCHITECTURE_GRAPH_DESCRIPTION . . . . .	276
6.48.4.32 ARG_MULTISCALE_ARCHITECTURE_GRAPH_NAME_BOTH . . . . .	276
6.48.4.33 ARG_MULTISCALE_ARCHITECTURE_GRAPH_NAME_LONG . . . . .	276
6.48.4.34 ARG_SPATIAL_TEMPORAL_TRACES_DESCRIPTION . . . . .	276
6.48.4.35 ARG_SPATIAL_TEMPORAL_TRACES_NAME_BOTH . . . . .	277
6.48.4.36 ARG_SPATIAL_TEMPORAL_TRACES_NAME_LONG . . . . .	277
6.48.4.37 ARG_TYPE_I_ERROR_DESCRIPTION . . . . .	277
6.48.4.38 ARG_TYPE_I_ERROR_NAME_LONG . . . . .	277
6.48.4.39 ARG_TYPE_II_ERROR_DESCRIPTION . . . . .	277
6.48.4.40 ARG_TYPE_II_ERROR_NAME_LONG . . . . .	277
6.48.4.41 ARG_VARIANCE_THRESHOLD_DESCRIPTION . . . . .	278
6.48.4.42 ARG_VARIANCE_THRESHOLD_NAME_LONG . . . . .	278
6.48.4.43 ARG_VERBOSE_DESCRIPTION . . . . .	278
6.48.4.44 ARG_VERBOSE_NAME_BOTH . . . . .	278
6.48.4.45 ARG_VERBOSE_NAME_LONG . . . . .	278
6.48.4.46 CONFIG_CAPTION_ALLOWED_ARGUMENTS . . . . .	279
6.48.4.47 CONFIG_CAPTION_APPROXIMATE_BAYESIAN_MODEL_CHECKER_ARGUMENTS . . . . .	279
6.48.4.48 CONFIG_CAPTION_APPROXIMATE_PROBABILISTIC_MODEL_CHECKER_ARGUMENTS . . . . .	279
6.48.4.49 CONFIG_CAPTION_BAYESIAN_MODEL_CHECKER_ARGUMENTS . . . . .	279
6.48.4.50 CONFIG_CAPTION_MODEL_CHECKER_TYPE_SPECIFIC_ARGUMENTS . . . . .	279
6.48.4.51 CONFIG_CAPTION_OPTIONAL_ARGUMENTS . . . . .	279
6.48.4.52 CONFIG_CAPTION_PROBABILISTIC_BLACK_BOX_MODEL_CHECKER_ARGUMENTS . . . . .	280
6.48.4.53 CONFIG_CAPTION_REQUIRED_ARGUMENTS . . . . .	280
6.48.4.54 CONFIG_CAPTION_STATISTICAL_MODEL_CHECKER_ARGUMENTS . . . . .	280
6.48.4.55 ERR_INVALID_COMMAND_LINE_ARGUMENTS . . . . .	280

6.48.4.56	ERR_INVALID_MODEL_CHECKING_ARGUMENTS	. . . . .	280
6.48.4.57	ERR_INVALID_MODEL_CHECKING_TYPE	. . . . .	280
6.48.4.58	extraEvaluationProgramPath	. . . . .	281
6.48.4.59	extraEvaluationTime	. . . . .	281
6.48.4.60	HELP_AUTHOR_LABEL	. . . . .	281
6.48.4.61	HELP_AUTHOR_MSG	. . . . .	281
6.48.4.62	HELP_COPYRIGHT_LABEL	. . . . .	281
6.48.4.63	HELP_COPYRIGHT_MSG	. . . . .	281
6.48.4.64	HELP_DESCRIPTION_LABEL	. . . . .	282
6.48.4.65	HELP_DESCRIPTION_MSG	. . . . .	282
6.48.4.66	HELP_NAME_LABEL	. . . . .	282
6.48.4.67	HELP_NAME_MSG	. . . . .	282
6.48.4.68	HELP_REPORTING_BUGS_LABEL	. . . . .	282
6.48.4.69	HELP_REPORTING_BUGS_MSG	. . . . .	283
6.48.4.70	HELP_USAGE_LABEL	. . . . .	283
6.48.4.71	HELP_USAGE_MSG	. . . . .	283
6.48.4.72	logicQueriesFilepath	. . . . .	283
6.48.4.73	MODEL_CHECKER_APPROXIMATE_BAYESIAN_- NAME	. . . . .	283
6.48.4.74	MODEL_CHECKER_APPROXIMATE_BAYESIAN_- PARAMETERS_BEGIN	. . . . .	283
6.48.4.75	MODEL_CHECKER_APPROXIMATE_BAYESIAN_- PARAMETERS_END	. . . . .	284
6.48.4.76	MODEL_CHECKER_APPROXIMATE_BAYESIAN_- PARAMETERS_MIDDLE1	. . . . .	284
6.48.4.77	MODEL_CHECKER_APPROXIMATE_BAYESIAN_- PARAMETERS_MIDDLE2	. . . . .	284
6.48.4.78	MODEL_CHECKER_APPROXIMATE_PROBABILI- STIC_NAME	. . . . .	284
6.48.4.79	MODEL_CHECKER_APPROXIMATE_PROBABILI- STIC_PARAMETERS_BEGIN	. . . . .	284
6.48.4.80	MODEL_CHECKER_APPROXIMATE_PROBABILI- STIC_PARAMETERS_END	. . . . .	284
6.48.4.81	MODEL_CHECKER_APPROXIMATE_PROBABILI- STIC_PARAMETERS_MIDDLE	. . . . .	285
6.48.4.82	MODEL_CHECKER_BAYESIAN_NAME	. . . . .	285

6.48.4.83 MODEL_CHECKER_BAYESIAN_PARAMETERS_- BEGIN . . . . .	285
6.48.4.84 MODEL_CHECKER_BAYESIAN_PARAMETERS_- END . . . . .	285
6.48.4.85 MODEL_CHECKER_BAYESIAN_PARAMETERS_- MIDDLE1 . . . . .	285
6.48.4.86 MODEL_CHECKER_BAYESIAN_PARAMETERS_- MIDDLE2 . . . . .	285
6.48.4.87 MODEL_CHECKER_PROBABILISTIC_BLACK_BO- X_NAME . . . . .	286
6.48.4.88 MODEL_CHECKER_PROBABILISTIC_BLACK_BO- X_PARAMETERS . . . . .	286
6.48.4.89 MODEL_CHECKER_STATISTICAL_NAME . . . . .	286
6.48.4.90 MODEL_CHECKER_STATISTICAL_PARAMETER- S_BEGIN . . . . .	286
6.48.4.91 MODEL_CHECKER_STATISTICAL_PARAMETER- S_END . . . . .	286
6.48.4.92 MODEL_CHECKER_STATISTICAL_PARAMETER- S_MIDDLE . . . . .	286
6.48.4.93 MODEL_CHECKER_TYPE_APPROXIMATE_BAY- ESIAN . . . . .	287
6.48.4.94 MODEL_CHECKER_TYPE_APPROXIMATE_PRO- BABILISTIC . . . . .	287
6.48.4.95 MODEL_CHECKER_TYPE_BAYESIAN . . . . .	287
6.48.4.96 MODEL_CHECKER_TYPE_PROBABILISTIC_BLA- CK_BOX . . . . .	287
6.48.4.97 MODEL_CHECKER_TYPE_STATISTICAL . . . . .	287
6.48.4.98 modelCheckerFactory . . . . .	288
6.48.4.99 modelCheckerParameters . . . . .	288
6.48.4.100modelCheckerType . . . . .	288
6.48.4.101modelCheckerTypeName . . . . .	288
6.48.4.102modelCheckerTypeSpecificArguments . . . . .	288
6.48.4.103modelCheckingManager . . . . .	289
6.48.4.104MSG_MODEL_CHECKING_HELP_REQUESTED . .	289
6.48.4.105multiscaleArchitectureGraphFilepath . . .	289
6.48.4.106optionalArguments . . . . .	289
6.48.4.107requiredArguments . . . . .	289
6.48.4.108shouldVerboseDetailedResults . . . . .	290

6.48.4.109tracesFolderPath . . . . .	290
6.48.4.110variablesMap . . . . .	290
6.49 multiscale::verification::ComparatorAttribute Class Reference . . . . .	290
6.49.1 Detailed Description . . . . .	291
6.49.2 Member Data Documentation . . . . .	291
6.49.2.1 comparatorType . . . . .	291
6.50 multiscale::verification::ComparatorEvaluator Class Reference . . . . .	291
6.50.1 Detailed Description . . . . .	292
6.50.2 Member Function Documentation . . . . .	292
6.50.2.1 evaluate . . . . .	292
6.50.2.2 evaluate . . . . .	292
6.51 multiscale::verification::ComparatorGrammar< Iterator > Class - Template Reference . . . . .	293
6.51.1 Detailed Description . . . . .	295
6.51.2 Constructor & Destructor Documentation . . . . .	296
6.51.2.1 ComparatorGrammar . . . . .	296
6.51.3 Member Function Documentation . . . . .	296
6.51.3.1 assignNamesToRules . . . . .	296
6.51.3.2 initialise . . . . .	296
6.51.3.3 initialiseDebugSupport . . . . .	296
6.51.3.4 initialiseGrammar . . . . .	297
6.51.3.5 initialiseRulesDebugging . . . . .	297
6.51.4 Member Data Documentation . . . . .	297
6.51.4.1 comparatorRule . . . . .	297
6.51.4.2 comparatorTypeParser . . . . .	297
6.52 multiscale::verification::ComparatorNonEqualTypeParser Struct - Reference . . . . .	297
6.52.1 Detailed Description . . . . .	298
6.52.2 Constructor & Destructor Documentation . . . . .	298
6.52.2.1 ComparatorNonEqualTypeParser . . . . .	298
6.53 multiscale::verification::ComparatorTypeParser Struct Reference . . . . .	298
6.53.1 Detailed Description . . . . .	298
6.53.2 Constructor & Destructor Documentation . . . . .	298
6.53.2.1 ComparatorTypeParser . . . . .	299

6.54 multiscaletest::CompleteTraceTest Class Reference . . . . .	299
6.54.1 Detailed Description . . . . .	302
6.54.2 Member Function Documentation . . . . .	302
6.54.2.1 InitialiseTrace . . . . .	302
6.54.3 Member Data Documentation . . . . .	302
6.54.3.1 clustersAreaMaxValue . . . . .	302
6.54.3.2 clustersAreaMinValue . . . . .	302
6.55 multiscale::ConsolePrinter Class Reference . . . . .	303
6.55.1 Detailed Description . . . . .	306
6.55.2 Member Function Documentation . . . . .	306
6.55.2.1 getUnixColourCode . . . . .	306
6.55.2.2 isStdOutTerminalWhichSupportsColour . . . . .	306
6.55.2.3 printColouredMessage . . . . .	307
6.55.2.4 printColouredMessageWithColouredTag . . . . .	307
6.55.2.5 printEmptyLine . . . . .	308
6.55.2.6 printMessage . . . . .	308
6.55.2.7 printMessageUsingColour . . . . .	308
6.55.2.8 printMessageWithColouredTag . . . . .	309
6.55.2.9 printNewLine . . . . .	309
6.55.2.10 printNonColouredMessage . . . . .	310
6.55.2.11 printWarningMessage . . . . .	310
6.55.2.12 terminalSupportsColour . . . . .	310
6.55.2.13 terminalSupportsColour . . . . .	311
6.55.2.14 unixColourCodeToString . . . . .	311
6.55.3 Member Data Documentation . . . . .	311
6.55.3.1 CSI_COLOUR_CODE_END_TAG . . . . .	311
6.55.3.2 CSI_COLOUR_START_VALUE . . . . .	311
6.55.3.3 CSI_RESET_CODE . . . . .	312
6.55.3.4 CSI_SEPARATOR . . . . .	312
6.55.3.5 CSI_START_TAG . . . . .	312
6.55.3.6 ERR_INVALID_COLOUR_CODE . . . . .	312
6.55.3.7 SEPARATOR . . . . .	312
6.55.3.8 TERM_ENV_VARIABLE . . . . .	312
6.55.3.9 WARNING_TAG . . . . .	312

6.56 multiscale::verification::ConstraintAttribute Class Reference . . . . .	313
6.56.1 Detailed Description . . . . .	315
6.56.2 Member Data Documentation . . . . .	315
6.56.2.1 firstConstraint . . . . .	315
6.56.2.2 nextConstraints . . . . .	315
6.57 multiscale::verification::ConstraintVisitor Class Reference . . . . .	315
6.57.1 Detailed Description . . . . .	318
6.57.2 Constructor & Destructor Documentation . . . . .	319
6.57.2.1 ConstraintVisitor . . . . .	319
6.57.3 Member Function Documentation . . . . .	319
6.57.3.1 evaluate . . . . .	319
6.57.3.2 evaluate . . . . .	319
6.57.3.3 evaluateFilterNumericMeasure . . . . .	320
6.57.3.4 evaluateNextConstraints . . . . .	320
6.57.3.5 evaluateNumericMeasure . . . . .	320
6.57.3.6 evaluateScaleAndSubsystemConstraint . . . . .	321
6.57.3.7 evaluateSpatialMeasureConstraint . . . . .	321
6.57.3.8 evaluateUnaryScaleAndSubsystemConstraint . . . . .	322
6.57.3.9 evaluateUnarySpatialConstraint . . . . .	322
6.57.3.10 filterSpatialEntitiesWrtScaleAndSubsystem . . . . .	323
6.57.3.11 filterSpatialEntitiesWrtScaleAndSubsystem . . . . .	323
6.57.3.12 filterSpatialEntitiesWrtScaleAndSubsystemConsidering-EqualComparator . . . . .	324
6.57.3.13 filterSpatialEntitiesWrtScaleAndSubsystemConsidering-NonEqualComparator . . . . .	324
6.57.3.14 filterSpatialEntitiesWrtSpatialMeasure . . . . .	325
6.57.3.15 operator() . . . . .	326
6.57.3.16 operator() . . . . .	326
6.57.3.17 operator() . . . . .	326
6.57.3.18 operator() . . . . .	326
6.57.3.19 operator() . . . . .	327
6.57.3.20 operator() . . . . .	327
6.57.3.21 operator() . . . . .	327
6.57.3.22 operator() . . . . .	328

6.57.3.23 operator() . . . . .	328
6.57.3.24 operator() . . . . .	328
6.57.4 Member Data Documentation . . . . .	329
6.57.4.1 constraintTimePoint . . . . .	329
6.57.4.2 initialTimePoint . . . . .	329
6.57.4.3 multiscaleArchitectureGraph . . . . .	329
6.58 multiscale::analysis::DBSCAN< PointType > Class Template Reference	329
6.58.1 Detailed Description . . . . .	333
6.58.2 Constructor & Destructor Documentation . . . . .	333
6.58.2.1 DBSCAN . . . . .	333
6.58.2.2 ~DBSCAN . . . . .	333
6.58.3 Member Function Documentation . . . . .	333
6.58.3.1 addUnclassifiedNodesToSeedsList . . . . .	333
6.58.3.2 allocateDistancesMatrix . . . . .	334
6.58.3.3 allocateNeighboursIndicesMatrix . . . . .	334
6.58.3.4 assignBorderNodesToClusters . . . . .	334
6.58.3.5 constructDistancesMatrix . . . . .	335
6.58.3.6 constructNeighboursIndicesMatrix . . . . .	335
6.58.3.7 expandCoreCluster . . . . .	336
6.58.3.8 findClosestCoreDataPoint . . . . .	336
6.58.3.9 labelUnclassifiedAndNoiseAsBorder . . . . .	337
6.58.3.10 processSeeds . . . . .	337
6.58.3.11 retrieveNeighbours . . . . .	338
6.58.3.12 run . . . . .	338
6.58.3.13 runAlgorithm . . . . .	339
6.58.4 Member Data Documentation . . . . .	339
6.58.4.1 CLUSTERING_BORDER . . . . .	339
6.58.4.2 CLUSTERING_NOISE . . . . .	339
6.58.4.3 CLUSTERING_UNCLASSIFIED . . . . .	339
6.58.4.4 distancesMatrix . . . . .	340
6.58.4.5 eps . . . . .	340
6.58.4.6 minPoints . . . . .	340
6.58.4.7 neighboursIndicesMatrix . . . . .	340
6.58.4.8 nrOfDataPoints . . . . .	341

6.59 multiscale::analysis::Detector Class Reference . . . . .	341
6.59.1 Detailed Description . . . . .	347
6.59.2 Constructor & Destructor Documentation . . . . .	348
6.59.2.1 Detector . . . . .	348
6.59.2.2 ~Detector . . . . .	348
6.59.3 Member Function Documentation . . . . .	348
6.59.3.1 addAverageMeasuresToPropertyTree . . . . .	348
6.59.3.2 addNumericStateVariableToPropertyTree . . . . .	348
6.59.3.3 addSpatialEntitiesToPropertyTree . . . . .	349
6.59.3.4 addSpatialEntityPropertiesToTree . . . . .	349
6.59.3.5 addSpatialEntityTypeToPropertyTree . . . . .	350
6.59.3.6 clearPreviousDetectionResults . . . . .	350
6.59.3.7 computeDistanceFromOrigin . . . . .	350
6.59.3.8 computeDistanceFromOrigin . . . . .	350
6.59.3.9 computePolygonAngle . . . . .	351
6.59.3.10 computePolygonAngle . . . . .	351
6.59.3.11 computePolygonAngle . . . . .	351
6.59.3.12 constructSpatialEntityPropertyTree . . . . .	352
6.59.3.13 createDetectorSpecificTrackbars . . . . .	352
6.59.3.14 createTrackbars . . . . .	352
6.59.3.15 createTrackbarsWindow . . . . .	352
6.59.3.16 detect . . . . .	353
6.59.3.17 detect . . . . .	353
6.59.3.18 detectInDebugMode . . . . .	353
6.59.3.19 detectInReleaseMode . . . . .	353
6.59.3.20 displayImage . . . . .	353
6.59.3.21 displayResultsInWindow . . . . .	354
6.59.3.22 getCollectionOfSpatialEntityPseudo3D . . . . .	354
6.59.3.23 getDetectorTypeAsString . . . . .	354
6.59.3.24 initialise . . . . .	354
6.59.3.25 initialiseDetectorSpecificFields . . . . .	355
6.59.3.26 initialiseDetectorSpecificFieldsIfNotSet . . . . .	355
6.59.3.27 initialiseDetectorSpecificImageDependentFields . . . . .	355
6.59.3.28 initialiseImage . . . . .	355

6.59.3.29 initialiseImageDependentFields . . . . .	356
6.59.3.30 initialiseImageOrigin . . . . .	356
6.59.3.31 initialiseScaledImage . . . . .	356
6.59.3.32 isValidInputImage . . . . .	356
6.59.3.33 offsetPolygons . . . . .	356
6.59.3.34 outputAveragedMeasuresToCsvFile . . . . .	357
6.59.3.35 outputResults . . . . .	357
6.59.3.36 outputResultsToCsvFile . . . . .	357
6.59.3.37 outputResultsToCsvFile . . . . .	357
6.59.3.38 outputResultsToFile . . . . .	358
6.59.3.39 outputResultsToImage . . . . .	358
6.59.3.40 outputResultsToXMLFile . . . . .	358
6.59.3.41 outputResultsToXMLFile . . . . .	358
6.59.3.42 outputSpatialEntitiesToCsvFile . . . . .	359
6.59.3.43 printOutputErrorMessage . . . . .	359
6.59.3.44 processImageAndDetect . . . . .	359
6.59.3.45 processPressedKeyRequest . . . . .	359
6.59.3.46 setDetectorSpecificFieldsInitialisationFlag . . . . .	360
6.59.3.47 storeOutputImageOnDisk . . . . .	360
6.59.4 Member Data Documentation . . . . .	360
6.59.4.1 avgClusterednessDegree . . . . .	360
6.59.4.2 avgDensity . . . . .	360
6.59.4.3 CSV_EXTENSION . . . . .	361
6.59.4.4 detectMethodCalled . . . . .	361
6.59.4.5 detectorSpecificFieldsInitialised . . . . .	361
6.59.4.6 ERR_INVALID_IMAGE . . . . .	361
6.59.4.7 ERR_OUTPUT_FILE . . . . .	361
6.59.4.8 ERR_OUTPUT_WITHOUT_DETECT . . . . .	361
6.59.4.9 image . . . . .	362
6.59.4.10 IMG_EXTENSION . . . . .	362
6.59.4.11 INTENSITY_MAX . . . . .	362
6.59.4.12 isDebugMode . . . . .	362
6.59.4.13 KEY_ESC . . . . .	362
6.59.4.14 KEY_SAVE . . . . .	363

6.59.4.15 LABEL_ATTRIBUTE . . . . .	363
6.59.4.16 LABEL_AVG_CLUSTEREDNESS . . . . .	363
6.59.4.17 LABEL_AVG_DENSITY . . . . .	363
6.59.4.18 LABEL_COMMENT . . . . .	363
6.59.4.19 LABEL_COMMENT_CONTENTS . . . . .	363
6.59.4.20 LABEL_EXPERIMENT_TIMEPOINT_NUMERIC_STATE_VARIABLE . . . . .	363
6.59.4.21 LABEL_EXPERIMENT_TIMEPOINT_NUMERIC_STATE_VARIABLE_NAME . . . . .	364
6.59.4.22 LABEL_EXPERIMENT_TIMEPOINT_NUMERIC_STATE_VARIABLE_VALUE . . . . .	364
6.59.4.23 LABEL_EXPERIMENT_TIMEPOINT_SPATIAL_ENTITY . . . . .	364
6.59.4.24 LABEL_SPATIAL_ENTITY_ANGLE . . . . .	364
6.59.4.25 LABEL_SPATIAL_ENTITY_AREA . . . . .	364
6.59.4.26 LABEL_SPATIAL_ENTITY_CENTROID_X . . . . .	364
6.59.4.27 LABEL_SPATIAL_ENTITY_CENTROID_Y . . . . .	365
6.59.4.28 LABEL_SPATIAL_ENTITY_CIRCLE_MEASURE . . . . .	365
6.59.4.29 LABEL_SPATIAL_ENTITY_CLUSTEREDNESS . . . . .	365
6.59.4.30 LABEL_SPATIAL_ENTITY_DENSITY . . . . .	365
6.59.4.31 LABEL_SPATIAL_ENTITY_DISTANCE_FROM_ORIGIN . . . . .	365
6.59.4.32 LABEL_SPATIAL_ENTITY_PERIMETER . . . . .	365
6.59.4.33 LABEL_SPATIAL_ENTITY_RECTANGLE_MEASURE . . . . .	365
6.59.4.34 LABEL_SPATIAL_ENTITY_SPATIAL_TYPE . . . . .	366
6.59.4.35 LABEL_SPATIAL_ENTITY_TRIANGLE_MEASURE . . . . .	366
6.59.4.36 origin . . . . .	366
6.59.4.37 OUTPUT_CLUSTEREDNESS . . . . .	366
6.59.4.38 OUTPUT_DENSITY . . . . .	366
6.59.4.39 outputfilepath . . . . .	366
6.59.4.40 outputImage . . . . .	367
6.59.4.41 WIN_OUTPUT_IMAGE . . . . .	367
6.59.4.42 XML_EXTENSION . . . . .	367
6.60 multiscale::Distribution Class Reference . . . . .	367
6.60.1 Detailed Description . . . . .	370

6.60.2 Member Function Documentation . . . . .	370
6.60.2.1 validateProbability . . . . .	370
6.60.3 Member Data Documentation . . . . .	370
6.60.3.1 ERR_PROBABILITY_VALUE_BEGIN . . . . .	370
6.60.3.2 ERR_PROBABILITY_VALUE_END . . . . .	370
6.61 multiscale::DivisionOperation Class Reference . . . . .	370
6.61.1 Detailed Description . . . . .	371
6.61.2 Member Function Documentation . . . . .	371
6.61.2.1 operator() . . . . .	371
6.62 multiscaletest::EmptyTraceTest Class Reference . . . . .	371
6.62.1 Detailed Description . . . . .	374
6.62.2 Member Function Documentation . . . . .	374
6.62.2.1 InitialiseTrace . . . . .	374
6.63 multiscale::analysis::Entity Class Reference . . . . .	374
6.63.1 Detailed Description . . . . .	377
6.63.2 Constructor & Destructor Documentation . . . . .	377
6.63.2.1 Entity . . . . .	377
6.63.2.2 Entity . . . . .	377
6.63.2.3 ~Entity . . . . .	377
6.63.3 Member Function Documentation . . . . .	377
6.63.3.1 areValid . . . . .	377
6.63.3.2 distanceTo . . . . .	378
6.63.3.3 getArea . . . . .	378
6.63.3.4 getCentre . . . . .	378
6.63.3.5 getContourPoints . . . . .	378
6.63.3.6 getPerimeter . . . . .	378
6.63.3.7 getPileUpDegree . . . . .	378
6.63.3.8 toString . . . . .	379
6.63.3.9 validateInputValues . . . . .	379
6.63.4 Member Data Documentation . . . . .	379
6.63.4.1 area . . . . .	379
6.63.4.2 centre . . . . .	379
6.63.4.3 contourPoints . . . . .	379
6.63.4.4 ERR_DISTANCE . . . . .	380

6.63.4.5	ERR_INPUT . . . . .	380
6.63.4.6	OUTPUT_SEPARATOR . . . . .	380
6.63.4.7	perimeter . . . . .	380
6.63.4.8	pileUpDegree . . . . .	380
6.64	multiscale::verification::EquivalenceConstraintAttribute Class Reference	380
6.64.1	Detailed Description . . . . .	381
6.64.2	Member Data Documentation . . . . .	381
6.64.2.1	constraint . . . . .	381
6.65	multiscale::verification::EquivalenceLogicPropertyAttribute Class Reference . . . . .	381
6.65.1	Detailed Description . . . . .	381
6.65.2	Member Data Documentation . . . . .	382
6.65.2.1	logicProperty . . . . .	382
6.66	EuclideanDataPoint Class Reference . . . . .	382
6.66.1	Detailed Description . . . . .	382
6.66.2	Constructor & Destructor Documentation . . . . .	382
6.66.2.1	EuclideanDataPoint . . . . .	382
6.66.2.2	EuclideanDataPoint . . . . .	382
6.66.2.3	~EuclideanDataPoint . . . . .	383
6.66.3	Member Function Documentation . . . . .	383
6.66.3.1	distanceTo . . . . .	383
6.66.4	Member Data Documentation . . . . .	383
6.66.4.1	x . . . . .	383
6.66.4.2	y . . . . .	383
6.67	multiscale::ExceptionHandler Class Reference . . . . .	383
6.67.1	Detailed Description . . . . .	384
6.67.2	Member Function Documentation . . . . .	384
6.67.2.1	printDetailedErrorMessage . . . . .	384
6.67.2.2	printHelpMessage . . . . .	384
6.67.2.3	printRawErrorMessage . . . . .	384
6.68	multiscale::FileOpenException Class Reference . . . . .	385
6.68.1	Detailed Description . . . . .	388
6.68.2	Constructor & Destructor Documentation . . . . .	388
6.68.2.1	FileOpenException . . . . .	388

6.68.2.2	FileOpenException . . . . .	388
6.68.2.3	FileOpenException . . . . .	388
6.69	multiscale::Filesystem Class Reference . . . . .	388
6.69.1	Detailed Description . . . . .	390
6.69.2	Member Function Documentation . . . . .	390
6.69.2.1	getFilesInFolder . . . . .	390
6.69.2.2	isValidFilePath . . . . .	391
6.69.2.3	isValidFolderPath . . . . .	391
6.69.2.4	isValidFilePath . . . . .	391
6.69.2.5	isValidFolderPath . . . . .	391
6.69.2.6	nativeFormatFilePath . . . . .	392
6.69.3	Member Data Documentation . . . . .	392
6.69.3.1	ERR_INVALID_PATH_BEGIN . . . . .	392
6.69.3.2	ERR_INVALID_PATH_END . . . . .	392
6.70	multiscale::verification::FilterNumericMeasureAttribute Class Reference . . . . .	393
6.70.1	Detailed Description . . . . .	393
6.70.2	Member Data Documentation . . . . .	393
6.70.2.1	filterNumericMeasure . . . . .	393
6.71	multiscale::verification::FilterNumericVisitor Class Reference . . . . .	393
6.71.1	Detailed Description . . . . .	395
6.71.2	Constructor & Destructor Documentation . . . . .	395
6.71.2.1	FilterNumericVisitor . . . . .	395
6.71.3	Member Function Documentation . . . . .	395
6.71.3.1	evaluate . . . . .	395
6.71.3.2	evaluate . . . . .	396
6.71.3.3	operator() . . . . .	396
6.71.3.4	operator() . . . . .	396
6.71.3.5	operator() . . . . .	397
6.71.3.6	operator() . . . . .	397
6.71.3.7	operator() . . . . .	397
6.71.4	Member Data Documentation . . . . .	398
6.71.4.1	multiscaleArchitectureGraph . . . . .	398
6.71.4.2	spatialEntity . . . . .	398
6.71.4.3	timePoint . . . . .	398

6.72 multiscale::verification::FilterSubsetAttribute Class Reference . . . . .	398
6.72.1 Detailed Description . . . . .	399
6.72.2 Member Data Documentation . . . . .	399
6.72.2.1 constraint . . . . .	399
6.72.2.2 subsetSpecific . . . . .	400
6.73 multiscale::verification::FutureLogicPropertyAttribute Class Reference . . . . .	400
6.73.1 Detailed Description . . . . .	400
6.73.2 Member Data Documentation . . . . .	400
6.73.2.1 endTimepoint . . . . .	400
6.73.2.2 logicProperty . . . . .	400
6.73.2.3 startTimepoint . . . . .	401
6.74 multiscale::Geometry2D Class Reference . . . . .	401
6.74.1 Detailed Description . . . . .	404
6.74.2 Member Function Documentation . . . . .	405
6.74.2.1 angleBtwPoints . . . . .	405
6.74.2.2 angleOfLineWrtOxAxis . . . . .	405
6.74.2.3 areaOfTriangle . . . . .	405
6.74.2.4 areCollinear . . . . .	406
6.74.2.5 areEqualPoints . . . . .	406
6.74.2.6 areIdenticalLines . . . . .	406
6.74.2.7 areIdenticalLines . . . . .	407
6.74.2.8 areOnTheSameSideOfLine . . . . .	407
6.74.2.9 centroid . . . . .	408
6.74.2.10 computeConvexHull . . . . .	408
6.74.2.11 convertPoints . . . . .	408
6.74.2.12 distanceBtwPoints . . . . .	409
6.74.2.13 distanceBtwPoints . . . . .	409
6.74.2.14 distanceFromPointToLine . . . . .	409
6.74.2.15 findPointsOnEdge . . . . .	410
6.74.2.16 inverseTranslate . . . . .	410
6.74.2.17 isAngleBetween . . . . .	411
6.74.2.18 isAngleBetweenNonReflex . . . . .	411
6.74.2.19 isBetweenCoordinates . . . . .	411
6.74.2.20 isConvexPolygon . . . . .	412

6.74.2.21	isOppositeAngleBetween . . . . .	412
6.74.2.22	isOppositeAngleBetweenNonReflex . . . . .	412
6.74.2.23	isPointInsidePolygon . . . . .	413
6.74.2.24	isPointOnEdge . . . . .	413
6.74.2.25	isPointOnLineSegment . . . . .	413
6.74.2.26	isToTheLeftOfLine . . . . .	414
6.74.2.27	isToTheRightOfLine . . . . .	414
6.74.2.28	lineCircleIntersection . . . . .	415
6.74.2.29	lineCircleOneIntersectionPoint . . . . .	415
6.74.2.30	lineCircleTwoIntersectionPoints . . . . .	416
6.74.2.31	lineEquationDeterminedByPoints . . . . .	416
6.74.2.32	lineIntersection . . . . .	417
6.74.2.33	lineIntersection . . . . .	417
6.74.2.34	lineIntersection . . . . .	418
6.74.2.35	lineSegmentCircleIntersection . . . . .	418
6.74.2.36	lineSegmentIntersection . . . . .	419
6.74.2.37	middlePoint . . . . .	419
6.74.2.38	minimumDistancePointIndex . . . . .	420
6.74.2.39	minimumDistancePointIndex . . . . .	420
6.74.2.40	oppositeAngle . . . . .	420
6.74.2.41	orthogonalLineToAnotherLineEdgePoints . . . . .	421
6.74.2.42	sideOfLine . . . . .	421
6.74.2.43	slopeOfLine . . . . .	422
6.74.2.44	tangentsFromPointToPolygon . . . . .	422
6.74.2.45	translate . . . . .	422
6.74.3	Member Data Documentation . . . . .	423
6.74.3.1	MATRIX_START_INDEX . . . . .	423
6.74.3.2	PI . . . . .	423
6.75	multiscale::verification::GlobalLogicPropertyAttribute Class Reference . .	423
6.75.1	Detailed Description . . . . .	424
6.75.2	Member Data Documentation . . . . .	424
6.75.2.1	endTimeepoint . . . . .	424
6.75.2.2	logicProperty . . . . .	424
6.75.2.3	startTimeepoint . . . . .	424

6.76 grammar Class Reference . . . . .	424
6.77 multiscale::verification::HeterogeneousTimeseriesComponentAttribute - Class Reference . . . . .	425
6.77.1 Detailed Description . . . . .	425
6.77.2 Member Data Documentation . . . . .	425
6.77.2.1 heterogeneousTimeseriesComponent . . . . .	425
6.78 multiscale::verification::HeterogeneousTimeseriesComponentType- Parser Struct Reference . . . . .	425
6.78.1 Detailed Description . . . . .	426
6.78.2 Constructor & Destructor Documentation . . . . .	426
6.78.2.1 HeterogeneousTimeseriesComponentTypeParser . .	426
6.79 multiscale::verification::TimeseriesComponentEvaluator::Homogeneous- ComponentEvaluator< Relation > Class Template Reference . . . . .	426
6.79.1 Detailed Description . . . . .	427
6.79.2 Member Function Documentation . . . . .	427
6.79.2.1 computeEndIndex . . . . .	427
6.79.2.2 hasValidSuccessor . . . . .	427
6.79.2.3 startIndex . . . . .	428
6.80 multiscale::verification::HomogeneousHomogeneousTimeseries- Attribute Class Reference . . . . .	428
6.80.1 Detailed Description . . . . .	429
6.80.2 Member Data Documentation . . . . .	429
6.80.2.1 homogeneousTimeseriesComponent . . . . .	429
6.80.2.2 homogeneousTimeseriesMeasure . . . . .	429
6.80.2.3 temporalNumericMeasureCollection . . . . .	430
6.81 multiscale::verification::HomogeneousTimeseriesComponentAttribute Class Reference . . . . .	430
6.81.1 Detailed Description . . . . .	430
6.81.2 Member Data Documentation . . . . .	430
6.81.2.1 homogeneousTimeseriesComponent . . . . .	430
6.82 multiscale::verification::HomogeneousTimeseriesComponentType- Parser Struct Reference . . . . .	431
6.82.1 Detailed Description . . . . .	431
6.82.2 Constructor & Destructor Documentation . . . . .	431
6.82.2.1 HomogeneousTimeseriesComponentTypeParser . .	431

6.83	multiscale::verification::HomogeneousTimeseriesMeasureAttribute	-
	Class Reference	431
6.83.1	Detailed Description	432
6.83.2	Member Data Documentation	432
6.83.2.1	homogeneousTimeseriesMeasure	432
6.84	multiscale::verification::HomogeneousTimeseriesMeasureTypeParser	
	Struct Reference	432
6.84.1	Detailed Description	432
6.84.2	Constructor & Destructor Documentation	432
6.84.2.1	HomogeneousTimeseriesMeasureTypeParser	432
6.85	multiscale::verification::ImplicationConstraintAttribute	Class Reference
	433	
6.85.1	Detailed Description	433
6.85.2	Member Data Documentation	433
6.85.2.1	constraint	433
6.86	multiscale::verification::ImplicationLogicPropertyAttribute	Class Reference
	433	
6.86.1	Detailed Description	434
6.86.2	Member Data Documentation	434
6.86.2.1	logicProperty	434
6.87	multiscale::IndexOutOfBoundsException	Class Reference
	434	
6.87.1	Detailed Description	437
6.87.2	Constructor & Destructor Documentation	437
6.87.2.1	IndexOutOfBoundsException	437
6.87.2.2	IndexOutOfBoundsException	437
6.87.2.3	IndexOutOfBoundsException	437
6.88	multiscale::InvalidInputException	Class Reference
	437	
6.88.1	Detailed Description	440
6.88.2	Constructor & Destructor Documentation	440
6.88.2.1	InvalidInputException	440
6.88.2.2	InvalidInputException	440
6.88.2.3	InvalidInputException	440
6.89	multiscale::InvalidOutputException	Class Reference
	440	
6.89.1	Detailed Description	443
6.89.2	Constructor & Destructor Documentation	443
6.89.2.1	InvalidOutputException	443

6.89.2.2	InvalidOutputException	443
6.89.2.3	InvalidOutputException	443
6.90	multiscale::IOException Class Reference	443
6.90.1	Detailed Description	446
6.90.2	Constructor & Destructor Documentation	446
6.90.2.1	IOException	446
6.90.2.2	IOException	446
6.90.2.3	IOException	446
6.91	multiscale::LexicographicNumberIterator Class Reference	446
6.91.1	Detailed Description	449
6.91.2	Constructor & Destructor Documentation	450
6.91.2.1	LexicographicNumberIterator	450
6.91.2.2	~LexicographicNumberIterator	450
6.91.3	Member Function Documentation	450
6.91.3.1	digitsToNumber	450
6.91.3.2	hasNextInitialised	450
6.91.3.3	initialise	450
6.91.3.4	isLargerThanUpperBound	451
6.91.3.5	number	451
6.91.3.6	numberToDigits	451
6.91.3.7	padWithZeros	452
6.91.3.8	resetCurrentNumber	452
6.91.3.9	reverseDigits	452
6.91.4	Member Data Documentation	452
6.91.4.1	currentNumberDigits	452
6.91.4.2	upperBoundDigits	453
6.92	multiscale::verification::LogicPropertyAttribute Class Reference	453
6.92.1	Detailed Description	455
6.92.2	Constructor & Destructor Documentation	455
6.92.2.1	LogicPropertyAttribute	455
6.92.2.2	LogicPropertyAttribute	455
6.92.3	Member Data Documentation	455
6.92.3.1	firstLogicProperty	455
6.92.3.2	nextLogicProperties	456

6.93 multiscale::verification::LogicPropertyDataReader Class Reference . . . . .	456
6.93.1 Detailed Description . . . . .	458
6.93.2 Member Function Documentation . . . . .	459
6.93.2.1 appendLineUsingStringBuilder . . . . .	459
6.93.2.2 createNewLogicProperty . . . . .	459
6.93.2.3 processLineFromInputFile . . . . .	459
6.93.2.4 readLogicPropertiesFromFile . . . . .	460
6.93.2.5 readLogicPropertiesFromOpenStream . . . . .	460
6.93.2.6 readLogicPropertiesFromValidFilepath . . . . .	460
6.93.2.7 removeStringBuilderContents . . . . .	461
6.93.3 Member Data Documentation . . . . .	461
6.93.3.1 CHAR_START_COMMENT . . . . .	461
6.93.3.2 CHAR_START_LOGIC_PROPERTY . . . . .	461
6.93.3.3 ERR_INVALID_INPUT_PATH . . . . .	461
6.93.3.4 ERR_OPEN_INPUT_FILE . . . . .	461
6.93.3.5 stringBuilder . . . . .	461
6.94 multiscale::verification::LogicPropertyGrammar< Iterator > Class - Template Reference . . . . .	462
6.94.1 Detailed Description . . . . .	467
6.94.2 Constructor & Destructor Documentation . . . . .	467
6.94.2.1 LogicPropertyGrammar . . . . .	467
6.94.3 Member Function Documentation . . . . .	467
6.94.3.1 assignNamesToComparatorRules . . . . .	467
6.94.3.2 assignNamesToComposedLogicPropertyRules . . . . .	467
6.94.3.3 assignNamesToLogicPropertiesRules . . . . .	467
6.94.3.4 assignNamesToLogicPropertyRules . . . . .	468
6.94.3.5 assignNamesToPrimaryLogicPropertyRules . . . . .	468
6.94.3.6 assignNamesToProbabilisticLogicPropertyRules . . . . .	468
6.94.3.7 assignNamesToRules . . . . .	468
6.94.3.8 assignNamesToSimilarityMeasureRules . . . . .	468
6.94.3.9 initialise . . . . .	468
6.94.3.10 initialiseComparatorRuleDebugging . . . . .	469
6.94.3.11 initialiseComparatorRules . . . . .	469
6.94.3.12 initialiseComposedLogicPropertyErrorHandlerSupport	469

6.94.3.13 initialiseComposedLogicPropertyRule . . . . .	469
6.94.3.14 initialiseComposedLogicPropertyRuleDebugging . . . . .	469
6.94.3.15 initialiseDebugSupport . . . . .	470
6.94.3.16 initialiseErrorHandlingSupport . . . . .	470
6.94.3.17 initialiseGrammar . . . . .	470
6.94.3.18 initialiseLogicPropertiesErrorHandlingSupport . . . . .	470
6.94.3.19 initialiseLogicPropertiesRules . . . . .	470
6.94.3.20 initialiseLogicPropertiesRulesDebugging . . . . .	470
6.94.3.21 initialiseLogicPropertyRule . . . . .	471
6.94.3.22 initialiseLogicPropertyRuleDebugging . . . . .	471
6.94.3.23 initialisePrimaryLogicPropertyErrorHandlingSupport . . . . .	471
6.94.3.24 initialisePrimaryLogicPropertyRule . . . . .	471
6.94.3.25 initialisePrimaryLogicPropertyRuleDebugging . . . . .	471
6.94.3.26 initialiseProbabilisticLogicPropertyErrorHandling-Support . . . . .	471
6.94.3.27 initialiseProbabilisticLogicPropertyRule . . . . .	472
6.94.3.28 initialiseProbabilisticLogicPropertyRuleDebugging . . . . .	472
6.94.3.29 initialiseRulesDebugging . . . . .	472
6.94.3.30 initialiseSimilarityMeasureRuleDebugging . . . . .	472
6.94.3.31 initialiseSimilarityMeasureRules . . . . .	472
6.94.4 Member Data Documentation . . . . .	472
6.94.4.1 andLogicPropertyRule . . . . .	473
6.94.4.2 binaryNumericNumericRule . . . . .	473
6.94.4.3 changeMeasureRule . . . . .	473
6.94.4.4 changeTemporalNumericMeasureRule . . . . .	473
6.94.4.5 comparatorNonEqualTypeParser . . . . .	473
6.94.4.6 comparatorRule . . . . .	473
6.94.4.7 equivalenceLogicPropertyRule . . . . .	474
6.94.4.8 futureLogicPropertyRule . . . . .	474
6.94.4.9 globalLogicPropertyRule . . . . .	474
6.94.4.10 implicationLogicPropertyRule . . . . .	474
6.94.4.11 logicPropertyRule . . . . .	474
6.94.4.12 nextKLogicPropertyRule . . . . .	474
6.94.4.13 nextLogicPropertyRule . . . . .	475

6.94.4.14	notLogicPropertyRule . . . . .	475
6.94.4.15	orLogicPropertyRule . . . . .	475
6.94.4.16	primaryLogicPropertyRule . . . . .	475
6.94.4.17	primaryNumericMeasureRule . . . . .	475
6.94.4.18	probabilisticLogicPropertyComparatorRule . . . . .	475
6.94.4.19	probabilisticLogicPropertyRule . . . . .	476
6.94.4.20	probabilityRule . . . . .	476
6.94.4.21	similarityMeasureRule . . . . .	476
6.94.4.22	similarityMeasureTypeParser . . . . .	476
6.94.4.23	similarityTemporalNumericCollectionRule . . . . .	476
6.94.4.24	temporalNumericCollectionRule . . . . .	476
6.94.4.25	temporalNumericComparisonRule . . . . .	477
6.94.4.26	temporalNumericMeasureRule . . . . .	477
6.94.4.27	unaryNumericNumericRule . . . . .	477
6.94.4.28	untilLogicPropertyRule . . . . .	477
6.95	multiscale::verification::LogicPropertyVisitor Class Reference . . . . .	477
6.95.1	Detailed Description . . . . .	482
6.95.2	Constructor & Destructor Documentation . . . . .	482
6.95.2.1	LogicPropertyVisitor . . . . .	482
6.95.3	Member Function Documentation . . . . .	482
6.95.3.1	areSimilarValues . . . . .	482
6.95.3.2	computeDissimilarityValue . . . . .	483
6.95.3.3	constructEvaluationLogicProperty . . . . .	483
6.95.3.4	evaluate . . . . .	484
6.95.3.5	evaluate . . . . .	484
6.95.3.6	evaluateChangeLhsTemporalNumericMeasure . . . . .	485
6.95.3.7	evaluateChangeTemporalNumericMeasure . . . . .	485
6.95.3.8	evaluateNextKLogicProperty . . . . .	486
6.95.3.9	evaluateNextKLogicProperty . . . . .	486
6.95.3.10	evaluateNextLogicProperties . . . . .	486
6.95.3.11	evaluateNextLogicProperty . . . . .	487
6.95.3.12	evaluateSimilarTemporalNumericCollection . . . . .	487
6.95.3.13	evaluateTemporalLogicPropertyWithStartAndEnd- Timepoints . . . . .	488

6.95.3.14 evaluateTemporalLogicPropertyWithStartAndEnd- Timepoints . . . . .	489
6.95.3.15 evaluateTemporalLogicPropertyWithStartAndEnd- Timepoints . . . . .	489
6.95.3.16 evaluateTemporalLogicPropertyWithStartEnd- Timepoints . . . . .	490
6.95.3.17 evaluateTemporalNumericComparison . . . . .	490
6.95.3.18 evaluateTemporalNumericMeasure . . . . .	491
6.95.3.19 isLhsSimilarToRhs . . . . .	491
6.95.3.20 operator() . . . . .	492
6.95.3.21 operator() . . . . .	492
6.95.3.22 operator() . . . . .	492
6.95.3.23 operator() . . . . .	493
6.95.3.24 operator() . . . . .	493
6.95.3.25 operator() . . . . .	494
6.95.3.26 operator() . . . . .	494
6.95.3.27 operator() . . . . .	494
6.95.3.28 operator() . . . . .	495
6.95.3.29 operator() . . . . .	495
6.95.3.30 operator() . . . . .	496
6.95.3.31 operator() . . . . .	496
6.95.3.32 operator() . . . . .	496
6.95.3.33 operator() . . . . .	497
6.95.3.34 operator() . . . . .	497
6.95.3.35 operator() . . . . .	498
6.95.3.36 printExceptionMessage . . . . .	498
6.95.4 Member Data Documentation . . . . .	498
6.95.4.1 evaluationLogicProperty . . . . .	498
6.95.4.2 multiscaleArchitectureGraph . . . . .	499
6.95.4.3 precedingTruthValue . . . . .	499
6.95.4.4 trace . . . . .	499
6.96 multiscale::analysis::MatFactory Class Reference . . . . .	499
6.96.1 Detailed Description . . . . .	503
6.96.2 Constructor & Destructor Documentation . . . . .	503
6.96.2.1 MatFactory . . . . .	503

---

6.96.2.2	~MatFactory . . . . .	503
6.96.3	Member Function Documentation . . . . .	503
6.96.3.1	closeInputStream . . . . .	503
6.96.3.2	createFromImageFile . . . . .	503
6.96.3.3	createFromTextFile . . . . .	504
6.96.3.4	initInputFile . . . . .	504
6.96.3.5	isValidInputImage . . . . .	504
6.96.3.6	readNextValueFromFile . . . . .	505
6.96.3.7	readValuesFromFile . . . . .	505
6.96.4	Member Data Documentation . . . . .	505
6.96.4.1	cols . . . . .	505
6.96.4.2	ERR_INPUT_FILE_EXTRA_DATA_BEGIN . . . . .	506
6.96.4.3	ERR_INPUT_FILE_EXTRA_DATA_END . . . . .	506
6.96.4.4	ERR_INVALID_IMAGE_FILE_BEGIN . . . . .	506
6.96.4.5	ERR_INVALID_IMAGE_FILE_END . . . . .	506
6.96.4.6	ERR_OPEN_INPUT_FILE_BEGIN . . . . .	506
6.96.4.7	ERR_OPEN_INPUT_FILE_END . . . . .	506
6.96.4.8	INPUT_VALUE_PRECISION . . . . .	506
6.96.4.9	rows . . . . .	507
6.96.4.10	simulationTime . . . . .	507
6.97	multiscale::MinEnclosingTriangleFinder Class Reference . . . . .	507
6.97.1	Detailed Description . . . . .	512
6.97.2	Constructor & Destructor Documentation . . . . .	512
6.97.2.1	MinEnclosingTriangleFinder . . . . .	512
6.97.2.2	~MinEnclosingTriangleFinder . . . . .	512
6.97.3	Member Function Documentation . . . . .	513
6.97.3.1	advance . . . . .	513
6.97.3.2	advanceBToRightChain . . . . .	513
6.97.3.3	almostEqual . . . . .	513
6.97.3.4	areIdenticalLines . . . . .	514
6.97.3.5	areIntersectingLines . . . . .	514
6.97.3.6	find . . . . .	515
6.97.3.7	findGammaIntersectionPoints . . . . .	515
6.97.3.8	findMinEnclosingTriangle . . . . .	516

6.97.3.9	findMinEnclosingTriangle	516
6.97.3.10	findMinTriangle	516
6.97.3.11	findVertexCOnSideB	517
6.97.3.12	gamma	517
6.97.3.13	height	518
6.97.3.14	height	518
6.97.3.15	initialise	518
6.97.3.16	initialiseAlgorithmVariables	519
6.97.3.17	initialiseConvexPolygon	519
6.97.3.18	intersects	519
6.97.3.19	intersectsAbove	520
6.97.3.20	intersectsAboveOrBelow	520
6.97.3.21	intersectsBelow	521
6.97.3.22	isFlushAngleBetweenPredecessorAndSuccessor	521
6.97.3.23	isGammaAngleBetween	521
6.97.3.24	isGammaAngleEqualTo	522
6.97.3.25	isLocalMinimalTriangle	522
6.97.3.26	isNotBTangency	522
6.97.3.27	isValidMinimalTriangle	523
6.97.3.28	lineEquationParameters	523
6.97.3.29	middlePointOfSideB	523
6.97.3.30	moveAIfLowAndBIfHigh	524
6.97.3.31	predecessor	524
6.97.3.32	returnMinEnclosingTriangle	524
6.97.3.33	searchForBTangency	525
6.97.3.34	successor	525
6.97.3.35	updateMinEnclosingTriangle	525
6.97.3.36	updateSideB	526
6.97.3.37	updateSidesBA	526
6.97.3.38	updateSidesCA	526
6.97.4	Member Data Documentation	526
6.97.4.1	a	526
6.97.4.2	area	527
6.97.4.3	b	527

6.97.4.4	c	527
6.97.4.5	CONVEX_HULL_CLOCKWISE	527
6.97.4.6	EPSILON	527
6.97.4.7	ERR_MIDPOINT_SIDE_B	528
6.97.4.8	ERR_NR_POINTS	528
6.97.4.9	ERR_SIDE_B_GAMMA	528
6.97.4.10	ERR_TRIANGLE_VERTICES	528
6.97.4.11	ERR_VERTEX_C_ON_SIDE_B	528
6.97.4.12	INTERSECTS_ABOVE	528
6.97.4.13	INTERSECTS_BELOW	528
6.97.4.14	INTERSECTS_CRITICAL	529
6.97.4.15	INTERSECTS_LIMIT	529
6.97.4.16	nrOfPoints	529
6.97.4.17	polygon	529
6.97.4.18	sideAEndVertex	529
6.97.4.19	sideAStartVertex	530
6.97.4.20	sideBEndVertex	530
6.97.4.21	sideBStartVertex	530
6.97.4.22	sideCEndVertex	530
6.97.4.23	sideCStartVertex	530
6.97.4.24	VALIDATION_SIDE_A_TANGENT	531
6.97.4.25	VALIDATION_SIDE_B_TANGENT	531
6.97.4.26	VALIDATION_SIDES_FLUSH	531
6.97.4.27	validationFlag	531
6.97.4.28	vertexA	531
6.97.4.29	vertexB	532
6.97.4.30	vertexC	532
6.98	multiscaletest::MinEnclosingTriangleFinderTest Class Reference	532
6.98.1	Detailed Description	536
6.98.2	Constructor & Destructor Documentation	536
6.98.2.1	MinEnclosingTriangleFinderTest	536
6.98.2.2	~MinEnclosingTriangleFinderTest	536
6.98.3	Member Function Documentation	536
6.98.3.1	ArePointsEnclosed	536

6.98.3.2	GetRandomNrOfExecutions . . . . .	537
6.98.3.3	GetRandomNrOfPoints . . . . .	537
6.98.3.4	IsOneEdgeFlush . . . . .	537
6.98.3.5	IsTriangleTouchingPolygon . . . . .	537
6.98.3.6	RunTest . . . . .	537
6.98.3.7	TestMorePoints . . . . .	537
6.98.3.8	TestMorePointsAndNonEmptyTriangle . . . . .	538
6.98.3.9	TestNoPoints . . . . .	538
6.98.3.10	TestOnePoint . . . . .	538
6.98.3.11	TestPointsWithNegativeCoordinates . . . . .	538
6.98.3.12	TestPointsWithNegativeXCoordinate . . . . .	538
6.98.3.13	TestPointsWithNegativeYCoordinate . . . . .	538
6.98.3.14	TestRandomPoints . . . . .	538
6.98.3.15	TestThreePoints . . . . .	539
6.98.3.16	TestTwoPoints . . . . .	539
6.98.3.17	ValidateTestResults . . . . .	539
6.98.4	Member Data Documentation . . . . .	539
6.98.4.1	area . . . . .	539
6.98.4.2	convexHull . . . . .	539
6.98.4.3	MAX_NR_EXECUTIONS . . . . .	539
6.98.4.4	MAX_NR_POINTS . . . . .	540
6.98.4.5	MIN_NR_EXECUTIONS . . . . .	540
6.98.4.6	MIN_NR_POINTS . . . . .	540
6.98.4.7	POINT_IN_TRIANGLE_THRESH . . . . .	540
6.98.4.8	points . . . . .	540
6.98.4.9	triangle . . . . .	540
6.99	multiscale::verification::ModelChecker Class Reference . . . . .	540
6.99.1	Detailed Description . . . . .	544
6.99.2	Constructor & Destructor Documentation . . . . .	544
6.99.2.1	ModelChecker . . . . .	544
6.99.2.2	~ModelChecker . . . . .	545
6.99.3	Member Function Documentation . . . . .	545
6.99.3.1	acceptsMoreTraces . . . . .	545
6.99.3.2	doesPropertyHold . . . . .	545

6.99.3.3	doesPropertyHoldUsingPValues . . . . .	545
6.99.3.4	evaluate . . . . .	545
6.99.3.5	getDetailedResults . . . . .	546
6.99.3.6	getDetailedResultsUsingPValues . . . . .	546
6.99.3.7	isGreaterThanOrEqualToComparator . . . . .	546
6.99.3.8	requiresMoreTraces . . . . .	547
6.99.3.9	updateAlternativeHypothesisPValue . . . . .	547
6.99.3.10	updateDerivedModelCheckerForFalseEvaluation . . .	547
6.99.3.11	updateDerivedModelCheckerForTrueEvaluation . . .	548
6.99.3.12	updateHypothesesPValues . . . . .	548
6.99.3.13	updateHypothesesPValuesConsideringComparator .	548
6.99.3.14	updateHypothesesPValuesForGreaterThan . . . . .	549
6.99.3.15	updateHypothesesPValuesForLessThan . . . . .	549
6.99.3.16	updateModelChecker . . . . .	549
6.99.3.17	updateModelCheckerForEvaluationResult . . . . .	550
6.99.3.18	updateModelCheckerForFalseEvaluation . . . . .	550
6.99.3.19	updateModelCheckerForTrueEvaluation . . . . .	550
6.99.3.20	updateNullAndAlternativeHypothesesPValues . . .	550
6.99.3.21	updateNullHypothesisPValue . . . . .	551
6.99.4	Member Data Documentation . . . . .	551
6.99.4.1	abstractSyntaxTree . . . . .	551
6.99.4.2	alternativeHypothesisPValue . . . . .	552
6.99.4.3	arePValuesUpdatedFlag . . . . .	552
6.99.4.4	MSG_OUTPUT_P_VALUE_BEGIN . . . . .	552
6.99.4.5	MSG_OUTPUT_P_VALUE_END . . . . .	552
6.99.4.6	MSG_OUTPUT_P_VALUE_MIDDLE1 . . . . .	552
6.99.4.7	MSG_OUTPUT_P_VALUE_MIDDLE2 . . . . .	552
6.99.4.8	multiscaleArchitectureGraph . . . . .	553
6.99.4.9	nullHypothesisPValue . . . . .	553
6.99.4.10	totalNumberOfEvaluations . . . . .	553
6.99.4.11	totalNumberOfTrueEvaluations . . . . .	553
6.100	multiscale::verification::ModelCheckerFactory Class Reference . . . . .	554
6.100.1	Detailed Description . . . . .	554
6.100.2	Constructor & Destructor Documentation . . . . .	555

6.100.2.1 ModelCheckerFactory . . . . .	555
6.100.2.2 ~ModelCheckerFactory . . . . .	555
6.100.3 Member Function Documentation . . . . .	555
6.100.3.1 createInstance . . . . .	555
6.101 multiscaletest::ModelCheckerTest Class Reference . . . . .	555
6.101.1 Detailed Description . . . . .	558
6.101.2 Member Function Documentation . . . . .	558
6.101.2.1 Initialise . . . . .	558
6.101.2.2 InitialiseAbstractSyntaxTree . . . . .	559
6.101.2.3 InitialiseModelChecker . . . . .	559
6.101.2.4 InitialiseMultiscaleArchitectureGraph . . . . .	559
6.101.2.5 InitialiseSpatioTemporalTraces . . . . .	559
6.101.2.6 InitialiseSpatioTemporalTraceWithAreaValues . . . . .	559
6.101.2.7 RunModelCheckingTest . . . . .	560
6.101.2.8 RunTest . . . . .	560
6.101.2.9 ValidateTestResults . . . . .	560
6.101.3 Member Data Documentation . . . . .	560
6.101.3.1 abstractSyntaxTree . . . . .	560
6.101.3.2 evaluationResult . . . . .	561
6.101.3.3 modelChecker . . . . .	561
6.101.3.4 multiscaleArchitectureGraph . . . . .	561
6.101.3.5 traces . . . . .	561
6.102 multiscale::verification::ModelCheckingException Class Reference . . . . .	561
6.102.1 Detailed Description . . . . .	565
6.102.2 Constructor & Destructor Documentation . . . . .	565
6.102.2.1 ModelCheckingException . . . . .	565
6.102.2.2 ModelCheckingException . . . . .	565
6.103 multiscale::verification::ModelCheckingHelpRequestException Class Reference . . . . .	565
6.103.1 Detailed Description . . . . .	568
6.103.2 Constructor & Destructor Documentation . . . . .	568
6.103.2.1 ModelCheckingHelpRequestException . . . . .	568
6.103.2.2 ModelCheckingHelpRequestException . . . . .	568
6.104 multiscale::verification::ModelCheckingManager Class Reference . . . . .	568

6.104.1 Detailed Description . . . . .	572
6.104.2 Constructor & Destructor Documentation . . . . .	572
6.104.2.1 ModelCheckingManager . . . . .	572
6.104.2.2 ~ModelCheckingManager . . . . .	572
6.104.3 Member Function Documentation . . . . .	572
6.104.3.1 areUnfinishedModelCheckingTasks . . . . .	572
6.104.3.2 createModelCheckers . . . . .	572
6.104.3.3 createNewEvaluationResults . . . . .	573
6.104.3.4 executeExtraEvaluationProgram . . . . .	573
6.104.3.5 executeExtraEvaluationProgramAndPrintMessage . .	573
6.104.3.6 getNextSpatialTemporalTrace . . . . .	573
6.104.3.7 initialise . . . . .	574
6.104.3.8 initialiseExtraEvaluationTimeCounters . . . . .	574
6.104.3.9 initialiseLogicProperties . . . . .	575
6.104.3.10initialiseMultiscaleArchitectureGraph . . . . .	575
6.104.3.11isEvaluationTimeRemaining . . . . .	575
6.104.3.12isValidLogicProperty . . . . .	576
6.104.3.13outputDetailedEvaluationResults . . . . .	576
6.104.3.14outputModelCheckerResults . . . . .	576
6.104.3.15outputModelCheckersResults . . . . .	577
6.104.3.16outputModelCheckersResultsAndPrintMessage . .	577
6.104.3.17parseLogicProperties . . . . .	577
6.104.3.18parseLogicPropertiesAndPrintMessage . . . . .	577
6.104.3.19parseLogicProperty . . . . .	578
6.104.3.20parseLogicPropertyAndPrintMessages . . . . .	578
6.104.3.21printParsingMessage . . . . .	578
6.104.3.22runModelCheckerForTrace . . . . .	579
6.104.3.23runModelCheckers . . . . .	579
6.104.3.24runModelCheckersAndPrintMessage . . . . .	579
6.104.3.25runModelCheckersAndRequestAdditionalTraces . .	579
6.104.3.26runModelCheckersForCurrentlyExistingTraces . .	580
6.104.3.27runModelCheckersForTrace . . . . .	580
6.104.3.28runModelCheckingAndOutputResults . . . . .	580
6.104.3.29unModelCheckingTasks . . . . .	581

6.104.3.30setExtraEvaluationProgramPath . . . . .	581
6.104.3.31setShouldPrintDetailedEvaluation . . . . .	581
6.104.3.32storeNewSpatialTemporalTracePath . . . . .	582
6.104.3.33updateEvaluationResults . . . . .	582
6.104.3.34updateExtraEvaluationStartTime . . . . .	582
6.104.3.35updateTraceReader . . . . .	583
6.104.3.36waitForRetry . . . . .	583
6.104.4 Member Data Documentation . . . . .	583
6.104.4.1 abstractSyntaxTrees . . . . .	583
6.104.4.2 evaluationResults . . . . .	583
6.104.4.3 extraEvaluationElapsedTime . . . . .	583
6.104.4.4 extraEvaluationProgramPath . . . . .	584
6.104.4.5 extraEvaluationStartTime . . . . .	584
6.104.4.6 extraEvaluationTime . . . . .	584
6.104.4.7 logicProperties . . . . .	584
6.104.4.8 logicPropertyReader . . . . .	585
6.104.4.9 modelCheckers . . . . .	585
6.104.4.10multiscaleArchitectureGraph . . . . .	585
6.104.4.11parser . . . . .	585
6.104.4.12PARSER_EMPTY_LOGIC_PROPERTY . . . . .	585
6.104.4.13shouldPrintDetailedEvaluation . . . . .	586
6.104.4.14TRACE_INPUT_REFRESH_TIMEOUT . . . . .	586
6.104.4.15traceReader . . . . .	586
6.104.4.16tracesPaths . . . . .	586
6.105multiscale::verification::ModelCheckingOutputWriter Class Reference . . . . .	586
6.105.1 Detailed Description . . . . .	592
6.105.2 Member Function Documentation . . . . .	592
6.105.2.1 isTraceEvaluatedForLogicProperty . . . . .	592
6.105.2.2 isTraceEvaluatedTrueForLogicProperty . . . . .	593
6.105.2.3 printDetailedEvaluationResults . . . . .	593
6.105.2.4 printDetailedEvaluationResults . . . . .	594
6.105.2.5 printDetailedEvaluationResultsForLogicProperties . . . . .	594
6.105.2.6 printDetailedEvaluationResultsIntroductionMessage . . . . .	595
6.105.2.7 printDetailedTraceEvaluationResult . . . . .	595

6.105.2.8 printEvaluationResultsSummary . . . . .	596
6.105.2.9 printEvaluationResultsSummary . . . . .	596
6.105.2.10printExecuteExtraEvaluationProgramMessage . . . . .	597
6.105.2.11printFailedMessage . . . . .	597
6.105.2.12printInitialisationMessage . . . . .	597
6.105.2.13printIntroductionMessage . . . . .	598
6.105.2.14printLogicPropertyDetailedEvaluationResults . . . . .	598
6.105.2.15printLogicPropertyForResult . . . . .	599
6.105.2.16printLogicPropertyWithTag . . . . .	599
6.105.2.17printModelCheckingDetailedResult . . . . .	600
6.105.2.18printModelCheckingResult . . . . .	600
6.105.2.19printModelCheckingResultMessage . . . . .	600
6.105.2.20printModelCheckingResultsIntroductionMessage . . . . .	601
6.105.2.21printParsingLogicPropertiesBeginMessage . . . . .	601
6.105.2.22printParsingLogicPropertiesEndMessage . . . . .	601
6.105.2.23printParsingLogicPropertyMessage . . . . .	601
6.105.2.24printResultTag . . . . .	602
6.105.2.25printSeparatorTag . . . . .	602
6.105.2.26printStartModelCheckingExecutionMessage . . . . .	602
6.105.2.27printStartTraceEvaluationMessage . . . . .	603
6.105.2.28printSuccessMessage . . . . .	603
6.105.2.29printTimeoutMessage . . . . .	603
6.105.2.30printTraceEvaluationResult . . . . .	604
6.105.2.31printTruthValueDependentMessage . . . . .	604
6.105.2.32updateSummaryEvaluationResults . . . . .	604
6.105.3 Member Data Documentation . . . . .	605
6.105.3.1 MSG_EVALUATION_RESULTS_INTRODUCTION . . . . .	605
6.105.3.2 MSG_EVALUATION_SUMMARY_BEGIN . . . . .	605
6.105.3.3 MSG_EVALUATION_SUMMARY_END . . . . .	605
6.105.3.4 MSG_EXECUTION_TIMEOUT_BEGIN . . . . .	606
6.105.3.5 MSG_EXECUTION_TIMEOUT_END . . . . .	606
6.105.3.6 MSG_INIT_EXECUTION_PARAMETERS . . . . .	606
6.105.3.7 MSG_INIT_EXTRA_EVALUATION_TIME . . . . .	606
6.105.3.8 MSG_INIT_LOGIC_PROPERTIES_PATH . . . . .	606

6.105.3.9	MSG_INIT_TRACES_FOLDER_PATH	606
6.105.3.10	MSG_INTRO_CONTACT	607
6.105.3.11	MSG_INTRO_COPYRIGHT	607
6.105.3.12	MSG_INTRO_MODEL_CHECKING_PARAMETERS	607
6.105.3.13	MSG_INTRO_MODEL_CHECKING_TYPE	607
6.105.3.14	MSG_INTRO_NAME	607
6.105.3.15	MSG_LOGIC_PROPERTY_HOLDS	607
6.105.3.16	MSG_LOGIC_PROPERTY_HOLDS_FALSE	608
6.105.3.17	MSG_LOGIC_PROPERTY_HOLDS_TRUE	608
6.105.3.18	MSG_PARSING_INTRODUCTION	608
6.105.3.19	MSG_RESULTS_INTRODUCTION	608
6.105.3.20	MSG_START_EXTRA_EVALUATION_PROGRAM_EXECUTION	608
6.105.3.21	MSG_START_MODEL_CHECKING_EXECUTION	608
6.105.3.22	MSG_START_TRACE_EVALUATION	609
6.105.3.23	TAG_DETAILS	609
6.105.3.24	TAG_EXECUTE	609
6.105.3.25	TAG_FAILED	609
6.105.3.26	TAG_FALSE	609
6.105.3.27	TAG_INIT	609
6.105.3.28	TAG_INTRO	610
6.105.3.29	TAG_PARSING	610
6.105.3.30	TAG_RESULT	610
6.105.3.31	TAG_SEPARATOR	610
6.105.3.32	TAG_SUCCESS	610
6.105.3.33	TAG_TIMEOUT	610
6.105.3.34	TAG_TRUE	611
6.106	multiscale::verification::MSTMLSubfilesMerger Class Reference	611
6.106.1	Detailed Description	615
6.106.2	Constructor & Destructor Documentation	615
6.106.2.1	MSTMLSubfilesMerger	615
6.106.2.2	~MSTMLSubfilesMerger	616
6.106.3	Member Function Documentation	616
6.106.3.1	addNumericStateVariablesToResultingTraceTimepoint	616

6.106.3.2 addNumericStateVariableToTimepoint . . . . .	616
6.106.3.3 addSpatialEntitiesToResultingTraceTimepoint . . . . .	617
6.106.3.4 addSpatialEntitiesToResultingTraceTimepoint . . . . .	617
6.106.3.5 addSpatialEntityToTimepoint . . . . .	618
6.106.3.6 addSubtraceStateVariablesToEmptyResultingTrace . .	618
6.106.3.7 addSubtraceStateVariablesToNonEmptyResultingTrace	619
6.106.3.8 addSubtraceStateVariablesToResultingTrace . . . . .	619
6.106.3.9 addSubtracesToResultingTrace . . . . .	620
6.106.3.10addSubtraceToResultingTrace . . . . .	620
6.106.3.11areMismatchingTimepointValues . . . . .	620
6.106.3.12convertToTimepointValue . . . . .	621
6.106.3.13getResultingMergedTrace . . . . .	621
6.106.3.14initialise . . . . .	621
6.106.3.15mergeMSTMLSubfiles . . . . .	621
6.106.3.16outputResultingMSTMLFile . . . . .	622
6.106.3.17readTimepointsValues . . . . .	622
6.106.3.18readTimepointsValuesFromStream . . . . .	622
6.106.3.19updateResultingTraceTimepointsValues . . . . .	623
6.106.3.20validateNumberOfTimepointsInResultingTrace . . .	623
6.106.3.21validateSubtrace . . . . .	623
6.106.3.22validateSubtraceNumberOfTimepoints . . . . .	624
6.106.3.23validateSubtraceTimepointsValues . . . . .	624
6.106.4 Member Data Documentation . . . . .	624
6.106.4.1 ERR_EMPTY_RESULTING_MSTML_FILE . . . . .	625
6.106.4.2 ERR_INVALID_FORMAT_TIMEPOINT_VALUE_BE- GIN . . . . .	625
6.106.4.3 ERR_INVALID_FORMAT_TIMEPOINT_VALUE_END	625
6.106.4.4 ERR_INVALID_NR_TIMEPOINTS_BEGIN . . . . .	625
6.106.4.5 ERR_INVALID_NR_TIMEPOINTS_END . . . . .	625
6.106.4.6 ERR_INVALID_NR_TIMEPOINTS_MIDDLE1 . . . . .	625
6.106.4.7 ERR_INVALID_NR_TIMEPOINTS_MIDDLE2 . . . . .	626
6.106.4.8 ERR_INVALID_NR_TIMEPOINTS_MIDDLE3 . . . . .	626
6.106.4.9 ERR_INVALID_NR_TIMEPOINTS_RESULTING_T- RACE_BEGIN . . . . .	626

6.106.4.10	ERR_INVALID_TIMEPOINTS_VALUES_FILE_BEGIN . . . . .	626
6.106.4.11	ERR_INVALID_TIMEPOINTS_VALUES_FILE_END . . . . .	626
6.106.4.12	ERR_NON_MATCHING_TIMEPOINT_VALUE_BEGIN . . . . .	627
6.106.4.13	ERR_NON_MATCHING_TIMEPOINT_VALUE_END . . . . .	627
6.106.4.14	ERR_NUMERIC_STATE_VARIABLE_EXISTS_BEGIN . . . . .	627
6.106.4.15	ERR_NUMERIC_STATE_VARIABLE_EXISTS_END . . . . .	627
6.106.4.16	ERR_NUMERIC_STATE_VARIABLE_EXISTS_MIDDLE . . . . .	627
6.106.4.17	ERR_SPATIAL_ENTITY_EXISTS_BEGIN . . . . .	627
6.106.4.18	ERR_SPATIAL_ENTITY_EXISTS_END . . . . .	628
6.106.4.19	ERR_SPATIAL_ENTITY_EXISTS_MIDDLE . . . . .	628
6.106.4.20	resultingTrace . . . . .	628
6.106.4.21	timepointsValues . . . . .	628
6.106.4.22	timepointsValuesFilePath . . . . .	628
6.106.4.23	traceReader . . . . .	629
6.107	multiscale::MultiplicationOperation Class Reference . . . . .	629
6.107.1	Detailed Description . . . . .	629
6.107.2	Member Function Documentation . . . . .	629
6.107.2.1	operator() . . . . .	629
6.108	multiscale::verification::MultiscaleArchitectureGraph Class Reference . . . . .	630
6.108.1	Detailed Description . . . . .	637
6.108.2	Constructor & Destructor Documentation . . . . .	637
6.108.2.1	MultiscaleArchitectureGraph . . . . .	637
6.108.2.2	~MultiscaleArchitectureGraph . . . . .	637
6.108.3	Member Function Documentation . . . . .	637
6.108.3.1	addEdgeByEndpoints . . . . .	637
6.108.3.2	addEdgeFromPropertyTree . . . . .	637
6.108.3.3	addEdgesFromPropertyTree . . . . .	638
6.108.3.4	addRootVertex . . . . .	638
6.108.3.5	addRootVertexToEmptyMultiscaleArchitectureGraph . . . . .	639
6.108.3.6	addVertexAndCorrespondingEdge . . . . .	639
6.108.3.7	addVertexAndCorrespondingValidEdge . . . . .	640

6.108.3.8 addVertexByScaleAndSubsystem . . . . .	640
6.108.3.9 addVertexFromPropertyTree . . . . .	641
6.108.3.10addVerticesFromPropertyTree . . . . .	641
6.108.3.11checkIfComparedScaleAndSubsystemsAreEncoded- ByGraph . . . . .	641
6.108.3.12checkIfEdgeEndpointsAreDistinct . . . . .	642
6.108.3.13checkIfEdgeEndpointsAreEncodedByGraph . . . . .	642
6.108.3.14checkIfEdgeHeadEndpointHasIndegreeZero . . . . .	643
6.108.3.15checkIfExistsPathFromRootToOtherVertices . . . . .	643
6.108.3.16checkIfExistsPathFromRootToOtherVertices . . . . .	643
6.108.3.17checkIfExistsUniqueRootVertex . . . . .	644
6.108.3.18checkIfExistUnvisitedVertices . . . . .	644
6.108.3.19checkIfNewVertexDoesNotExist . . . . .	644
6.108.3.20checkIfPredecessorVertexExists . . . . .	645
6.108.3.21checkIfValidIndegreeForNonRootVertices . . . . .	645
6.108.3.22computeRootVertexIndex . . . . .	646
6.108.3.23computeVertexName . . . . .	646
6.108.3.24constructFromPropertyTree . . . . .	646
6.108.3.25existsScaleAndSubsystem . . . . .	647
6.108.3.26getEdges . . . . .	647
6.108.3.27getVertices . . . . .	647
6.108.3.28isEmpty . . . . .	647
6.108.3.29isScaleAndSubsystemSmallerThan . . . . .	648
6.108.3.30isUniqueRootVertex . . . . .	648
6.108.3.31isValidScaleAndSubsystemSmallerThan . . . . .	648
6.108.3.32read . . . . .	649
6.108.3.33readFromFile . . . . .	649
6.108.3.34readFromValidFile . . . . .	649
6.108.3.35readFromValidXmlFile . . . . .	650
6.108.3.36reset . . . . .	650
6.108.3.37validate . . . . .	650
6.108.4 Member Data Documentation . . . . .	650
6.108.4.1 ERR_ADD_ROOT_VERTEX_NON_EMPTY_GRAPH	650

6.108.4.2	ERR_COMPARE_SCALE_AND_SUBSYSTEMS_N- OT_ENCODED_BY_GRAPH_BEGIN . . . . .	651
6.108.4.3	ERR_COMPARE_SCALE_AND_SUBSYSTEMS_N- OT_ENCODED_BY_GRAPH_END . . . . .	651
6.108.4.4	ERR_COMPARE_SCALE_AND_SUBSYSTEMS_N- OT_ENCODED_BY_GRAPH_MIDDLE1 . . . . .	651
6.108.4.5	ERR_COMPARE_SCALE_AND_SUBSYSTEMS_N- OT_ENCODED_BY_GRAPH_MIDDLE2 . . . . .	651
6.108.4.6	ERR_INVALID_EDGE_BEGIN . . . . .	651
6.108.4.7	ERR_INVALID_EDGE_HEAD_ENDPOINT_INDEG- REE_NOT_ZERO_BEGIN . . . . .	651
6.108.4.8	ERR_INVALID_EDGE_HEAD_ENDPOINT_INDEG- REE_NOT_ZERO_END . . . . .	652
6.108.4.9	ERR_INVALID_EDGE_HEAD_ENDPOINT_INDEG- REE_NOT_ZERO_MIDDLE1 . . . . .	652
6.108.4.10	ERR_INVALID_EDGE_HEAD_ENDPOINT_INDEG- REE_NOT_ZERO_MIDDLE2 . . . . .	652
6.108.4.11	ERR_INVALID_EDGE_HEAD_ENDPOINT_NOT_E- NCODED_BY_GRAPH_BEGIN . . . . .	652
6.108.4.12	ERR_INVALID_EDGE_HEAD_ENDPOINT_NOT_E- NCODED_BY_GRAPH_END . . . . .	652
6.108.4.13	ERR_INVALID_EDGE_HEAD_ENDPOINT_NOT_E- NCODED_BY_GRAPH_MIDDLE1 . . . . .	652
6.108.4.14	ERR_INVALID_EDGE_HEAD_ENDPOINT_NOT_E- NCODED_BY_GRAPH_MIDDLE2 . . . . .	653
6.108.4.15	ERR_INVALID_EDGE_IDENTICAL_ENDPOINTS_- BEGIN . . . . .	653
6.108.4.16	ERR_INVALID_EDGE_IDENTICAL_ENDPOINTS_- END . . . . .	653
6.108.4.17	ERR_INVALID_EDGE_IDENTICAL_ENDPOINTS_- MIDDLE . . . . .	653
6.108.4.18	ERR_INVALID_EDGE_MIDDLE1 . . . . .	653
6.108.4.19	ERR_INVALID_EDGE_MIDDLE2 . . . . .	653
6.108.4.20	ERR_INVALID_EDGE_NEW_VERTEX_EXISTS_B- EGIN . . . . .	654
6.108.4.21	ERR_INVALID_EDGE_NEW_VERTEX_EXISTS_E- ND . . . . .	654
6.108.4.22	ERR_INVALID_EDGE_NEW_VERTEX_EXISTS_M- IDDLE1 . . . . .	654

6.108.4.23ERR_INVALID_EDGE_NEW_VERTEX_EXISTS_MIDDLE2 . . . . .	654
6.108.4.24ERR_INVALID_EDGE_PREDECESSOR_VERTEX_DOES_NOT_EXIST_BEGIN . . . . .	654
6.108.4.25ERR_INVALID_EDGE_PREDECESSOR_VERTEX_DOES_NOT_EXIST_END . . . . .	654
6.108.4.26ERR_INVALID_EDGE_PREDECESSOR_VERTEX_DOES_NOT_EXIST_MIDDLE1 . . . . .	655
6.108.4.27ERR_INVALID_EDGE_PREDECESSOR_VERTEX_DOES_NOT_EXIST_MIDDLE2 . . . . .	655
6.108.4.28ERR_INVALID_EDGE_TAIL_ENDPOINT_NOT_ENCODED_BY_GRAPH_BEGIN . . . . .	655
6.108.4.29ERR_INVALID_EDGE_TAIL_ENDPOINT_NOT_ENCODED_BY_GRAPH_END . . . . .	655
6.108.4.30ERR_INVALID_EDGE_TAIL_ENDPOINT_NOT_ENCODED_BY_GRAPH_MIDDLE1 . . . . .	655
6.108.4.31ERR_INVALID_EDGE_TAIL_ENDPOINT_NOT_ENCODED_BY_GRAPH_MIDDLE2 . . . . .	655
6.108.4.32ERR_INVALID_INPUT_FILE_PATH_BEGIN . . . . .	656
6.108.4.33ERR_INVALID_INPUT_FILE_PATH_END . . . . .	656
6.108.4.34ERR_INVALID_INPUT_FILE_RELATIVE_TO_XSD_BEGIN . . . . .	656
6.108.4.35ERR_INVALID_INPUT_FILE_RELATIVE_TO_XSD_END . . . . .	656
6.108.4.36ERR_NO_PATH_FROM_ROOT_TO_VERTEX_BEGIN . . . . .	656
6.108.4.37ERR_NO_PATH_FROM_ROOT_TO_VERTEX_END . . . . .	656
6.108.4.38ERR_NO_PATH_FROM_ROOT_TO_VERTEX_MIDDLE . . . . .	657
6.108.4.39ERR_NO_ROOT_VERTEX . . . . .	657
6.108.4.40ERR_ROOT_VERTEX_NOT_UNIQUE . . . . .	657
6.108.4.41inputFilePath . . . . .	657
6.108.4.42LABEL_EDGE . . . . .	657
6.108.4.43LABEL_EDGE_BEGIN_VERTEX . . . . .	657
6.108.4.44LABEL_EDGE_END_VERTEX . . . . .	658
6.108.4.45LABEL_EDGES . . . . .	658
6.108.4.46LABEL_MULTISCALE_ARCHITECTURE_GRAPH . . . . .	658
6.108.4.47LABEL_VERTEX . . . . .	658

6.108.4.48	LABEL_VERTEX_ID . . . . .	658
6.108.4.49	LABEL_VERTICES . . . . .	658
6.108.4.50	MULTISCALE_ARCHITECTURE_GRAPH_INPUT_- FILE_EXTENSION . . . . .	658
6.108.4.51	MULTISCALE_ARCHITECTURE_GRAPH_XSD_P- ATH . . . . .	659
6.108.4.52	NO_PREDECESSOR_INDEX . . . . .	659
6.108.4.53	predecessorIndices . . . . .	659
6.108.4.54	ROOT_VERTEX_PREDECESSOR_INDEX . . . . .	659
6.108.4.55	successorsIndices . . . . .	659
6.108.4.56	vertices . . . . .	660
6.109	multiscale::MultiscaleException Class Reference . . . . .	660
6.109.1	Detailed Description . . . . .	662
6.109.2	Constructor & Destructor Documentation . . . . .	662
6.109.2.1	MultiscaleException . . . . .	662
6.109.2.2	MultiscaleException . . . . .	662
6.109.2.3	MultiscaleException . . . . .	662
6.109.3	Member Function Documentation . . . . .	663
6.109.3.1	constructExplanatoryString . . . . .	663
6.109.3.2	rawMessage . . . . .	663
6.109.3.3	trimRight . . . . .	663
6.109.3.4	what . . . . .	663
6.109.4	Member Data Documentation . . . . .	664
6.109.4.1	explanatoryString . . . . .	664
6.109.4.2	message . . . . .	664
6.110	multiscaletest::MultiscaleTest Class Reference . . . . .	664
6.110.1	Detailed Description . . . . .	665
6.110.2	Constructor & Destructor Documentation . . . . .	665
6.110.2.1	~MultiscaleTest . . . . .	665
6.110.3	Member Function Documentation . . . . .	665
6.110.3.1	RunTest . . . . .	665
6.110.3.2	SetUp . . . . .	665
6.110.3.3	TearDown . . . . .	665
6.110.3.4	ValidateTestResults . . . . .	666

6.110.4 Member Data Documentation . . . . .	666
6.110.4.1 validationFlag . . . . .	666
6.111 multiscale::verification::NextKLogicPropertyAttribute Class Reference . .	666
6.111.1 Detailed Description . . . . .	666
6.111.2 Member Data Documentation . . . . .	666
6.111.2.1 logicProperty . . . . .	666
6.111.2.2 nrOfTimepointsAhead . . . . .	667
6.112 multiscale::verification::NextLogicPropertyAttribute Class Reference . .	667
6.112.1 Detailed Description . . . . .	667
6.112.2 Member Data Documentation . . . . .	667
6.112.2.1 logicProperty . . . . .	667
6.113 multiscale::verification::Nil Class Reference . . . . .	668
6.113.1 Detailed Description . . . . .	668
6.114 multiscale::verification::NotConstraintAttribute Class Reference . . . .	668
6.114.1 Detailed Description . . . . .	668
6.114.2 Member Data Documentation . . . . .	668
6.114.2.1 constraint . . . . .	668
6.115 multiscale::verification::NotLogicPropertyAttribute Class Reference . . .	669
6.115.1 Detailed Description . . . . .	669
6.115.2 Member Data Documentation . . . . .	669
6.115.2.1 logicProperty . . . . .	669
6.116 multiscale::NumberIterator Class Reference . . . . .	669
6.116.1 Detailed Description . . . . .	671
6.116.2 Constructor & Destructor Documentation . . . . .	671
6.116.2.1 NumberIterator . . . . .	671
6.116.2.2 ~NumberIterator . . . . .	671
6.116.3 Member Function Documentation . . . . .	671
6.116.3.1 hasNext . . . . .	671
6.116.3.2 hasNextInitialised . . . . .	672
6.116.3.3 hasNextNotInitialised . . . . .	672
6.116.3.4 init . . . . .	672
6.116.3.5 initialise . . . . .	672
6.116.3.6 number . . . . .	672
6.116.3.7 reset . . . . .	673

6.116.3.8 resetCurrentNumber . . . . .	673
6.116.4 Member Data Documentation . . . . .	673
6.116.4.1 isInitialised . . . . .	673
6.116.4.2 upperBound . . . . .	673
6.117 multiscale::Numeric Class Reference . . . . .	674
6.117.1 Detailed Description . . . . .	680
6.117.2 Member Function Documentation . . . . .	680
6.117.2.1 almostEqual . . . . .	680
6.117.2.2 applyOperation . . . . .	681
6.117.2.3 areOverflowUnderflowFlagsSet . . . . .	681
6.117.2.4 average . . . . .	682
6.117.2.5 average . . . . .	682
6.117.2.6 combinations . . . . .	682
6.117.2.7 computeCombinations . . . . .	683
6.117.2.8 computeKurtosisFirstTerm . . . . .	683
6.117.2.9 computeKurtosisLastTerm . . . . .	683
6.117.2.10 computeKurtosisMiddleTerm . . . . .	683
6.117.2.11 computeMode . . . . .	684
6.117.2.12 computeQuartileValue . . . . .	684
6.117.2.13 computeSkewFirstTerm . . . . .	684
6.117.2.14 computeSkewLastTerm . . . . .	685
6.117.2.15 covariance . . . . .	685
6.117.2.16 covariance . . . . .	685
6.117.2.17 division . . . . .	686
6.117.2.18 factorial . . . . .	687
6.117.2.19 geometricMean . . . . .	687
6.117.2.20 geometricMean . . . . .	687
6.117.2.21 greaterOrEqual . . . . .	687
6.117.2.22 harmonicMean . . . . .	688
6.117.2.23 harmonicMean . . . . .	688
6.117.2.24 isPositive . . . . .	689
6.117.2.25 kurtosis . . . . .	689
6.117.2.26 kurtosis . . . . .	689
6.117.2.27 lessOrEqual . . . . .	690

6.117.2.28og . . . . .	690
6.117.2.29maximum . . . . .	690
6.117.2.30maximum . . . . .	691
6.117.2.31maximum . . . . .	691
6.117.2.32median . . . . .	691
6.117.2.33median . . . . .	692
6.117.2.34minimum . . . . .	692
6.117.2.35minimum . . . . .	692
6.117.2.36mode . . . . .	693
6.117.2.37mode . . . . .	693
6.117.2.38numberInverse . . . . .	693
6.117.2.39percentile . . . . .	694
6.117.2.40percentile . . . . .	694
6.117.2.41printNoValuesWarningMessage . . . . .	694
6.117.2.42product . . . . .	695
6.117.2.43product . . . . .	695
6.117.2.44quartile . . . . .	695
6.117.2.45quartile . . . . .	696
6.117.2.46resetOverflowUnderflowFlags . . . . .	696
6.117.2.47round . . . . .	696
6.117.2.48sign . . . . .	696
6.117.2.49skew . . . . .	697
6.117.2.50skew . . . . .	697
6.117.2.51standardDeviation . . . . .	697
6.117.2.52standardDeviation . . . . .	698
6.117.2.53sum . . . . .	698
6.117.2.54sum . . . . .	698
6.117.2.55validateLogBase . . . . .	698
6.117.2.56validateLogNumber . . . . .	699
6.117.2.57validateLogNumberAndBase . . . . .	699
6.117.2.58validatePercentile . . . . .	699
6.117.2.59validateQuartile . . . . .	700
6.117.2.60variance . . . . .	700
6.117.2.61variance . . . . .	700

6.117.3 Member Data Documentation . . . . .	701
6.117.3.1 epsilon . . . . .	701
6.117.3.2 ERR_COMBINATIONS_END . . . . .	701
6.117.3.3 ERR_COMBINATIONS_MIDDLE . . . . .	701
6.117.3.4 ERR_COMBINATIONS_START . . . . .	701
6.117.3.5 ERR_LOG_BASE_END . . . . .	701
6.117.3.6 ERR_LOG_BASE_START . . . . .	701
6.117.3.7 ERR_LOG_NUMBER_END . . . . .	702
6.117.3.8 ERR_LOG_NUMBER_START . . . . .	702
6.117.3.9 ERR_OVERFLOW_UNDERFLOW . . . . .	702
6.117.3.10 ERR_PERCENTILE_VALUE_END . . . . .	702
6.117.3.11 ERR_PERCENTILE_VALUE_START . . . . .	702
6.117.3.12 ERR_QUARTILE_VALUE_END . . . . .	702
6.117.3.13 ERR_QUARTILE_VALUE_START . . . . .	702
6.117.3.14 WRN_AVERAGE_FUNCTION_NAME . . . . .	703
6.117.3.15 WRN_COVARIANCE_FUNCTION_NAME . . . . .	703
6.117.3.16 WRN_GEOMETRIC_MEAN_FUNCTION_NAME . . . . .	703
6.117.3.17 WRN_HARMONIC_MEAN_FUNCTION_NAME . . . . .	703
6.117.3.18 WRN_KURTOSIS_FUNCTION_NAME . . . . .	703
6.117.3.19 WRN_MAXIMUM_FUNCTION_NAME . . . . .	703
6.117.3.20 WRN_MEDIAN_FUNCTION_NAME . . . . .	703
6.117.3.21 WRN_MINIMUM_FUNCTION_NAME . . . . .	704
6.117.3.22 WRN_MODE_FUNCTION_NAME . . . . .	704
6.117.3.23 WRN_NOT_ENOUGH_VALUES_END . . . . .	704
6.117.3.24 WRN_NOT_ENOUGH_VALUES_START . . . . .	704
6.117.3.25 WRN_NUMBER_INVERSE . . . . .	704
6.117.3.26 WRN_PERCENTILE_FUNCTION_NAME . . . . .	704
6.117.3.27 WRN_PRODUCT_FUNCTION_NAME . . . . .	704
6.117.3.28 WRN_QUARTILE_FUNCTION_NAME . . . . .	705
6.117.3.29 WRN_SKEW_FUNCTION_NAME . . . . .	705
6.117.3.30 WRN_STANDARD_DEVIATION_FUNCTION_NAME	705
6.117.3.31 WRN_SUM_FUNCTION_NAME . . . . .	705
6.117.3.32 WRN_VARIANCE_FUNCTION_NAME . . . . .	705
6.118 multiscale::verification::NumericEvaluator Class Reference . . . . .	705

6.118.1 Detailed Description . . . . .	706
6.118.2 Member Function Documentation . . . . .	706
6.118.2.1 evaluate . . . . .	706
6.118.2.2 evaluate . . . . .	707
6.118.2.3 evaluate . . . . .	707
6.118.2.4 evaluate . . . . .	708
6.118.2.5 evaluate . . . . .	708
6.119 multiscale::NumericException Class Reference . . . . .	708
6.119.1 Detailed Description . . . . .	711
6.119.2 Constructor & Destructor Documentation . . . . .	711
6.119.2.1 NumericException . . . . .	711
6.119.2.2 NumericException . . . . .	711
6.119.2.3 NumericException . . . . .	711
6.120 multiscale::verification::NumericMeasureAttribute Class Reference . . . . .	711
6.120.1 Detailed Description . . . . .	712
6.120.2 Member Data Documentation . . . . .	712
6.120.2.1 numericMeasure . . . . .	712
6.121 multiscale::verification::NumericMeasureCollectionAttribute Class - Reference . . . . .	712
6.121.1 Detailed Description . . . . .	712
6.121.2 Member Data Documentation . . . . .	712
6.121.2.1 numericMeasureCollection . . . . .	712
6.122 multiscale::verification::NumericMeasureCollectionEvaluator Class - Reference . . . . .	713
6.122.1 Detailed Description . . . . .	713
6.122.2 Member Function Documentation . . . . .	713
6.122.2.1 evaluateSpatialMeasureCollection . . . . .	713
6.122.2.2 evaluateTemporalNumericCollection . . . . .	714
6.123 multiscale::verification::NumericMeasureCollectionVisitor Class - Reference . . . . .	715
6.123.1 Detailed Description . . . . .	718
6.123.2 Constructor & Destructor Documentation . . . . .	718
6.123.2.1 NumericMeasureCollectionVisitor . . . . .	718
6.123.3 Member Function Documentation . . . . .	718
6.123.3.1 computeHomogeneousComponentTimeSpans . . . . .	719

6.123.3.2 computeHomogeneousComponentValues . . . . .	719
6.123.3.3 computeIndicesSubCollection . . . . .	719
6.123.3.4 constructSubCollection . . . . .	720
6.123.3.5 evaluateChangeTemporalNumericCollection . . . . .	720
6.123.3.6 evaluateHomogeneousHomogeneousTimeseries . . .	721
6.123.3.7 evaluateTemporalNumericMeasureCollection . . . . .	721
6.123.3.8 evaluateTemporalNumericMeasureCollectionTimepoints	722
6.123.3.9 evaluateTimeseriesComponent . . . . .	722
6.123.3.10evaluateTimeseriesTimeseriesComponent . . . . .	723
6.123.3.11operator() . . . . .	723
6.123.3.12operator() . . . . .	724
6.123.3.13operator() . . . . .	724
6.123.3.14operator() . . . . .	724
6.123.3.15operator() . . . . .	725
6.123.3.16operator() . . . . .	725
6.123.4 Member Data Documentation . . . . .	726
6.123.4.1 multiscaleArchitectureGraph . . . . .	726
6.123.4.2 trace . . . . .	726
6.124multiscale::NumericRangeManipulator Class Reference . . . . .	726
6.124.1 Detailed Description . . . . .	727
6.124.2 Member Function Documentation . . . . .	727
6.124.2.1 convertFromRange . . . . .	727
6.125multiscale::verification::NumericSpatialMeasureAttribute Class Reference	728
6.125.1 Detailed Description . . . . .	728
6.125.2 Member Data Documentation . . . . .	728
6.125.2.1 numericSpatialMeasure . . . . .	728
6.126multiscale::verification::NumericStateVariableAttribute Class Reference .	728
6.126.1 Detailed Description . . . . .	730
6.126.2 Member Data Documentation . . . . .	730
6.126.2.1 scaleAndSubsystem . . . . .	730
6.126.2.2 stateVariable . . . . .	730
6.127multiscale::verification::NumericStateVariableEvaluator Class Reference	730
6.127.1 Detailed Description . . . . .	731
6.127.2 Member Function Documentation . . . . .	731

---

6.127.2.1 evaluate . . . . .	731
6.127.2.2 evaluate . . . . .	731
6.128 multiscale::verification::NumericStateVariableGrammar< Iterator > -	
Class Template Reference . . . . .	732
6.128.1 Detailed Description . . . . .	735
6.128.2 Constructor & Destructor Documentation . . . . .	735
6.128.2.1 NumericStateVariableGrammar . . . . .	735
6.128.3 Member Function Documentation . . . . .	735
6.128.3.1 assignNamesToRules . . . . .	735
6.128.3.2 initialise . . . . .	735
6.128.3.3 initialiseDebugSupport . . . . .	736
6.128.3.4 initialiseErrorHandlingSupport . . . . .	736
6.128.3.5 initialiseGrammar . . . . .	736
6.128.3.6 initialiseRulesDebugging . . . . .	736
6.128.4 Member Data Documentation . . . . .	736
6.128.4.1 numericStateVariableRule . . . . .	736
6.128.4.2 scaleAndSubsystemRule . . . . .	737
6.128.4.3 stateVariableNameRule . . . . .	737
6.128.4.4 stateVariableRule . . . . .	737
6.128.4.5 stateVariableScaleAndSubsystemRule . . . . .	737
6.129 multiscale::verification::NumericStateVariableId Class Reference . . . . .	737
6.129.1 Detailed Description . . . . .	740
6.129.2 Constructor & Destructor Documentation . . . . .	740
6.129.2.1 NumericStateVariableId . . . . .	740
6.129.2.2 ~NumericStateVariableId . . . . .	740
6.129.3 Member Function Documentation . . . . .	740
6.129.3.1 getName . . . . .	740
6.129.3.2 operator< . . . . .	741
6.129.3.3 setName . . . . .	741
6.129.3.4 toString . . . . .	741
6.129.4 Member Data Documentation . . . . .	741
6.129.4.1 name . . . . .	741
6.129.4.2 OUTPUT_STRING_BEGIN . . . . .	742
6.129.4.3 OUTPUT_STRING_END . . . . .	742

6.129.4.4 OUTPUT_STRING_SEPARATOR . . . . .	742
6.130 multiscaletest::NumericStateVariableTraceTest Class Reference . . . . .	742
6.130.1 Detailed Description . . . . .	745
6.130.2 Member Function Documentation . . . . .	745
6.130.2.1 InitialiseTrace . . . . .	745
6.130.3 Member Data Documentation . . . . .	745
6.130.3.1 clustersAreaMinValue . . . . .	745
6.131 multiscale::verification::NumericStatisticalMeasureAttribute Class - Reference . . . . .	745
6.131.1 Detailed Description . . . . .	746
6.131.2 Member Data Documentation . . . . .	746
6.131.2.1 numericStatisticalMeasure . . . . .	746
6.132 multiscale::verification::NumericVisitor Class Reference . . . . .	746
6.132.1 Detailed Description . . . . .	748
6.132.2 Constructor & Destructor Documentation . . . . .	748
6.132.2.1 NumericVisitor . . . . .	748
6.132.3 Member Function Documentation . . . . .	749
6.132.3.1 evaluate . . . . .	749
6.132.3.2 evaluateNumericSpatialMeasure . . . . .	749
6.132.3.3 evaluatePrimaryNumericMeasure . . . . .	749
6.132.3.4 operator() . . . . .	750
6.132.3.5 operator() . . . . .	750
6.132.3.6 operator() . . . . .	750
6.132.3.7 operator() . . . . .	751
6.132.3.8 operator() . . . . .	751
6.132.3.9 operator() . . . . .	751
6.132.3.10operator() . . . . .	752
6.132.3.11operator() . . . . .	752
6.132.3.12operator() . . . . .	753
6.132.3.13operator() . . . . .	753
6.132.4 Member Data Documentation . . . . .	753
6.132.4.1 multiscaleArchitectureGraph . . . . .	754
6.132.4.2 timePoint . . . . .	754
6.133 multiscale::OperatingSystem Class Reference . . . . .	754

6.133.1 Detailed Description . . . . .	756
6.133.2 Member Function Documentation . . . . .	756
6.133.2.1 executeProgram . . . . .	756
6.133.2.2 executeProgramAndVerifyPath . . . . .	757
6.133.2.3 executeProgramOSSpecific . . . . .	757
6.133.2.4 getEnvironmentVariable . . . . .	757
6.133.3 Member Data Documentation . . . . .	757
6.133.3.1 ERR_EXECUTE_PROGRAM . . . . .	757
6.133.3.2 ERR_INVALID_PROGRAM_PATH . . . . .	758
6.133.3.3 TIMEOUT_MAX_NR_SECONDS . . . . .	758
6.133.3.4 TIMEOUT_NR_SECONDS . . . . .	758
6.134 multiscale::verification::OrConstraintAttribute Class Reference . . . . .	758
6.134.1 Detailed Description . . . . .	758
6.134.2 Member Data Documentation . . . . .	758
6.134.2.1 constraint . . . . .	758
6.135 multiscale::verification::OrLogicPropertyAttribute Class Reference . . . . .	759
6.135.1 Detailed Description . . . . .	759
6.135.2 Member Data Documentation . . . . .	759
6.135.2.1 logicProperty . . . . .	759
6.136 multiscale::verification::Parser Class Reference . . . . .	759
6.136.1 Detailed Description . . . . .	760
6.136.2 Constructor & Destructor Documentation . . . . .	761
6.136.2.1 Parser . . . . .	761
6.136.2.2 ~Parser . . . . .	761
6.136.3 Member Function Documentation . . . . .	761
6.136.3.1 checkIfErrorCase . . . . .	761
6.136.3.2 initialise . . . . .	761
6.136.3.3 isStringParsedCompletely . . . . .	761
6.136.3.4 parse . . . . .	762
6.136.3.5 parseLogicalQuery . . . . .	762
6.136.3.6 parseLogicalQuery . . . . .	762
6.136.3.7 setLogicalQuery . . . . .	763
6.136.4 Member Data Documentation . . . . .	763
6.136.4.1 grammar . . . . .	763

6.136.4.2 logicalQuery . . . . .	763
6.136.4.3 logicalQueryEnd . . . . .	763
6.136.4.4 logicalQueryIterator . . . . .	763
6.137 multiscale::verification::ParserGrammarExceptionHandler Class - Reference . . . . .	764
6.137.1 Detailed Description . . . . .	765
6.137.2 Member Function Documentation . . . . .	765
6.137.2.1 getIntroductoryErrorMessage . . . . .	765
6.137.2.2 handleExpectedTokenAtEndOfString . . . . .	765
6.137.2.3 handleExtraInputException . . . . .	765
6.137.2.4 handleProbabilityException . . . . .	765
6.137.2.5 handleUnexpectedTokenException . . . . .	766
6.137.2.6 handleUnexpectedTokenInString . . . . .	766
6.137.2.7 handleUnparseableInputException . . . . .	766
6.137.2.8 trimRight . . . . .	767
6.138 multiscale::verification::ParserGrammarExtraInputException Class - Reference . . . . .	767
6.138.1 Detailed Description . . . . .	768
6.138.2 Constructor & Destructor Documentation . . . . .	769
6.138.2.1 ParserGrammarExtraInputException . . . . .	769
6.138.3 Member Function Documentation . . . . .	769
6.138.3.1 getErrorString . . . . .	769
6.138.4 Member Data Documentation . . . . .	769
6.138.4.1 errorString . . . . .	769
6.139 multiscale::verification::ParserGrammarProbabilityException Class - Reference . . . . .	769
6.139.1 Detailed Description . . . . .	771
6.139.2 Constructor & Destructor Documentation . . . . .	771
6.139.2.1 ParserGrammarProbabilityException . . . . .	771
6.139.3 Member Function Documentation . . . . .	771
6.139.3.1 getErrorString . . . . .	771
6.139.3.2 getExpectedToken . . . . .	771
6.139.4 Member Data Documentation . . . . .	771
6.139.4.1 errorString . . . . .	772
6.139.4.2 expectedToken . . . . .	772

6.140multiscale::verification::ParserGrammarUnexpectedTokenException -	
Class Reference . . . . .	772
6.140.1 Detailed Description . . . . .	774
6.140.2 Constructor & Destructor Documentation . . . . .	774
6.140.2.1 ParserGrammarUnexpectedTokenException . . . . .	774
6.140.3 Member Function Documentation . . . . .	774
6.140.3.1 getErrorString . . . . .	774
6.140.3.2 getToken . . . . .	774
6.140.4 Member Data Documentation . . . . .	774
6.140.4.1 errorString . . . . .	774
6.140.4.2 expectedToken . . . . .	775
6.141multiscale::verification::ParserGrammarUnparseableInputException -	
Class Reference . . . . .	775
6.141.1 Detailed Description . . . . .	776
6.141.2 Constructor & Destructor Documentation . . . . .	777
6.141.2.1 ParserGrammarUnparseableInputException . . . . .	777
6.141.3 Member Function Documentation . . . . .	777
6.141.3.1 getErrorString . . . . .	777
6.141.4 Member Data Documentation . . . . .	777
6.141.4.1 errorString . . . . .	777
6.142multiscale::visualisation::PolarCsvToInputFilesConverter Class Reference	777
6.142.1 Detailed Description . . . . .	781
6.142.2 Constructor & Destructor Documentation . . . . .	781
6.142.2.1 PolarCsvToInputFilesConverter . . . . .	781
6.142.2.2 ~PolarCsvToInputFilesConverter . . . . .	781
6.142.3 Member Function Documentation . . . . .	781
6.142.3.1 computeConcentration . . . . .	781
6.142.3.2 computeConcentrationWrtArea . . . . .	782
6.142.3.3 computeNextPositionConcentration . . . . .	782
6.142.3.4 computeNonScaledConcentration . . . . .	782
6.142.3.5 computeNormalisedConcentration . . . . .	783
6.142.3.6 computeScaledConcentration . . . . .	783
6.142.3.7 computeSimulationTime . . . . .	783
6.142.3.8 convert . . . . .	783

6.142.3.9 initInputFile . . . . .	784
6.142.3.10 initIterators . . . . .	784
6.142.3.11 initMaximumConcentration . . . . .	784
6.142.3.12 initOutputFile . . . . .	784
6.142.3.13 processInputFile . . . . .	785
6.142.3.14 processLine . . . . .	785
6.142.3.15 splitFirstPartInConcentrations . . . . .	785
6.142.3.16 splitLineInConcentrations . . . . .	786
6.142.3.17 splitOtherPartsInConcentrations . . . . .	786
6.142.3.18 updateMaximumConcentration . . . . .	786
6.142.3.19 validateInput . . . . .	786
6.142.3.20 validateInputLine . . . . .	787
6.142.3.21 validateSelectedConcentrationIndex . . . . .	787
6.142.4 Member Data Documentation . . . . .	787
6.142.4.1 circlesIterator . . . . .	787
6.142.4.2 concentrationsIndex . . . . .	787
6.142.4.3 ERR_INPUT_OPEN . . . . .	787
6.142.4.4 ERR_INVALID_VALUE_LINE . . . . .	788
6.142.4.5 ERR_INVALID_VALUE_TOKEN . . . . .	788
6.142.4.6 ERR_NEG_CONCENTRATION . . . . .	788
6.142.4.7 ERR_NEG_SIM_TIME . . . . .	788
6.142.4.8 ERR_NR_CONCENTRATIONS . . . . .	788
6.142.4.9 ERR_SELECTED_CONCENTRATION_INDEX . . . . .	788
6.142.4.10 INPUT_FILE_SEPARATOR . . . . .	788
6.142.4.11 inputFilepath . . . . .	789
6.142.4.12 maximumConcentration . . . . .	789
6.142.4.13 nrOfConcentrationsForPosition . . . . .	789
6.142.4.14 nrOfConcentricCircles . . . . .	789
6.142.4.15 nrOfSectors . . . . .	789
6.142.4.16 OUTPUT_EXTENSION . . . . .	789
6.142.4.17 OUTPUT_FILE_SEPARATOR . . . . .	789
6.142.4.18 OUTPUT_PRECISION . . . . .	790
6.142.4.19 OUTPUT_SEPARATOR . . . . .	790
6.142.4.20 outputPath . . . . .	790

6.142.4.21RADIUS_MIN . . . . .	790
6.142.4.22sectorsIterator . . . . .	790
6.142.4.23selectedConcentrationIndex . . . . .	790
6.142.4.24useLogScaling . . . . .	790
6.143multiscale::visualisation::PolarGnuplotScriptGenerator Class Reference . . . . .	791
6.143.1 Detailed Description . . . . .	794
6.143.2 Member Function Documentation . . . . .	794
6.143.2.1 generateBody . . . . .	794
6.143.2.2 generateFooter . . . . .	794
6.143.2.3 generateHeader . . . . .	794
6.143.2.4 generateScript . . . . .	795
6.143.2.5 outputContent . . . . .	795
6.143.2.6 outputFooter . . . . .	796
6.143.2.7 outputHeader . . . . .	796
6.143.2.8 readContentTemplate . . . . .	796
6.143.3 Member Data Documentation . . . . .	797
6.143.3.1 CONTENT_IN . . . . .	797
6.143.3.2 FOOTER_IN . . . . .	797
6.143.3.3 GNUPLOT_EXTENSION . . . . .	797
6.143.3.4 HEADER_IN . . . . .	797
6.143.3.5 REPLACE_CONTENT_CONCENTRATION . . . . .	797
6.143.3.6 REPLACE_CONTENT_END_ANGLE . . . . .	798
6.143.3.7 REPLACE_CONTENT_INDEX . . . . .	798
6.143.3.8 REPLACE_CONTENT_RADIUS . . . . .	798
6.143.3.9 REPLACE_CONTENT_START_ANGLE . . . . .	798
6.143.3.10REPLACE_HEADER_FILENAME . . . . .	798
6.143.3.11REPLACE_HEADER_SIM_TIME . . . . .	798
6.144multiscale::verification::PrimaryConstraintAttribute Class Reference . . . . .	799
6.144.1 Detailed Description . . . . .	799
6.144.2 Member Data Documentation . . . . .	799
6.144.2.1 primaryConstraint . . . . .	799
6.145multiscale::verification::PrimaryLogicPropertyAttribute Class Reference . . . . .	799
6.145.1 Detailed Description . . . . .	800
6.145.2 Member Data Documentation . . . . .	800

6.145.2.1 primaryLogicProperty . . . . .	800
6.146 multiscale::verification::PrimaryNumericMeasureAttribute Class Reference . . . . .	800
6.146.1 Detailed Description . . . . .	800
6.146.2 Member Data Documentation . . . . .	800
6.146.2.1 primaryNumericMeasure . . . . .	800
6.147 multiscale::verification::PrimaryNumericMeasureGrammar< Iterator > - Class Template Reference . . . . .	801
6.147.1 Detailed Description . . . . .	805
6.147.2 Constructor & Destructor Documentation . . . . .	805
6.147.2.1 PrimaryNumericMeasureGrammar . . . . .	805
6.147.3 Member Function Documentation . . . . .	805
6.147.3.1 assignNamesToNumericSpatialMeasureRules . . . . .	805
6.147.3.2 assignNamesToPrimaryNumericMeasureRules . . . . .	805
6.147.3.3 assignNamesToRules . . . . .	806
6.147.3.4 initialise . . . . .	806
6.147.3.5 initialiseDebugSupport . . . . .	806
6.147.3.6 initialiseErrorHandlingSupport . . . . .	806
6.147.3.7 initialiseGrammar . . . . .	806
6.147.3.8 initialiseNumericSpatialMeasureErrorHandlingSupport . . . . .	806
6.147.3.9 initialiseNumericSpatialMeasureRule . . . . .	807
6.147.3.10 initialiseNumericSpatialMeasureRuleDebugging . . . . .	807
6.147.3.11 initialisePrimaryNumericMeasureRule . . . . .	807
6.147.3.12 initialisePrimaryNumericMeasureRuleDebugging . . . . .	807
6.147.3.13 initialiseRulesDebugging . . . . .	807
6.147.4 Member Data Documentation . . . . .	807
6.147.4.1 binaryNumericMeasureRule . . . . .	808
6.147.4.2 binaryNumericNumericRule . . . . .	808
6.147.4.3 binaryStatisticalMeasureRule . . . . .	808
6.147.4.4 binaryStatisticalQuantileMeasureRule . . . . .	808
6.147.4.5 binaryStatisticalQuantileSpatialRule . . . . .	808
6.147.4.6 binaryStatisticalSpatialRule . . . . .	808
6.147.4.7 numericSpatialMeasureRule . . . . .	809
6.147.4.8 numericStateVariableRule . . . . .	809

6.147.4.9 primaryNumericMeasureRule . . . . .	809
6.147.4.10spatialMeasureCollectionRule . . . . .	809
6.147.4.11unaryNumericMeasureRule . . . . .	809
6.147.4.12unaryNumericNumericRule . . . . .	810
6.147.4.13unaryStatisticalMeasureRule . . . . .	810
6.147.4.14unaryStatisticalSpatialRule . . . . .	810
6.148multiscale::verification::PrimarySpatialMeasureCollectionAttribute - Class Reference . . . . .	810
6.148.1 Detailed Description . . . . .	811
6.148.2 Member Data Documentation . . . . .	811
6.148.2.1 spatialMeasure . . . . .	811
6.148.2.2 subset . . . . .	811
6.149multiscale::verification::ProbabilisticBlackBoxModelChecker Class Reference . . . . .	812
6.149.1 Detailed Description . . . . .	815
6.149.2 Constructor & Destructor Documentation . . . . .	815
6.149.2.1 ProbabilisticBlackBoxModelChecker . . . . .	815
6.149.2.2 ~ProbabilisticBlackBoxModelChecker . . . . .	815
6.149.3 Member Function Documentation . . . . .	815
6.149.3.1 acceptsMoreTraces . . . . .	816
6.149.3.2 doesPropertyHold . . . . .	816
6.149.3.3 getDetailedResults . . . . .	816
6.149.3.4 requiresMoreTraces . . . . .	816
6.149.3.5 updateDerivedModelCheckerForFalseEvaluation . . . . .	816
6.149.3.6 updateDerivedModelCheckerForTrueEvaluation . . . . .	817
6.150multiscale::verification::ProbabilisticBlackBoxModelCheckerFactory - Class Reference . . . . .	817
6.150.1 Detailed Description . . . . .	818
6.150.2 Constructor & Destructor Documentation . . . . .	818
6.150.2.1 ProbabilisticBlackBoxModelCheckerFactory . . . . .	819
6.150.2.2 ~ProbabilisticBlackBoxModelCheckerFactory . . . . .	819
6.150.3 Member Function Documentation . . . . .	819
6.150.3.1 createInstance . . . . .	819
6.151multiscaletest::ProbabilisticBlackBoxModelCheckerTest Class Reference	819
6.151.1 Detailed Description . . . . .	821

6.151.2 Member Function Documentation . . . . .	822
6.151.2.1 InitialiseModelChecker . . . . .	822
6.152multiscale::verification::ProbabilisticLogicPropertyAttribute Class - Reference . . . . .	822
6.152.1 Detailed Description . . . . .	824
6.152.2 Member Function Documentation . . . . .	824
6.152.2.1 evaluate . . . . .	824
6.152.2.2 getComparator . . . . .	824
6.152.2.3 getProbability . . . . .	825
6.152.3 Member Data Documentation . . . . .	825
6.152.3.1 comparator . . . . .	825
6.152.3.2 ERR_TRACE_LENGTH_ZERO . . . . .	825
6.152.3.3 evaluationLogicProperty . . . . .	825
6.152.3.4 logicProperty . . . . .	825
6.152.3.5 probability . . . . .	826
6.153multiscale::verification::ProbabilityErrorHandler Struct Reference . . . . .	826
6.153.1 Detailed Description . . . . .	826
6.153.2 Member Function Documentation . . . . .	827
6.153.2.1 getExpectedTokenAsString . . . . .	827
6.153.2.2 operator() . . . . .	827
6.154multiscale::visualisation::RectangularCsvToInputFilesConverter Class - Reference . . . . .	827
6.154.1 Detailed Description . . . . .	831
6.154.2 Constructor & Destructor Documentation . . . . .	831
6.154.2.1 RectangularCsvToInputFilesConverter . . . . .	831
6.154.2.2 ~RectangularCsvToInputFilesConverter . . . . .	831
6.154.3 Member Function Documentation . . . . .	831
6.154.3.1 computeConcentration . . . . .	831
6.154.3.2 computeNextPositionConcentration . . . . .	831
6.154.3.3 computeNonScaledConcentration . . . . .	832
6.154.3.4 computeNormalisedConcentration . . . . .	832
6.154.3.5 computeScaledConcentration . . . . .	832
6.154.3.6 computeSimulationTime . . . . .	833
6.154.3.7 convert . . . . .	833

6.154.3.8 initInputFile . . . . .	833
6.154.3.9 initIterators . . . . .	833
6.154.3.10 initMaximumConcentration . . . . .	833
6.154.3.11 initOutputFile . . . . .	834
6.154.3.12 processInputFile . . . . .	834
6.154.3.13 processLine . . . . .	834
6.154.3.14 splitLineInConcentrations . . . . .	835
6.154.3.15 splitLineInConcentrations . . . . .	835
6.154.3.16 updateMaximumConcentration . . . . .	835
6.154.3.17 validateInput . . . . .	836
6.154.3.18 validateInputLine . . . . .	836
6.154.3.19 validateSelectedConcentrationIndex . . . . .	836
<b>6.154.4 Member Data Documentation . . . . .</b>	<b>836</b>
6.154.4.1 columnsIterator . . . . .	836
6.154.4.2 concentrationsIndex . . . . .	836
6.154.4.3 ERR_INPUT_OPEN . . . . .	837
6.154.4.4 ERR_INVALID_VALUE_LINE . . . . .	837
6.154.4.5 ERR_INVALID_VALUE_TOKEN . . . . .	837
6.154.4.6 ERR_NEG_CONCENTRATION . . . . .	837
6.154.4.7 ERR_NEG_SIM_TIME . . . . .	837
6.154.4.8 ERR_NR_CONCENTRATIONS . . . . .	837
6.154.4.9 ERR_SELECTED_CONCENTRATION_INDEX . . . . .	837
6.154.4.10 height . . . . .	838
6.154.4.11 INPUT_FILE_SEPARATOR . . . . .	838
6.154.4.12 inputFilepath . . . . .	838
6.154.4.13 maximumConcentration . . . . .	838
6.154.4.14 nrOfConcentrationsForPosition . . . . .	838
6.154.4.15 OUTPUT_EXTENSION . . . . .	838
6.154.4.16 OUTPUT_FILE_SEPARATOR . . . . .	838
6.154.4.17 OUTPUT_PRECISION . . . . .	839
6.154.4.18 OUTPUT_SEPARATOR . . . . .	839
6.154.4.19 outputPath . . . . .	839
6.154.4.20 rowsIterator . . . . .	839
6.154.4.21 selectedConcentrationIndex . . . . .	839

6.154.4.22useLogScaling . . . . .	839
6.154.4.23width . . . . .	840
6.155multiscale::visualisation::RectangularEntityCsvToInputFilesConverter Class Reference . . . . .	840
6.155.1 Detailed Description . . . . .	843
6.155.2 Constructor & Destructor Documentation . . . . .	843
6.155.2.1 RectangularEntityCsvToInputFilesConverter . . . . .	844
6.155.2.2 ~RectangularEntityCsvToInputFilesConverter . . . . .	844
6.155.3 Member Function Documentation . . . . .	844
6.155.3.1 computeCoordinate . . . . .	844
6.155.3.2 computeSimulationTime . . . . .	844
6.155.3.3 convert . . . . .	844
6.155.3.4 initInputFile . . . . .	845
6.155.3.5 initIterators . . . . .	845
6.155.3.6 initOutputFile . . . . .	845
6.155.3.7 processInputFile . . . . .	845
6.155.3.8 processLine . . . . .	846
6.155.3.9 splitLineInCoordinates . . . . .	846
6.155.3.10validateCoordinate . . . . .	846
6.155.3.11validateEntitiesGrid . . . . .	847
6.155.3.12validateInput . . . . .	847
6.155.3.13validateInputLine . . . . .	847
6.155.3.14validateMaxNrOfEntitiesPerPosition . . . . .	847
6.155.3.15validateSimulationTime . . . . .	848
6.155.4 Member Data Documentation . . . . .	848
6.155.4.1 entitiesIterator . . . . .	848
6.155.4.2 ERR_INPUT_OPEN . . . . .	848
6.155.4.3 ERR_INVALID_NR_ENTITIES . . . . .	848
6.155.4.4 ERR_INVALID_OX_COORDINATE . . . . .	848
6.155.4.5 ERR_INVALID_OY_COORDINATE . . . . .	848
6.155.4.6 ERR_INVALID_VALUE_LINE . . . . .	849
6.155.4.7 ERR_INVALID_VALUE_TOKEN . . . . .	849
6.155.4.8 ERR_MAX_NR_ENTITIES . . . . .	849
6.155.4.9 ERR_NEG_SIM_TIME . . . . .	849

6.155.4.10ERR_NR_COORDINATES . . . . .	849
6.155.4.11height . . . . .	849
6.155.4.12INPUT_FILE_SEPARATOR . . . . .	849
6.155.4.13inputFilepath . . . . .	850
6.155.4.14maxNrOfEntitiesPerPosition . . . . .	850
6.155.4.15nrOfEntities . . . . .	850
6.155.4.16OUTPUT_EXTENSION . . . . .	850
6.155.4.17OUTPUT_FILE_SEPARATOR . . . . .	850
6.155.4.18OUTPUT_PRECISION . . . . .	850
6.155.4.19OUTPUT_SEPARATOR . . . . .	850
6.155.4.20outputFilepath . . . . .	851
6.155.4.21width . . . . .	851
6.156multiscale::visualisation::RectangularGnuplotScriptGenerator Class - Reference . . . . .	851
6.156.1 Detailed Description . . . . .	854
6.156.2 Member Function Documentation . . . . .	854
6.156.2.1 generateBody . . . . .	854
6.156.2.2 generateFooter . . . . .	854
6.156.2.3 generateHeader . . . . .	854
6.156.2.4 generateScript . . . . .	855
6.156.2.5 outputContent . . . . .	855
6.156.2.6 outputFooter . . . . .	856
6.156.2.7 outputHeader . . . . .	856
6.156.3 Member Data Documentation . . . . .	857
6.156.3.1 CONTENT_IN . . . . .	857
6.156.3.2 FOOTER_IN . . . . .	857
6.156.3.3 GNUPLOT_EXTENSION . . . . .	857
6.156.3.4 HEADER_IN . . . . .	857
6.156.3.5 OUTPUT_PRECISION . . . . .	857
6.156.3.6 OUTPUT_SEPARATOR . . . . .	857
6.156.3.7 REPLACE_DIMENSION_EXTRA . . . . .	858
6.156.3.8 REPLACE_HEADER_FILENAME . . . . .	858
6.156.3.9 REPLACE_HEADER_HEIGHT . . . . .	858
6.156.3.10REPLACE_HEADER_SIM_TIME . . . . .	858

6.156.3.11REPLACE_HEADER_WIDTH . . . . .	858
6.157multiscale::analysis::RectangularMatFactory Class Reference . . . . .	859
6.157.1 Detailed Description . . . . .	862
6.157.2 Constructor & Destructor Documentation . . . . .	862
6.157.2.1 RectangularMatFactory . . . . .	862
6.157.2.2 ~RectangularMatFactory . . . . .	862
6.157.3 Member Function Documentation . . . . .	862
6.157.3.1 createFromImageFile . . . . .	862
6.157.3.2 readValuesFromFile . . . . .	863
6.157.3.3 validateValue . . . . .	863
6.157.4 Member Data Documentation . . . . .	864
6.157.4.1 ERR_INVALID_VALUE . . . . .	864
6.158multiscale::verification::Region Class Reference . . . . .	864
6.158.1 Detailed Description . . . . .	866
6.159multiscale::analysis::Region Class Reference . . . . .	867
6.159.1 Detailed Description . . . . .	871
6.159.2 Constructor & Destructor Documentation . . . . .	871
6.159.2.1 Region . . . . .	871
6.159.2.2 ~Region . . . . .	872
6.159.3 Member Function Documentation . . . . .	872
6.159.3.1 areValidInputPolygons . . . . .	872
6.159.3.2 areValidInputPolygons . . . . .	872
6.159.3.3 areValidInputValues . . . . .	873
6.159.3.4 computeArealfOuterBoderDefined . . . . .	873
6.159.3.5 computeClusterednessDegreeIfOuterBorderDefined . . . . .	873
6.159.3.6 computeSpatialEntityShapeArealfOuterBoderDefined . . . . .	874
6.159.3.7 getInnerBorderPolygons . . . . .	874
6.159.3.8 getOuterBorderPolygon . . . . .	874
6.159.3.9 isCircularMeasure . . . . .	874
6.159.3.10isRectangularMeasure . . . . .	875
6.159.3.11isTriangularMeasure . . . . .	875
6.159.3.12isValidInputPolygon . . . . .	875
6.159.3.13type . . . . .	875
6.159.3.14updateArea . . . . .	876

6.159.3.15updateCentrePoint . . . . .	876
6.159.3.16updateCentrePointWhenRegionDefinedByMultiple- Points . . . . .	876
6.159.3.17updateCentrePointWhenRegionDefinedBySinglePoint	876
6.159.3.18updateClusterednessDegree . . . . .	876
6.159.3.19updateDensity . . . . .	877
6.159.3.20updatePerimeter . . . . .	877
6.159.3.21updateSpatialEntityShapeArea . . . . .	877
6.159.3.22validateInputValues . . . . .	877
6.159.4 Member Data Documentation . . . . .	878
6.159.4.1 CONTOUR_CLOSED . . . . .	878
6.159.4.2 CONTOUR_ORIENTED . . . . .	878
6.159.4.3 innerBorderPolygons . . . . .	878
6.159.4.4 outerBorderPolygon . . . . .	879
6.160multiscale::analysis::RegionDetector Class Reference . . . . .	879
6.160.1 Detailed Description . . . . .	886
6.160.2 Constructor & Destructor Documentation . . . . .	886
6.160.2.1 RegionDetector . . . . .	886
6.160.2.2 ~RegionDetector . . . . .	886
6.160.3 Member Function Documentation . . . . .	886
6.160.3.1 approximatePolygonOuterBorder . . . . .	886
6.160.3.2 changeContrastAndBrightness . . . . .	887
6.160.3.3 clearPreviousDetectionResults . . . . .	887
6.160.3.4 computeAverageCentroidDistance . . . . .	887
6.160.3.5 computeAverageClusterednessDegree . . . . .	887
6.160.3.6 computeAverageDensity . . . . .	888
6.160.3.7 computeAverageIntensity . . . . .	888
6.160.3.8 computeAverageMeasures . . . . .	888
6.160.3.9 computeRegionDensity . . . . .	889
6.160.3.10computeSumOfAverageCentroidDistances . . . . .	889
6.160.3.11convertAlpha . . . . .	889
6.160.3.12convertBeta . . . . .	890
6.160.3.13createDetectorSpecificTrackbars . . . . .	890
6.160.3.14createMaskForPolygon . . . . .	890

6.160.3.15createPolygon . . . . .	891
6.160.3.16createPolygons . . . . .	891
6.160.3.17createPolygonsFromContours . . . . .	892
6.160.3.18createRegionFromPolygon . . . . .	892
6.160.3.19existContours . . . . .	892
6.160.3.20findPolygonsInImage . . . . .	893
6.160.3.21findRegions . . . . .	893
6.160.3.22getAlpha . . . . .	893
6.160.3.23getBeta . . . . .	893
6.160.3.24getBlurKernelSize . . . . .	894
6.160.3.25getCollectionOfSpatialEntityPseudo3D . . . . .	894
6.160.3.26getDetectorTypeAsString . . . . .	894
6.160.3.27getEpsilon . . . . .	894
6.160.3.28getMorphologicalCloselIterations . . . . .	894
6.160.3.29getOriginXCoordinate . . . . .	894
6.160.3.30getOriginYCoordinate . . . . .	895
6.160.3.31getRegionAreaThresh . . . . .	895
6.160.3.32getRegions . . . . .	895
6.160.3.33getThresholdValue . . . . .	895
6.160.3.34initialiseDetectorSpecificFields . . . . .	895
6.160.3.35initialiseDetectorSpecificImageDependentFields . . . . .	895
6.160.3.36isValidContour . . . . .	896
6.160.3.37isValidHole . . . . .	896
6.160.3.38morphologicalClose . . . . .	896
6.160.3.39outputRegionInnerBordersToImage . . . . .	897
6.160.3.40outputRegionOuterBorderToImage . . . . .	897
6.160.3.41outputRegionToImage . . . . .	897
6.160.3.42outputResultsToImage . . . . .	898
6.160.3.43processImageAndDetect . . . . .	898
6.160.3.44setAlpha . . . . .	898
6.160.3.45setBeta . . . . .	898
6.160.3.46setBlurKernelSize . . . . .	899
6.160.3.47setEpsilon . . . . .	899
6.160.3.48setMorphologicalCloselIterations . . . . .	899

6.160.3.49setOriginXCoordinate . . . . .	900
6.160.3.50setOriginYCoordinate . . . . .	900
6.160.3.51setPolygonInnerContours . . . . .	900
6.160.3.52setPolygonOuterContour . . . . .	901
6.160.3.53setRegionAreaThresh . . . . .	901
6.160.3.54setThresholdValue . . . . .	901
6.160.3.55smoothImage . . . . .	901
6.160.3.56thresholdImage . . . . .	902
6.160.4 Member Data Documentation . . . . .	902
6.160.4.1 alpha . . . . .	902
6.160.4.2 ALPHA_MAX . . . . .	902
6.160.4.3 ALPHA_REAL_MAX . . . . .	902
6.160.4.4 ALPHA_REAL_MIN . . . . .	903
6.160.4.5 beta . . . . .	903
6.160.4.6 BETA_MAX . . . . .	903
6.160.4.7 BETA_REAL_MAX . . . . .	903
6.160.4.8 BETA_REAL_MIN . . . . .	903
6.160.4.9 blurKernelSize . . . . .	903
6.160.4.10CANNY_THRESH_MAX . . . . .	904
6.160.4.11CONTOUR_AREA_ORIENTED . . . . .	904
6.160.4.12DETECTOR_TYPE . . . . .	904
6.160.4.13DISPLAY_LINE_THICKNESS . . . . .	904
6.160.4.14epsilon . . . . .	904
6.160.4.15EPSILON_MAX . . . . .	904
6.160.4.16HIERARCHY_FIRST_CHILD_INDEX . . . . .	904
6.160.4.17HIERARCHY_NEXT_INDEX . . . . .	905
6.160.4.18HIERARCHY_PARENT_INDEX . . . . .	905
6.160.4.19HIERARCHY_PREV_INDEX . . . . .	905
6.160.4.20KERNEL_MAX . . . . .	905
6.160.4.21MORPH_ITER_MAX . . . . .	905
6.160.4.22morphologicalCloseIterations . . . . .	905
6.160.4.23POLYGON_CLOSED . . . . .	905
6.160.4.24REGION_AREA_THRESH_MAX . . . . .	906
6.160.4.25regionAreaThresh . . . . .	906

6.160.4.26regions . . . . .	906
6.160.4.27THRESHOLD_CLUSTEREDNESS . . . . .	906
6.160.4.28THRESHOLD_HOLE_AREA . . . . .	906
6.160.4.29THRESHOLD_MAX . . . . .	906
6.160.4.30thresholdValue . . . . .	907
6.160.4.31TRACKBAR_ALPHA . . . . .	907
6.160.4.32TRACKBAR_BETA . . . . .	907
6.160.4.33TRACKBAR_CANNY . . . . .	907
6.160.4.34TRACKBAR_EPSILON . . . . .	907
6.160.4.35TRACKBAR_KERNEL . . . . .	907
6.160.4.36TRACKBAR_MORPH . . . . .	908
6.160.4.37TRACKBAR_REGION_AREA_THRESH . . . . .	908
6.160.4.38TRACKBAR_THRESHOLD . . . . .	908
6.161 multiscale::verification::ProbabilityErrorHandler::result< typename, typename, typename > Struct Template Reference . . . . .	908
6.161.1 Detailed Description . . . . .	908
6.161.2 Member Typedef Documentation . . . . .	909
6.161.2.1 type . . . . .	909
6.162 multiscale::verification::UnexpectedTokenErrorHandler::result< type- name, typename, typename > Struct Template Reference . . . . .	909
6.162.1 Detailed Description . . . . .	909
6.162.2 Member Typedef Documentation . . . . .	909
6.162.2.1 type . . . . .	909
6.163 multiscale::RGBColourGenerator Class Reference . . . . .	910
6.163.1 Detailed Description . . . . .	910
6.163.2 Member Function Documentation . . . . .	911
6.163.2.1 computeRGBValues . . . . .	911
6.163.2.2 convertHSVToRGB . . . . .	911
6.163.2.3 convertRGBToString . . . . .	911
6.163.2.4 generate . . . . .	911
6.163.2.5 generate . . . . .	912
6.163.3 Member Data Documentation . . . . .	912
6.163.3.1 blue . . . . .	912
6.163.3.2 green . . . . .	912

6.163.3.3 HUE_MAX . . . . .	913
6.163.3.4 HUE_MIN . . . . .	913
6.163.3.5 red . . . . .	913
6.163.3.6 SATURATION . . . . .	913
6.163.3.7 VALUE . . . . .	913
6.164 multiscale::RuntimeException Class Reference . . . . .	913
6.164.1 Detailed Description . . . . .	916
6.164.2 Constructor & Destructor Documentation . . . . .	916
6.164.2.1 RuntimeException . . . . .	916
6.164.2.2 RuntimeException . . . . .	916
6.164.2.3 RuntimeException . . . . .	916
6.165 multiscale::verification::ScaleAndSubsystem Class Reference . . . . .	916
6.165.1 Detailed Description . . . . .	917
6.165.2 Member Data Documentation . . . . .	917
6.165.2.1 DEFAULT_VALUE . . . . .	918
6.166 multiscale::verification::ScaleAndSubsystemAttribute Class Reference . . . . .	918
6.166.1 Detailed Description . . . . .	919
6.166.2 Constructor & Destructor Documentation . . . . .	920
6.166.2.1 ScaleAndSubsystemAttribute . . . . .	920
6.166.3 Member Data Documentation . . . . .	920
6.166.3.1 scaleAndSubsystem . . . . .	920
6.167 multiscale::verification::ScaleAndSubsystemEvaluator Class Reference . . . . .	920
6.167.1 Detailed Description . . . . .	922
6.167.2 Member Function Documentation . . . . .	922
6.167.2.1 areEqualScalesAndSubsystems . . . . .	922
6.167.2.2 validateScaleAndSubsystem . . . . .	922
6.167.3 Member Data Documentation . . . . .	923
6.167.3.1 ERR_INVALID_SCALE_AND_SUBSYSTEM_BEGIN . . . . .	923
6.167.3.2 ERR_INVALID_SCALE_AND_SUBSYSTEM_END . . . . .	923
6.168 multiscale::verification::ScaleAndSubsystemGrammar< Iterator > - Class Template Reference . . . . .	923
6.168.1 Detailed Description . . . . .	926
6.168.2 Constructor & Destructor Documentation . . . . .	926
6.168.2.1 ScaleAndSubsystemGrammar . . . . .	926

6.168.3 Member Function Documentation . . . . .	926
6.168.3.1 assignNamesToRules . . . . .	926
6.168.3.2 initialise . . . . .	926
6.168.3.3 initialiseDebugSupport . . . . .	927
6.168.3.4 initialiseErrorHandlingSupport . . . . .	927
6.168.3.5 initialiseGrammar . . . . .	927
6.168.3.6 initialiseRulesDebugging . . . . .	927
6.168.4 Member Data Documentation . . . . .	927
6.168.4.1 scaleAndSubsystemRule . . . . .	927
6.168.4.2 scaleAndSubsystemStringRule . . . . .	927
6.169 multiscale::verification::ScaleAndSubsystemStringGrammar< Iterator > Class Template Reference . . . . .	928
6.169.1 Detailed Description . . . . .	931
6.169.2 Constructor & Destructor Documentation . . . . .	931
6.169.2.1 ScaleAndSubsystemStringGrammar . . . . .	931
6.169.3 Member Function Documentation . . . . .	931
6.169.3.1 assignNamesToRules . . . . .	931
6.169.3.2 initialise . . . . .	932
6.169.3.3 initialiseDebugSupport . . . . .	932
6.169.3.4 initialiseErrorHandlingSupport . . . . .	932
6.169.3.5 initialiseGrammar . . . . .	932
6.169.3.6 initialiseRulesDebugging . . . . .	932
6.169.4 Member Data Documentation . . . . .	932
6.169.4.1 SCALE_AND_SUBSYSTEM_LABEL . . . . .	933
6.169.4.2 SCALE_AND_SUBSYSTEM_STRING_PATTERN . .	933
6.169.4.3 scaleAndSubsystemStringRule . . . . .	933
6.170 multiscale::analysis::Silhouette Class Reference . . . . .	933
6.170.1 Detailed Description . . . . .	934
6.170.2 Member Function Documentation . . . . .	934
6.170.2.1 computeAverageDissimilarityBtwEntityAndCluster . .	934
6.170.2.2 computeAverageDissimilarityToOtherClusters . . .	935
6.170.2.3 computeAverageDissimilarityWithinCluster . . . .	935
6.170.2.4 computeAverageMeasure . . . . .	935
6.170.2.5 computeMeasure . . . . .	936

6.170.2.6 computeOverallAverageMeasure . . . . .	936
6.170.2.7 validateClusterIndex . . . . .	936
6.170.2.8 validateElementIndex . . . . .	937
6.170.2.9 validateEntityIndex . . . . .	937
6.171 multiscale::verification::SimilarityMeasureAttribute Class Reference . . . . .	937
6.171.1 Detailed Description . . . . .	938
6.171.2 Member Data Documentation . . . . .	938
6.171.2.1 similarityMeasure . . . . .	938
6.172 multiscale::verification::SimilarityMeasureTypeParser Struct Reference . . . . .	938
6.172.1 Detailed Description . . . . .	938
6.172.2 Constructor & Destructor Documentation . . . . .	938
6.172.2.1 SimilarityMeasureTypeParser . . . . .	938
6.173 multiscale::verification::SimilarityTemporalNumericCollectionAttribute Class Reference . . . . .	939
6.173.1 Detailed Description . . . . .	939
6.173.2 Member Data Documentation . . . . .	939
6.173.2.1 lhsTemporalNumericCollection . . . . .	940
6.173.2.2 rhsTemporalNumericCollection . . . . .	940
6.173.2.3 similarityMeasure . . . . .	940
6.173.2.4 toleratedSimilarityDifference . . . . .	940
6.174 multiscale::analysis::SimulationClusterDetector Class Reference . . . . .	940
6.174.1 Detailed Description . . . . .	944
6.174.2 Constructor & Destructor Documentation . . . . .	944
6.174.2.1 SimulationClusterDetector . . . . .	944
6.174.2.2 ~SimulationClusterDetector . . . . .	944
6.174.3 Member Function Documentation . . . . .	944
6.174.3.1 computeEntityCentrePoint . . . . .	944
6.174.3.2 computeEntityContourPoints . . . . .	945
6.174.3.3 computePileUpDegreeAtPosition . . . . .	945
6.174.3.4 detectEntitiesInImage . . . . .	945
6.174.3.5 initialiseDetectorSpecificImageDependentFields . . . . .	946
6.174.3.6 initialiseThresholedImage . . . . .	946
6.174.3.7 isEntityAtPosition . . . . .	946
6.174.3.8 outputClusterCircularShape . . . . .	946

6.174.3.9 outputClusterRectangularShape . . . . .	947
6.174.3.10outputClusterShape . . . . .	947
6.174.3.11outputClusterTolImage . . . . .	948
6.174.3.12outputClusterTriangularShape . . . . .	948
6.174.3.13outputResultsTolImage . . . . .	948
6.174.4 Member Data Documentation . . . . .	948
6.174.4.1 DATAPOINT_THICKNESS . . . . .	949
6.174.4.2 DATAPOINT_WIDTH . . . . .	949
6.174.4.3 ENTITY_THRESH . . . . .	949
6.174.4.4 entityHeight . . . . .	949
6.174.4.5 entityWidth . . . . .	949
6.174.4.6 height . . . . .	949
6.174.4.7 THRESHOLD . . . . .	950
6.174.4.8 THRESHOLD_MAX . . . . .	950
6.174.4.9 thresholdedImage . . . . .	950
6.174.4.10width . . . . .	950
6.175multiscaletest::SpatialEntitiesTraceTest Class Reference . . . . .	950
6.175.1 Detailed Description . . . . .	953
6.175.2 Member Function Documentation . . . . .	953
6.175.2.1 InitialiseTrace . . . . .	953
6.175.3 Member Data Documentation . . . . .	953
6.175.3.1 clustersAreaMaxValue . . . . .	953
6.175.3.2 clustersAreaMinValue . . . . .	953
6.176multiscale::verification::SpatialEntity Class Reference . . . . .	953
6.176.1 Detailed Description . . . . .	956
6.176.2 Constructor & Destructor Documentation . . . . .	956
6.176.2.1 SpatialEntity . . . . .	956
6.176.2.2 ~SpatialEntity . . . . .	957
6.176.3 Member Function Documentation . . . . .	957
6.176.3.1 getSpatialMeasureValue . . . . .	957
6.176.3.2 operator!= . . . . .	957
6.176.3.3 operator!= . . . . .	957
6.176.3.4 operator< . . . . .	958
6.176.3.5 operator< . . . . .	958

6.176.3.6 operator== . . . . .	958
6.176.3.7 operator== . . . . .	959
6.176.3.8 setSpatialMeasureValue . . . . .	959
6.176.3.9 toString . . . . .	959
6.176.3.10 validateSpatialMeasureValue . . . . .	959
6.176.4 Member Data Documentation . . . . .	960
6.176.4.1 ERR_INVALID_SPATIAL_MEASURE_BEGIN . . . . .	960
6.176.4.2 ERR_INVALID_SPATIAL_MEASURE_END . . . . .	960
6.176.4.3 ERR_INVALID_SPATIAL_MEASURE_MIDDLE . . . . .	960
6.176.4.4 OUTPUT_SPATIAL_MEASURE_VALUE_SEPARA- TOR . . . . .	960
6.176.4.5 spatialMeasureValues . . . . .	960
6.177 multiscale::analysis::SpatialEntityPseudo3D Class Reference . . . . .	961
6.177.1 Detailed Description . . . . .	966
6.177.2 Constructor & Destructor Documentation . . . . .	966
6.177.2.1 SpatialEntityPseudo3D . . . . .	966
6.177.2.2 ~SpatialEntityPseudo3D . . . . .	967
6.177.3 Member Function Documentation . . . . .	967
6.177.3.1 fieldNamesToString . . . . .	967
6.177.3.2 fieldValuesToString . . . . .	967
6.177.3.3 getAngle . . . . .	967
6.177.3.4 getArea . . . . .	967
6.177.3.5 getCentre . . . . .	967
6.177.3.6 getCircularMeasure . . . . .	968
6.177.3.7 getClusterednessDegree . . . . .	968
6.177.3.8 getDensity . . . . .	968
6.177.3.9 getDistanceFromOrigin . . . . .	968
6.177.3.10 getPerimeter . . . . .	968
6.177.3.11 getRectangularMeasure . . . . .	969
6.177.3.12 getShape . . . . .	969
6.177.3.13 getShapeAsString . . . . .	969
6.177.3.14 getTriangularMeasure . . . . .	969
6.177.3.15 initialise . . . . .	969
6.177.3.16 isCircularMeasure . . . . .	970

6.177.3.17sRectangularMeasure . . . . .	970
6.177.3.18sTriangularMeasure . . . . .	970
6.177.3.19normalisedShapeMeasure . . . . .	970
6.177.3.20shapeAsString . . . . .	971
6.177.3.21toString . . . . .	971
6.177.3.22type . . . . .	971
6.177.3.23typeAsString . . . . .	971
6.177.3.24updateArea . . . . .	971
6.177.3.25updateCentrePoint . . . . .	972
6.177.3.26updateClusterednessDegree . . . . .	972
6.177.3.27updateDensity . . . . .	972
6.177.3.28updateMeasures . . . . .	972
6.177.3.29updateMeasuresIfRequired . . . . .	972
6.177.3.30updatePerimeter . . . . .	973
6.177.3.31updateShape . . . . .	973
6.177.3.32updateSpatialEntityShapeArea . . . . .	973
6.177.4 Member Data Documentation . . . . .	973
6.177.4.1 angle . . . . .	973
6.177.4.2 area . . . . .	974
6.177.4.3 centre . . . . .	974
6.177.4.4 circularMeasure . . . . .	974
6.177.4.5 clusterednessDegree . . . . .	974
6.177.4.6 CONVEX_HULL_CLOCKWISE . . . . .	974
6.177.4.7 density . . . . .	975
6.177.4.8 distanceFromOrigin . . . . .	975
6.177.4.9 ERR_INPUT . . . . .	975
6.177.4.10ERR_UNDEFINED_TYPE . . . . .	975
6.177.4.11minAreaEnclosingCircleCentre . . . . .	975
6.177.4.12minAreaEnclosingCircleRadius . . . . .	976
6.177.4.13minAreaEnclosingRect . . . . .	976
6.177.4.14minAreaEnclosingTriangle . . . . .	976
6.177.4.15OUTPUT_SEPARATOR . . . . .	976
6.177.4.16perimeter . . . . .	976
6.177.4.17rectangularMeasure . . . . .	977

6.177.4.18shape . . . . .	977
6.177.4.19spatialEntityShapeArea . . . . .	977
6.177.4.20STR_CIRCLE . . . . .	977
6.177.4.21STR_CLUSTER . . . . .	977
6.177.4.22STR_RECTANGLE . . . . .	978
6.177.4.23STR_REGION . . . . .	978
6.177.4.24STR_TRIANGLE . . . . .	978
6.177.4.25STR_UNDEFINED . . . . .	978
6.177.4.26triangularMeasure . . . . .	978
6.177.4.27updateFlag . . . . .	978
6.178multiscale::verification::SpatialMeasureAttribute Class Reference . . . . .	979
6.178.1 Detailed Description . . . . .	979
6.178.2 Member Data Documentation . . . . .	979
6.178.2.1 spatialMeasureType . . . . .	979
6.179multiscale::analysis::SpatialMeasureCalculator Class Reference . . . . .	979
6.179.1 Detailed Description . . . . .	981
6.179.2 Member Function Documentation . . . . .	981
6.179.2.1 computeCircleArea . . . . .	981
6.179.2.2 computeNrOfMinValuePointSides . . . . .	981
6.179.2.3 computePolygonArea . . . . .	981
6.179.2.4 computePolygonHoleArea . . . . .	982
6.179.2.5 computePolygonPerimeter . . . . .	982
6.179.2.6 drawFilledCircleOnImage . . . . .	983
6.179.2.7 drawFilledPolygonOnImage . . . . .	983
6.179.2.8 isMinValuePointDown . . . . .	983
6.179.2.9 isMinValuePointLeft . . . . .	984
6.179.2.10sMinValuePointRight . . . . .	984
6.179.2.11isMinValuePointUp . . . . .	984
6.179.2.12subtractPolygonBorderFromImage . . . . .	985
6.179.3 Member Data Documentation . . . . .	985
6.179.3.1 POINT_MAX_VALUE . . . . .	985
6.179.3.2 POINT_MIN_VALUE . . . . .	985
6.180multiscale::verification::SpatialMeasureCollectionAttribute Class Reference . . . . .	986

6.180.1 Detailed Description . . . . .	986
6.180.2 Member Data Documentation . . . . .	986
6.180.2.1 spatialMeasureCollection . . . . .	986
6.181 multiscale::verification::SpatialMeasureCollectionGrammar< Iterator > Class Template Reference . . . . .	986
6.181.1 Detailed Description . . . . .	991
6.181.2 Constructor & Destructor Documentation . . . . .	991
6.181.2.1 SpatialMeasureCollectionGrammar . . . . .	991
6.181.3 Member Function Documentation . . . . .	991
6.181.3.1 assignNamesToComposedConstraintRules . . . . .	991
6.181.3.2 assignNamesToConstraintRules . . . . .	992
6.181.3.3 assignNamesToConstraintsRules . . . . .	992
6.181.3.4 assignNamesToFilterNumericMeasureRules . . . . .	992
6.181.3.5 assignNamesToNumericSpatialMeasureRules . . . . .	992
6.181.3.6 assignNamesToPrimaryConstraintRules . . . . .	992
6.181.3.7 assignNamesToRules . . . . .	992
6.181.3.8 assignNamesToSpatialMeasureCollectionRules . . . . .	993
6.181.3.9 assignNamesToSpatialMeasureRules . . . . .	993
6.181.3.10 assignNamesToSubsetRules . . . . .	993
6.181.3.11 initialise . . . . .	993
6.181.3.12 initialiseComposedConstraintErrorHandlingSupport . . . . .	993
6.181.3.13 initialiseComposedConstraintRule . . . . .	994
6.181.3.14 initialiseComposedConstraintRuleDebugging . . . . .	994
6.181.3.15 initialiseConstraintRule . . . . .	994
6.181.3.16 initialiseConstraintRuleDebugging . . . . .	994
6.181.3.17 initialiseConstraintsErrorHandlingSupport . . . . .	994
6.181.3.18 initialiseConstraintsRules . . . . .	994
6.181.3.19 initialiseConstraintsRulesDebugging . . . . .	995
6.181.3.20 initialiseDebugSupport . . . . .	995
6.181.3.21 initialiseErrorHandlingSupport . . . . .	995
6.181.3.22 initialiseFilterNumericMeasureErrorHandlingSupport . . . . .	995
6.181.3.23 initialiseFilterNumericMeasureRule . . . . .	995
6.181.3.24 initialiseFilterNumericMeasureRuleDebugging . . . . .	995
6.181.3.25 initialiseGrammar . . . . .	996

6.181.3.26initialisePrimaryConstraintErrorHandlerSupport . . . . .	996
6.181.3.27initialisePrimaryConstraintRule . . . . .	996
6.181.3.28initialisePrimaryConstraintRuleDebugging . . . . .	996
6.181.3.29initialiseRulesDebugging . . . . .	996
6.181.3.30initialiseSpatialMeasureCollectionErrorHandler- Support . . . . .	997
6.181.3.31initialiseSpatialMeasureCollectionRule . . . . .	997
6.181.3.32initialiseSpatialMeasureCollectionRuleDebugging . . .	997
6.181.3.33initialiseSpatialMeasureRule . . . . .	997
6.181.3.34initialiseSpatialMeasureRuleDebugging . . . . .	997
6.181.3.35initialiseSubsetErrorHandlerSupport . . . . .	998
6.181.3.36initialiseSubsetRule . . . . .	998
6.181.3.37initialiseSubsetRuleDebugging . . . . .	998
6.181.4 Member Data Documentation . . . . .	998
6.181.4.1 andConstraintRule . . . . .	998
6.181.4.2 binaryNumericFilterRule . . . . .	998
6.181.4.3 binaryNumericMeasureRule . . . . .	999
6.181.4.4 binaryNumericSpatialMeasureCollectionRule . . . .	999
6.181.4.5 comparatorRule . . . . .	999
6.181.4.6 constraintRule . . . . .	999
6.181.4.7 equivalenceConstraintRule . . . . .	999
6.181.4.8 filterNumericMeasureRule . . . . .	999
6.181.4.9 filterSubsetRule . . . . .	1000
6.181.4.10implicationConstraintRule . . . . .	1000
6.181.4.11notConstraintRule . . . . .	1000
6.181.4.12orConstraintRule . . . . .	1000
6.181.4.13primaryConstraintRule . . . . .	1000
6.181.4.14primaryNumericMeasureRule . . . . .	1001
6.181.4.15primarySpatialMeasureCollectionRule . . . . .	1001
6.181.4.16scaleAndSubsystemRule . . . . .	1001
6.181.4.17spatialMeasureCollectionRule . . . . .	1001
6.181.4.18spatialMeasureRule . . . . .	1001
6.181.4.19spatialMeasureTypeParser . . . . .	1002
6.181.4.20subsetOperationTypeParser . . . . .	1002

6.181.4.21subsetRule . . . . .	1002
6.181.4.22subsetSpecificRule . . . . .	1002
6.181.4.23subsetSpecificTypeParser . . . . .	1002
6.181.4.24subsetSubsetOperationRule . . . . .	1002
6.181.4.25unaryNumericFilterRule . . . . .	1003
6.181.4.26unaryNumericMeasureRule . . . . .	1003
6.181.4.27unaryNumericSpatialMeasureCollectionRule . . . . .	1003
6.181.4.28unaryScaleAndSubsystemConstraintRule . . . . .	1003
6.181.4.29unarySpatialConstraintRule . . . . .	1003
6.182multiscale::verification::SpatialMeasureCollectionVisitor Class Reference	1004
6.182.1 Detailed Description . . . . .	1005
6.182.2 Constructor & Destructor Documentation . . . . .	1005
6.182.2.1 SpatialMeasureCollectionVisitor . . . . .	1005
6.182.3 Member Function Documentation . . . . .	1005
6.182.3.1 computeNrOfConsideredSpatialMeasureValuesPairs .	1005
6.182.3.2 evaluateBinaryNumericSpatialMeasureCollection .	1006
6.182.3.3 evaluateSpatialMeasureCollection . . . . .	1007
6.182.3.4 evaluateUnaryNumericSpatialMeasureCollection .	1007
6.182.3.5 operator() . . . . .	1008
6.182.3.6 operator() . . . . .	1008
6.182.3.7 operator() . . . . .	1008
6.182.4 Member Data Documentation . . . . .	1009
6.182.4.1 multiscaleArchitectureGraph . . . . .	1009
6.182.4.2 timePoint . . . . .	1009
6.183multiscale::verification::SpatialMeasureEvaluator Class Reference . . .	1009
6.183.1 Detailed Description . . . . .	1010
6.183.2 Member Function Documentation . . . . .	1010
6.183.2.1 evaluate . . . . .	1010
6.184multiscale::verification::SpatialMeasureTypeParser Struct Reference . .	1010
6.184.1 Detailed Description . . . . .	1011
6.184.2 Constructor & Destructor Documentation . . . . .	1011
6.184.2.1 SpatialMeasureTypeParser . . . . .	1011
6.185multiscale::verification::SpatialNumericComparisonAttribute Class - Reference . . . . .	1011

6.185.1 Detailed Description . . . . .	1012
6.185.2 Member Data Documentation . . . . .	1012
6.185.2.1 comparator . . . . .	1012
6.185.2.2 numericMeasure . . . . .	1012
6.185.2.3 spatialMeasure . . . . .	1012
6.186 multiscale::verification::SpatialTemporalDataReader Class Reference . . . . .	1012
6.186.1 Detailed Description . . . . .	1016
6.186.2 Constructor & Destructor Documentation . . . . .	1016
6.186.2.1 SpatialTemporalDataReader . . . . .	1016
6.186.2.2 ~SpatialTemporalDataReader . . . . .	1016
6.186.3 Member Function Documentation . . . . .	1017
6.186.3.1 addEntitiesToTimePoint . . . . .	1017
6.186.3.2 addNumericStateVariableToTimePoint . . . . .	1017
6.186.3.3 addSpatialEntityToTimePoint . . . . .	1017
6.186.3.4 addTimePointToTrace . . . . .	1018
6.186.3.5 clearInputFilesSets . . . . .	1018
6.186.3.6 constructSpatialTemporalTrace . . . . .	1018
6.186.3.7 constructSpatialTemporalTrace . . . . .	1019
6.186.3.8 convertTimePointPropertyTreeToTrace . . . . .	1019
6.186.3.9 createDerivedSpatialEntity . . . . .	1019
6.186.3.10 generateSpatialTemporalTrace . . . . .	1020
6.186.3.11 generateSpatialTemporalTrace . . . . .	1020
6.186.3.12 getFilesInFolder . . . . .	1020
6.186.3.13 getNextSpatialTemporalTrace . . . . .	1021
6.186.3.14 getNextSpatialTemporalTrace . . . . .	1021
6.186.3.15 getRandomUnprocessedInputFile . . . . .	1021
6.186.3.16 getRandomValidUnprocessedInputFilepath . . . . .	1021
6.186.3.17 hasNext . . . . .	1022
6.186.3.18 hasValidNext . . . . .	1022
6.186.3.19 initialise . . . . .	1022
6.186.3.20 initialise . . . . .	1022
6.186.3.21 isValidInputFile . . . . .	1022
6.186.3.22 processInvalidInputFile . . . . .	1023
6.186.3.23 processValidInputFile . . . . .	1023

6.186.3.24refresh . . . . .	1024
6.186.3.25setSpatialEntityMeasureValues . . . . .	1024
6.186.3.26setSpatialEntityScaleAndSubsystem . . . . .	1024
6.186.3.27setTimePointValue . . . . .	1025
6.186.3.28timePointHasValue . . . . .	1025
6.186.3.29updateInputFilesSets . . . . .	1025
6.186.3.30validateFolderPath . . . . .	1025
6.186.4 Member Data Documentation . . . . .	1026
6.186.4.1 ERR_INVALID_FOLDER_PATH_BEGIN . . . . .	1026
6.186.4.2 ERR_INVALID_FOLDER_PATH_END . . . . .	1026
6.186.4.3 ERR_NO_VALID_INPUT_FILES_REMAINING . . . . .	1026
6.186.4.4 ERR_UNDEFINED_SPATIAL_ENTITY_TYPE . . . . .	1026
6.186.4.5 folderPath . . . . .	1026
6.186.4.6 INPUT_FILES_EXTENSION . . . . .	1027
6.186.4.7 INPUT_FILES_SCHEMA_PATH . . . . .	1027
6.186.4.8 LABEL_EXPERIMENT . . . . .	1027
6.186.4.9 LABEL_NUMERIC_STATE_VARIABLE . . . . .	1027
6.186.4.10LABEL_NUMERIC_STATE_VARIABLE_NAME . . . . .	1027
6.186.4.11LABEL_NUMERIC_STATE_VARIABLE_SCALE_AN- D_SUBSYSTEM . . . . .	1027
6.186.4.12LABEL_NUMERIC_STATE_VARIABLE_VALUE . . . . .	1028
6.186.4.13LABEL_SPATIAL_ENTITY . . . . .	1028
6.186.4.14LABEL_SPATIAL_ENTITY_SCALE_AND_SUBSYS- TEM . . . . .	1028
6.186.4.15LABEL_SPATIAL_ENTITY_SPATIAL_TYPE . . . . .	1028
6.186.4.16LABEL_TIMEPOINT_VALUE . . . . .	1028
6.186.4.17processedInputFiles . . . . .	1028
6.186.4.18unprocessedInputFiles . . . . .	1029
6.187multiscale::verification::SpatialTemporalDataWriter Class Reference . . . . .	1029
6.187.1 Detailed Description . . . . .	1032
6.187.2 Member Function Documentation . . . . .	1032
6.187.2.1 addAttributeToTree . . . . .	1032
6.187.2.2 addAutoGeneratedCommentToPropertyTree . . . . .	1033
6.187.2.3 addNumericStateVariablesToTimepointTree . . . . .	1033

6.187.2.4 addScaleAndSubsystemAttributeToTree . . . . .	1033
6.187.2.5 addSpatialEntitiesOfSpecificTypeToTimepointTree . .	1034
6.187.2.6 addSpatialEntitiesToTimepointTree . . . . .	1034
6.187.2.7 addSpatialEntityOfSpecificTypeToTimepointTree . .	1034
6.187.2.8 addSpatialMeasuresValuesToTree . . . . .	1035
6.187.2.9 addSpatialTypeAttributeToTree . . . . .	1035
6.187.2.10addTimepointsToExperimentPropertyTree . . . . .	1035
6.187.2.11addValueAttributeToTimepointTree . . . . .	1036
6.187.2.12constructPropertyTreeFromNumericStateVariable . .	1036
6.187.2.13constructPropertyTreeFromTimepoint . . . . .	1036
6.187.2.14constructPropertyTreeFromTrace . . . . .	1037
6.187.2.15outputPropertyTreeInXmlFormatToFile . . . . .	1037
6.187.2.16outputTraceInXmlFormatToFile . . . . .	1037
6.187.3 Member Data Documentation . . . . .	1038
6.187.3.1 CONTENTS_COMMENT_AUTO_GENERATED . . .	1038
6.187.3.2 LABEL_ATTRIBUTE . . . . .	1038
6.187.3.3 LABEL_ATTRIBUTE_SEPARATOR . . . . .	1038
6.187.3.4 LABEL_COMMENT . . . . .	1038
6.187.3.5 LABEL_EXPERIMENT . . . . .	1038
6.187.3.6 LABEL_NUMERIC_STATE_VARIABLE . . . . .	1038
6.187.3.7 LABEL_NUMERIC_STATE_VARIABLE_NAME . . .	1038
6.187.3.8 LABEL_NUMERIC_STATE_VARIABLE_VALUE . . .	1039
6.187.3.9 LABEL_SCALE_AND_SUBSYSTEM_ATTRIBUTE . .	1039
6.187.3.10LABEL_SPATIAL_ENTITY . . . . .	1039
6.187.3.11LABEL_SPATIAL_TYPE_ATTRIBUTE . . . . .	1039
6.187.3.12LABEL_TIMEPOINT . . . . .	1039
6.187.3.13LABEL_TIMEPOINT_VALUE_ATTRIBUTE . . . .	1039
6.188multiscale::verification::SpatialTemporalException Class Reference . . .	1039
6.188.1 Detailed Description . . . . .	1043
6.188.2 Constructor & Destructor Documentation . . . . .	1043
6.188.2.1 SpatialTemporalException . . . . .	1043
6.188.2.2 SpatialTemporalException . . . . .	1043
6.189multiscale::verification::SpatialTemporalTrace Class Reference . . . .	1043
6.189.1 Detailed Description . . . . .	1047

6.189.2 Constructor & Destructor Documentation . . . . .	1047
6.189.2.1 SpatialTemporalTrace . . . . .	1047
6.189.2.2 SpatialTemporalTrace . . . . .	1047
6.189.2.3 ~SpatialTemporalTrace . . . . .	1047
6.189.3 Member Function Documentation . . . . .	1047
6.189.3.1 addTimePoint . . . . .	1047
6.189.3.2 addTimePointsToSubTrace . . . . .	1048
6.189.3.3 advanceTraceBeginIndex . . . . .	1048
6.189.3.4 clear . . . . .	1048
6.189.3.5 getMostRecentlyStoredSubTraceBeginIndex . . . . .	1048
6.189.3.6 getTimePoint . . . . .	1049
6.189.3.7 getTimePointReference . . . . .	1049
6.189.3.8 getTimePointReference . . . . .	1049
6.189.3.9 indexOfFirstTimePointGreaterOrEqualToValue . . . . .	1050
6.189.3.10initialise . . . . .	1050
6.189.3.11length . . . . .	1050
6.189.3.12nextTimePointValue . . . . .	1051
6.189.3.13nextTimePointValueForLastTimePoint . . . . .	1051
6.189.3.14operator== . . . . .	1051
6.189.3.15restoreSubTraceBeginIndex . . . . .	1051
6.189.3.16setSubTraceIndex . . . . .	1052
6.189.3.17setSubTraceWithTimepointsValuesGreaterOrEqualTo . . . . .	1052
6.189.3.18setTraceBeginIndex . . . . .	1052
6.189.3.19storeSubTraceBeginIndex . . . . .	1053
6.189.3.20updateLastTimePointValue . . . . .	1053
6.189.3.21validateAbsoluteIndex . . . . .	1053
6.189.3.22validateIndexRelativeToBeginIndex . . . . .	1054
6.189.3.23validateTimePointValue . . . . .	1054
6.189.3.24validateTimePointValue . . . . .	1054
6.189.3.25validateValue . . . . .	1055
6.189.4 Member Data Documentation . . . . .	1055
6.189.4.1 beginIndex . . . . .	1055
6.189.4.2 beginIndices . . . . .	1055

6.189.4.3	ERR_ABSOLUTE_TIMEPOINT_INDEX_OUT_OF_BOUNDS_END	1056
6.189.4.4	ERR_ABSOLUTE_TIMEPOINT_INDEX_OUT_OF_BOUNDS_START	1056
6.189.4.5	ERR_ITERATOR_NEXT	1056
6.189.4.6	ERR_NEXT_TIMEPOINT_VALUE_NOT_EXISTS	1056
6.189.4.7	ERR_RELATIVE_TIMEPOINT_INDEX_OUT_OF_BOUNDS_END	1056
6.189.4.8	ERR_RELATIVE_TIMEPOINT_INDEX_OUT_OF_BOUNDS_MIDDLE	1056
6.189.4.9	ERR_RELATIVE_TIMEPOINT_INDEX_OUT_OF_BOUNDS_START	1057
6.189.4.10	ERR_TIMEPOINT_END_END	1057
6.189.4.11	ERR_TIMEPOINT_END_MIDDLE	1057
6.189.4.12	ERR_TIMEPOINT_END_START	1057
6.189.4.13	ERR_TIMEPOINT_VALUE_INVALID_END	1057
6.189.4.14	ERR_TIMEPOINT_VALUE_INVALID_MIDDLE	1057
6.189.4.15	ERR_TIMEPOINT_VALUE_INVALID_START	1057
6.189.4.16	ERR_TIMEPOINT_VALUE_OUT_OF_BOUNDS_END	1058
6.189.4.17	ERR_TIMEPOINT_VALUE_OUT_OF_BOUNDS_START	1058
6.189.4.18	sLastTimePointValueInitialised	1058
6.189.4.19	lastTimePointValue	1058
6.189.4.20	TIMEPOINT_INDEX_NOT_FOUND	1058
6.189.4.21	timePoints	1058
6.190	multiscale::StandardNumberIterator Class Reference	1059
6.190.1	Detailed Description	1062
6.190.2	Constructor & Destructor Documentation	1062
6.190.2.1	StandardNumberIterator	1062
6.190.2.2	~StandardNumberIterator	1062
6.190.3	Member Function Documentation	1062
6.190.3.1	hasNextInitialised	1062
6.190.3.2	initialise	1063
6.190.3.3	number	1063
6.190.3.4	resetCurrentNumber	1063

6.190.4 Member Data Documentation . . . . .	1063
6.190.4.1 currentNumber . . . . .	1063
6.191 multiscale::verification::StateVariable Class Reference . . . . .	1063
6.191.1 Detailed Description . . . . .	1066
6.191.2 Constructor & Destructor Documentation . . . . .	1066
6.191.2.1 StateVariable . . . . .	1066
6.191.2.2 ~StateVariable . . . . .	1066
6.191.3 Member Function Documentation . . . . .	1066
6.191.3.1 getScaleAndSubsystem . . . . .	1066
6.191.3.2 setScaleAndSubsystem . . . . .	1066
6.191.4 Member Data Documentation . . . . .	1067
6.191.4.1 scaleAndSubsystem . . . . .	1067
6.192 multiscale::verification::StateVariableAttribute Class Reference . . . . .	1067
6.192.1 Detailed Description . . . . .	1068
6.192.2 Member Data Documentation . . . . .	1068
6.192.2.1 name . . . . .	1069
6.193 multiscale::verification::StatisticalModelChecker Class Reference . . . . .	1069
6.193.1 Detailed Description . . . . .	1074
6.193.2 Constructor & Destructor Documentation . . . . .	1074
6.193.2.1 StatisticalModelChecker . . . . .	1074
6.193.2.2 ~StatisticalModelChecker . . . . .	1075
6.193.3 Member Function Documentation . . . . .	1075
6.193.3.1 acceptsMoreTraces . . . . .	1075
6.193.3.2 computeFPrimeValue . . . . .	1075
6.193.3.3 computeFPrimeValueFirstTerm . . . . .	1075
6.193.3.4 computeFPrimeValueSecondTerm . . . . .	1075
6.193.3.5 computeFValue . . . . .	1076
6.193.3.6 computeFValueFirstTerm . . . . .	1076
6.193.3.7 computeFValueSecondTerm . . . . .	1076
6.193.3.8 computeIndifferenceIntervalHalf . . . . .	1077
6.193.3.9 doesPropertyHold . . . . .	1077
6.193.3.10 doesPropertyHoldConsideringProbabilityComparator .	1077
6.193.3.11 doesPropertyHoldConsideringResult . . . . .	1078
6.193.3.12 getDetailedResults . . . . .	1078

6.193.3.13getDetailedUpdatedResults . . . . .	1078
6.193.3.14initialise . . . . .	1078
6.193.3.15sValidTypeError . . . . .	1079
6.193.3.16requiresMoreTraces . . . . .	1079
6.193.3.17updateDerivedModelCheckerForFalseEvaluation . . . . .	1079
6.193.3.18updateDerivedModelCheckerForTrueEvaluation . . . . .	1079
6.193.3.19updateInitialisedModelCheckingResult . . . . .	1080
6.193.3.20updateModelCheckingResult . . . . .	1080
6.193.3.21updateModelCheckingResult . . . . .	1080
6.193.3.22updateModelCheckingResultEnoughTraces . . . . .	1080
6.193.3.23updateModelCheckingResultNotEnoughTraces . . . . .	1081
6.193.3.24validateTypesErrors . . . . .	1081
6.193.4 Member Data Documentation . . . . .	1081
6.193.4.1 a1FromPaper . . . . .	1081
6.193.4.2 a2FromPaper . . . . .	1082
6.193.4.3 b1FromPaper . . . . .	1082
6.193.4.4 b2FromPaper . . . . .	1082
6.193.4.5 ERR_TYPES_ERROR_VALUES_BEGIN . . . . .	1082
6.193.4.6 ERR_TYPES_ERROR_VALUES_END . . . . .	1082
6.193.4.7 ERR_TYPES_ERROR_VALUES_MIDDLE . . . . .	1083
6.193.4.8 ERR_UNEXPECTED_MODEL_CHECKING_RESU- LT . . . . .	1083
6.193.4.9 INDIFFERENCE_INTERVAL_HALF_K . . . . .	1083
6.193.4.10ndifferenceIntervalHalf . . . . .	1083
6.193.4.11LOGARITHM_ZERO_VALUE . . . . .	1083
6.193.4.12minTypesError . . . . .	1084
6.193.4.13modelCheckingResult . . . . .	1084
6.193.4.14MSG_OUTPUT_MORE_TRACES_REQUIRED . . . . .	1084
6.193.4.15MSG_OUTPUT_RESULT_BEGIN . . . . .	1084
6.193.4.16MSG_OUTPUT_RESULT_END . . . . .	1084
6.193.4.17MSG_OUTPUT_RESULT_MIDDLE . . . . .	1085
6.193.4.18MSG_OUTPUT_SEPARATOR . . . . .	1085
6.193.4.19probability . . . . .	1085
6.193.4.20typeIError . . . . .	1085

6.193.4.2 <code>typellError</code>	1085
6.194 <code>multiscale::verification::StatisticalModelCheckerFactory</code> Class Reference	1086
6.194.1 Detailed Description	1087
6.194.2 Constructor & Destructor Documentation	1088
6.194.2.1 <code>StatisticalModelCheckerFactory</code>	1088
6.194.2.2 <code>~StatisticalModelCheckerFactory</code>	1088
6.194.3 Member Function Documentation	1088
6.194.3.1 <code>createInstance</code>	1088
6.194.4 Member Data Documentation	1088
6.194.4.1 <code>typelError</code>	1088
6.194.4.2 <code>typellError</code>	1089
6.195 <code>multiscaletest::StatisticalModelCheckerTest</code> Class Reference	1089
6.195.1 Detailed Description	1092
6.195.2 Constructor & Destructor Documentation	1092
6.195.2.1 <code>StatisticalModelCheckerTest</code>	1092
6.195.3 Member Function Documentation	1092
6.195.3.1 <code>InitialiseModelChecker</code>	1092
6.195.3.2 <code>SetTypelError</code>	1092
6.195.3.3 <code>SetTypellError</code>	1093
6.195.4 Member Data Documentation	1093
6.195.4.1 <code>typelError</code>	1093
6.195.4.2 <code>typellError</code>	1093
6.196 <code>multiscale::StringManipulator</code> Class Reference	1093
6.196.1 Detailed Description	1095
6.196.2 Member Function Documentation	1095
6.196.2.1 <code>convert</code>	1095
6.196.2.2 <code>count</code>	1096
6.196.2.3 <code>escapeCarriageReturns</code>	1096
6.196.2.4 <code>filenameFromPath</code>	1096
6.196.2.5 <code>replace</code>	1097
6.196.2.6 <code>split</code>	1097
6.196.2.7 <code>toString</code>	1097
6.196.2.8 <code>trimRight</code>	1098
6.196.2.9 <code>trimRight</code>	1099

6.196.3 Member Data Documentation . . . . .	1099
6.196.3.1 DIR_SEPARATOR . . . . .	1099
6.196.3.2 ERR_INVALID_CONVERSION_BEGIN . . . . .	1099
6.196.3.3 ERR_INVALID_CONVERSION_END . . . . .	1099
6.196.3.4 ERR_INVALID_CONVERSION_MIDDLE . . . . .	1099
6.197 multiscale::verification::SubsetAttribute Class Reference . . . . .	1100
6.197.1 Detailed Description . . . . .	1100
6.197.2 Member Data Documentation . . . . .	1100
6.197.2.1 subset . . . . .	1100
6.198 multiscale::verification::SubsetOperationAttribute Class Reference . . . . .	1100
6.198.1 Detailed Description . . . . .	1101
6.198.2 Member Data Documentation . . . . .	1101
6.198.2.1 subsetOperationType . . . . .	1101
6.199 multiscale::verification::SubsetOperationTypeParser Struct Reference . . . . .	1101
6.199.1 Detailed Description . . . . .	1101
6.199.2 Constructor & Destructor Documentation . . . . .	1101
6.199.2.1 SubsetOperationTypeParser . . . . .	1101
6.200 multiscale::verification::SubsetSpecificAttribute Class Reference . . . . .	1102
6.200.1 Detailed Description . . . . .	1102
6.200.2 Member Data Documentation . . . . .	1102
6.200.2.1 subsetSpecificType . . . . .	1102
6.201 multiscale::verification::SubsetSpecificTypeParser Struct Reference . . . . .	1102
6.201.1 Detailed Description . . . . .	1103
6.201.2 Constructor & Destructor Documentation . . . . .	1103
6.201.2.1 SubsetSpecificTypeParser . . . . .	1103
6.202 multiscale::verification::SubsetSubsetOperationAttribute Class Reference	1103
6.202.1 Detailed Description . . . . .	1104
6.202.2 Member Data Documentation . . . . .	1104
6.202.2.1 firstSubset . . . . .	1104
6.202.2.2 secondSubset . . . . .	1105
6.202.2.3 subsetOperation . . . . .	1105
6.203 multiscale::verification::SubsetVisitor Class Reference . . . . .	1105
6.203.1 Detailed Description . . . . .	1107
6.203.2 Constructor & Destructor Documentation . . . . .	1107

---

6.203.2.1 SubsetVisitor . . . . .	1107
6.203.3 Member Function Documentation . . . . .	1107
6.203.3.1 evaluate . . . . .	1107
6.203.3.2 evaluateSubsetOperation . . . . .	1108
6.203.3.3 filterTimePoint . . . . .	1108
6.203.3.4 operator() . . . . .	1108
6.203.3.5 operator() . . . . .	1109
6.203.3.6 operator() . . . . .	1109
6.203.3.7 operator() . . . . .	1109
6.203.3.8 setTimePointConsideredSpatialEntityType . . . . .	1110
6.203.4 Member Data Documentation . . . . .	1110
6.203.4.1 multiscaleArchitectureGraph . . . . .	1110
6.203.4.2 timePoint . . . . .	1110
6.204 multiscale::SubtractionOperation Class Reference . . . . .	1111
6.204.1 Detailed Description . . . . .	1111
6.204.2 Member Function Documentation . . . . .	1111
6.204.2.1 operator() . . . . .	1111
6.205 multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, - ReferencePointType > Class Template Reference . . . . .	1111
6.205.1 Detailed Description . . . . .	1115
6.205.2 Constructor & Destructor Documentation . . . . .	1115
6.205.2.1 TangentsFromPointToPolygonFinder . . . . .	1115
6.205.3 Member Function Documentation . . . . .	1115
6.205.3.1 computeLeftMostTangentPoint . . . . .	1115
6.205.3.2 computeRightMostTangentPoint . . . . .	1116
6.205.3.3 computeTangentsPointsForConvexPolygon . . . . .	1116
6.205.3.4 computeTangentsPointsWhenReferencePoint- OutsidePolygon . . . . .	1117
6.205.3.5 computeTangentsPointsWithInitialisedAlgorithm- Variables . . . . .	1118
6.205.3.6 initialise . . . . .	1118
6.205.3.7 initialiseAlgorithmVariables . . . . .	1119
6.205.3.8 initialisePolygon . . . . .	1119
6.205.3.9 isFirstPolygonPointTheLeftMostTangentPoint . . . . .	1120
6.205.3.10sFirstPolygonPointTheRightMostTangentPoint . . . . .	1120

6.205.3.11isPointAAbovePointC . . . . .	1120
6.205.3.12isPointABelowPointC . . . . .	1121
6.205.3.13isPointCLeftMostTangentPoint . . . . .	1121
6.205.3.14isPointCRightMostTangentPoint . . . . .	1122
6.205.3.15predecessor . . . . .	1122
6.205.3.16searchForLeftMostTangentPoint . . . . .	1122
6.205.3.17searchForRightMostTangentPoint . . . . .	1123
6.205.3.18setTangentPointsCoordinatesZero . . . . .	1123
6.205.3.19successor . . . . .	1124
6.205.3.20updateCValue . . . . .	1124
6.205.3.21updateEdgeADownFlag . . . . .	1124
6.205.3.22updateEdgeAUpFlag . . . . .	1125
6.205.3.23updateEdgeCFlag . . . . .	1125
6.205.3.24updateLeftMostTangentPointSubChain . . . . .	1126
6.205.3.25updateLeftMostTangentPointSubChainEdgeADown . . . . .	1126
6.205.3.26updateLeftMostTangentPointSubChainEdgeAUp . . . . .	1126
6.205.3.27updateRightMostTangentPointSubChain . . . . .	1127
6.205.3.28updateRightMostTangentPointSubChainEdgeADown . . . . .	1127
6.205.3.29updateRightMostTangentPointSubChainEdgeAUp . . . . .	1128
6.205.4 Member Data Documentation . . . . .	1128
6.205.4.1 a . . . . .	1128
6.205.4.2 b . . . . .	1129
6.205.4.3 c . . . . .	1129
6.205.4.4 isEdgeADown . . . . .	1130
6.205.4.5 isEdgeAUp . . . . .	1130
6.205.4.6 isEdgeCDown . . . . .	1130
6.205.4.7 nrOfPolygonPoints . . . . .	1131
6.205.4.8 polygon . . . . .	1131
6.205.4.9 referencePoint . . . . .	1132
6.206multiscale::verification::TemporalDataReader Class Reference . . . . .	1132
6.206.1 Detailed Description . . . . .	1135
6.206.2 Constructor & Destructor Documentation . . . . .	1135
6.206.2.1 TemporalDataReader . . . . .	1135
6.206.3 Member Function Documentation . . . . .	1135

6.206.3.1 addNumericStateVariablesToTimePoint . . . . .	1135
6.206.3.2 createTimePointFromTokens . . . . .	1136
6.206.3.3 processLineTokens . . . . .	1136
6.206.3.4 readFromValidInputFile . . . . .	1137
6.206.3.5 readFromValidOpenedInputFile . . . . .	1137
6.206.3.6 readInputFileContents . . . . .	1137
6.206.3.7 readInputFileHeader . . . . .	1137
6.206.3.8 readTimeseriesFromFile . . . . .	1138
6.206.3.9 setTimePointValue . . . . .	1138
6.206.3.10 validateObservableVariables . . . . .	1138
6.206.4 Member Data Documentation . . . . .	1139
6.206.4.1 currentLineNumber . . . . .	1139
6.206.4.2 ERR_EMPTY_OBSERVABLE_VARIABLE_NAME . .	1139
6.206.4.3 ERR_INVALID_INPUT_FILE_PATH_BEGIN . . .	1139
6.206.4.4 ERR_INVALID_INPUT_FILE_PATH_END . . . .	1139
6.206.4.5 ERR_INVALID_NR_LINE_TOKENS_BEGIN . . .	1139
6.206.4.6 ERR_INVALID_NR_LINE_TOKENS_END . . . .	1140
6.206.4.7 ERR_INVALID_NR_LINE_TOKENS_MIDDLE . . .	1140
6.206.4.8 ERR_INVALID_NR_OBSERVABLE_VARIABLES_- BEGIN . . . . .	1140
6.206.4.9 ERR_INVALID_NR_OBSERVABLE_VARIABLES_- END . . . . .	1140
6.206.4.10 ERR_OPEN_INPUT_FILE_BEGIN . . . . .	1140
6.206.4.11 ERR_OPEN_INPUT_FILE_END . . . . .	1140
6.206.4.12 filePath . . . . .	1141
6.206.4.13 INPUT_FILE_DELIMITER . . . . .	1141
6.206.4.14 INPUT_FILE_EXTENSION . . . . .	1141
6.206.4.15 observableVariables . . . . .	1141
6.207 multiscale::verification::TemporalNumericCollectionAttribute Class - Reference . . . . .	1141
6.207.1 Detailed Description . . . . .	1142
6.207.2 Member Data Documentation . . . . .	1142
6.207.2.1 temporalNumericCollection . . . . .	1142
6.208 multiscale::verification::TemporalNumericCollectionGrammar< Iterator > Class Template Reference . . . . .	1142

6.208.1 Detailed Description . . . . .	1146
6.208.2 Constructor & Destructor Documentation . . . . .	1146
6.208.2.1 TemporalNumericCollectionGrammar . . . . .	1146
6.208.3 Member Function Documentation . . . . .	1146
6.208.3.1 assignNamesToNumericMeasureRules . . . . .	1146
6.208.3.2 assignNamesToRules . . . . .	1147
6.208.3.3 assignNamesToTemporalNumericCollectionRules . .	1147
6.208.3.4 assignNamesToTimeseriesComponentRules . . . .	1147
6.208.3.5 assignNamesToTimeseriesMeasureRules . . . . .	1147
6.208.3.6 initialise . . . . .	1147
6.208.3.7 initialiseDebugSupport . . . . .	1148
6.208.3.8 initialiseErrorHandlingSupport . . . . .	1148
6.208.3.9 initialiseGrammar . . . . .	1148
6.208.3.10 initialiseNumericMeasureErrorHandlingSupport . .	1148
6.208.3.11 initialiseNumericMeasureRule . . . . .	1148
6.208.3.12 initialiseNumericMeasureRuleDebugging . . . .	1148
6.208.3.13 initialiseRulesDebugging . . . . .	1149
6.208.3.14 initialiseTemporalNumericCollectionErrorHandling- Support . . . . .	1149
6.208.3.15 initialiseTemporalNumericCollectionRule . . . . .	1149
6.208.3.16 initialiseTemporalNumericCollectionRuleDebugging . .	1149
6.208.3.17 initialiseTimeseriesComponentRule . . . . .	1149
6.208.3.18 initialiseTimeseriesComponentRuleDebugging . . . .	1149
6.208.3.19 initialiseTimeseriesMeasureRule . . . . .	1150
6.208.3.20 initialiseTimeseriesMeasureRuleDebugging . . . . .	1150
6.208.4 Member Data Documentation . . . . .	1150
6.208.4.1 binaryNumericMeasureRule . . . . .	1150
6.208.4.2 binaryNumericNumericRule . . . . .	1150
6.208.4.3 changeMeasureRule . . . . .	1150
6.208.4.4 changeTemporalNumericCollectionRule . . . . .	1151
6.208.4.5 heterogeneousTimeseriesComponentRule . . . . .	1151
6.208.4.6 heterogeneousTimeseriesComponentTypeParser . . .	1151
6.208.4.7 homogeneousHomogeneousTimeseriesRule . . . . .	1151
6.208.4.8 homogeneousTimeseriesComponentRule . . . . .	1151

6.208.4.9 homogeneousTimeseriesComponentTypeParser . . . . .	1152
6.208.4.10homogeneousTimeseriesMeasureRule . . . . .	1152
6.208.4.11homogeneousTimeseriesMeasureTypeParser . . . . .	1152
6.208.4.12numericMeasureRule . . . . .	1152
6.208.4.13primaryNumericMeasureRule . . . . .	1152
6.208.4.14spatialMeasureCollectionGrammar . . . . .	1153
6.208.4.15temporalNumericCollectionRule . . . . .	1153
6.208.4.16temporalNumericMeasureCollectionRule . . . . .	1153
6.208.4.17timeseriesComponentRule . . . . .	1153
6.208.4.18timeseriesMeasureRule . . . . .	1153
6.208.4.19timeseriesMeasureTypeParser . . . . .	1154
6.208.4.20timeseriesTimeseriesComponentRule . . . . .	1154
6.208.4.21unaryNumericMeasureRule . . . . .	1154
6.208.4.22unaryNumericNumericRule . . . . .	1154
6.209multiscale::verification::TemporalNumericComparisonAttribute Class Reference . . . . .	1154
6.209.1 Detailed Description . . . . .	1155
6.209.2 Member Data Documentation . . . . .	1155
6.209.2.1 comparator . . . . .	1155
6.209.2.2 lhsTemporalNumericMeasure . . . . .	1156
6.209.2.3 rhsTemporalNumericMeasure . . . . .	1156
6.210multiscale::verification::TemporalNumericMeasureAttribute Class Reference . . . . .	1156
6.210.1 Detailed Description . . . . .	1156
6.210.2 Member Data Documentation . . . . .	1156
6.210.2.1 temporalNumericMeasure . . . . .	1157
6.211multiscale::verification::TemporalNumericMeasureCollectionAttribute Class Reference . . . . .	1157
6.211.1 Detailed Description . . . . .	1157
6.211.2 Member Data Documentation . . . . .	1157
6.211.2.1 endTimepoint . . . . .	1157
6.211.2.2 numericMeasure . . . . .	1157
6.211.2.3 startTimepoint . . . . .	1158
6.212multiscale::verification::TemporalNumericMeasureGrammar< Iterator > Class Template Reference . . . . .	1158

6.212.1 Detailed Description . . . . .	1162
6.212.2 Constructor & Destructor Documentation . . . . .	1162
6.212.2.1 TemporalNumericMeasureGrammar . . . . .	1162
6.212.3 Member Function Documentation . . . . .	1162
6.212.3.1 assignNamesToNumericMeasureCollectionRules . . .	1162
6.212.3.2 assignNamesToNumericStatisticalMeasureRules . .	1163
6.212.3.3 assignNamesToRules . . . . .	1163
6.212.3.4 assignNamesToTemporalNumericMeasureRules . .	1163
6.212.3.5 initialise . . . . .	1163
6.212.3.6 initialiseDebugSupport . . . . .	1163
6.212.3.7 initialiseErrorHandlingSupport . . . . .	1163
6.212.3.8 initialiseGrammar . . . . .	1164
6.212.3.9 initialiseNumericMeasureCollectionRule . . . .	1164
6.212.3.10 initialiseNumericMeasureCollectionRuleDebugging . .	1164
6.212.3.11 initialiseNumericStatisticalMeasureErrorHandling- Support . . . . .	1164
6.212.3.12 initialiseNumericStatisticalMeasureRule . . . . .	1164
6.212.3.13 initialiseNumericStatisticalMeasureRuleDebugging . .	1165
6.212.3.14 initialiseRulesDebugging . . . . .	1165
6.212.3.15 initialiseTemporalNumericMeasureErrorHandling- Support . . . . .	1165
6.212.3.16 initialiseTemporalNumericMeasureRule . . . . .	1165
6.212.3.17 initialiseTemporalNumericMeasureRuleDebugging . .	1165
6.212.4 Member Data Documentation . . . . .	1165
6.212.4.1 binaryNumericMeasureRule . . . . .	1166
6.212.4.2 binaryNumericTemporalRule . . . . .	1166
6.212.4.3 binaryStatisticalMeasureRule . . . . .	1166
6.212.4.4 binaryStatisticalNumericRule . . . . .	1166
6.212.4.5 binaryStatisticalQuantileMeasureRule . . . . .	1166
6.212.4.6 binaryStatisticalQuantileNumericRule . . . . .	1166
6.212.4.7 numericMeasureCollectionRule . . . . .	1167
6.212.4.8 numericStateVariableRule . . . . .	1167
6.212.4.9 numericStatisticalMeasureRule . . . . .	1167
6.212.4.10 spatialMeasureCollectionRule . . . . .	1167

6.212.4.11temporalNumericCollectionRule . . . . .	1167
6.212.4.12temporalNumericMeasureRule . . . . .	1168
6.212.4.13unaryNumericMeasureRule . . . . .	1168
6.212.4.14unaryNumericTemporalRule . . . . .	1168
6.212.4.15unaryStatisticalMeasureRule . . . . .	1168
6.212.4.16unaryStatisticalNumericRule . . . . .	1168
6.213multiscale::verification::TemporalNumericVisitor Class Reference . . . . .	1169
6.213.1 Detailed Description . . . . .	1172
6.213.2 Constructor & Destructor Documentation . . . . .	1172
6.213.2.1 TemporalNumericVisitor . . . . .	1172
6.213.3 Member Function Documentation . . . . .	1172
6.213.3.1 evaluate . . . . .	1172
6.213.3.2 evaluateNumericMeasureCollection . . . . .	1172
6.213.3.3 evaluateNumericStatisticalMeasure . . . . .	1173
6.213.3.4 operator() . . . . .	1173
6.213.3.5 operator() . . . . .	1173
6.213.3.6 operator() . . . . .	1174
6.213.3.7 operator() . . . . .	1174
6.213.3.8 operator() . . . . .	1174
6.213.3.9 operator() . . . . .	1175
6.213.3.10operator() . . . . .	1175
6.213.3.11operator() . . . . .	1176
6.213.3.12operator() . . . . .	1176
6.213.4 Member Data Documentation . . . . .	1176
6.213.4.1 multiscaleArchitectureGraph . . . . .	1177
6.213.4.2 trace . . . . .	1177
6.214multiscale::TestException Class Reference . . . . .	1177
6.214.1 Detailed Description . . . . .	1180
6.214.2 Constructor & Destructor Documentation . . . . .	1180
6.214.2.1 TestException . . . . .	1180
6.214.2.2 TestException . . . . .	1180
6.214.2.3 TestException . . . . .	1180
6.215multiscale::verification::TimePoint Class Reference . . . . .	1180
6.215.1 Detailed Description . . . . .	1184

---

6.215.2 Constructor & Destructor Documentation . . . . .	1185
6.215.2.1 TimePoint . . . . .	1185
6.215.2.2 TimePoint . . . . .	1185
6.215.2.3 ~TimePoint . . . . .	1185
6.215.3 Member Function Documentation . . . . .	1185
6.215.3.1 addConsideredSpatialEntityType . . . . .	1185
6.215.3.2 addNumericStateVariable . . . . .	1185
6.215.3.3 addSpatialEntity . . . . .	1186
6.215.3.4 addSpatialEntityAndType . . . . .	1186
6.215.3.5 areEqualNumericStateVariables . . . . .	1187
6.215.3.6 areEqualSpatialEntities . . . . .	1187
6.215.3.7 areEqualSpatialEntitiesOfSpecificType . . . . .	1187
6.215.3.8 containsNumericStateVariable . . . . .	1188
6.215.3.9 containsNumericStateVariable . . . . .	1188
6.215.3.10containsSpatialEntity . . . . .	1188
6.215.3.11containsSpatialEntity . . . . .	1189
6.215.3.12containsSpatialEntity . . . . .	1189
6.215.3.13containsSpatialEntity . . . . .	1189
6.215.3.14getConsideredSpatialEntities . . . . .	1190
6.215.3.15getConsideredSpatialEntityTypes . . . . .	1190
6.215.3.16getNumericStateVariablesBeginIterator . . . . .	1190
6.215.3.17getNumericStateVariablesBeginIterator . . . . .	1190
6.215.3.18getNumericStateVariablesEndIterator . . . . .	1190
6.215.3.19getNumericStateVariablesEndIterator . . . . .	1191
6.215.3.20getNumericStateVariableValue . . . . .	1191
6.215.3.21getSpatialEntitiesBeginIterator . . . . .	1191
6.215.3.22getSpatialEntitiesBeginIterator . . . . .	1192
6.215.3.23getSpatialEntitiesBeginIterator . . . . .	1192
6.215.3.24getSpatialEntitiesBeginIterator . . . . .	1193
6.215.3.25getSpatialEntitiesEndIterator . . . . .	1193
6.215.3.26getSpatialEntitiesEndIterator . . . . .	1193
6.215.3.27getSpatialEntitiesEndIterator . . . . .	1194
6.215.3.28getSpatialEntitiesEndIterator . . . . .	1194
6.215.3.29getValue . . . . .	1194

6.215.3.30numberOfSpatialEntities . . . . .	1195
6.215.3.31operator!= . . . . .	1195
6.215.3.32operator== . . . . .	1195
6.215.3.33removeSpatialEntity . . . . .	1195
6.215.3.34setConsideredSpatialEntityType . . . . .	1196
6.215.3.35setValue . . . . .	1196
6.215.3.36spatialEntitiesSetOperation . . . . .	1197
6.215.3.37timePointDifference . . . . .	1197
6.215.3.38timePointIntersection . . . . .	1198
6.215.3.39timePointSetOperation . . . . .	1198
6.215.3.40timePointUnion . . . . .	1198
6.215.3.41updateConsideredSpatialEntityTypes . . . . .	1199
6.215.3.42updateSpatialEntities . . . . .	1199
6.215.4 Member Data Documentation . . . . .	1199
6.215.4.1 consideredSpatialEntityTypes . . . . .	1200
6.215.4.2 ERR_GET_NUMERIC_STATE_VARIABLE_PREFIX . . . . .	1200
6.215.4.3 ERR_GET_NUMERIC_STATE_VARIABLE_SUFFIX . . . . .	1200
6.215.4.4 numericStateVariables . . . . .	1200
6.215.4.5 spatialEntities . . . . .	1200
6.215.4.6 value . . . . .	1201
6.216 multiscale::verification::TimePointEvaluator Class Reference . . . . .	1201
6.216.1 Detailed Description . . . . .	1201
6.216.2 Member Function Documentation . . . . .	1202
6.216.2.1 getSpatialMeasureValues . . . . .	1202
6.216.2.2 getSpatialMeasureValues . . . . .	1202
6.217 multiscale::verification::TimeseriesComponentAttribute Class Reference . . . . .	1203
6.217.1 Detailed Description . . . . .	1203
6.217.2 Member Data Documentation . . . . .	1203
6.217.2.1 timeseriesComponent . . . . .	1203
6.218 multiscale::verification::TimeseriesComponentEvaluator Class Reference . . . . .	1203
6.218.1 Detailed Description . . . . .	1204
6.218.2 Member Function Documentation . . . . .	1204
6.218.2.1 evaluate . . . . .	1204
6.218.2.2 evaluateHomogeneousComponentIndices . . . . .	1205

6.219multiscale::verification::TimeseriesComponentVisitor Class Reference . . . . .	1205
6.219.1 Detailed Description . . . . .	1207
6.219.2 Constructor & Destructor Documentation . . . . .	1207
6.219.2.1 TimeseriesComponentVisitor . . . . .	1207
6.219.3 Member Function Documentation . . . . .	1208
6.219.3.1 duplicateCollectionElements . . . . .	1208
6.219.3.2 evaluateHeterogeneousComponentsIndices . . . . .	1208
6.219.3.3 operator() . . . . .	1208
6.219.3.4 operator() . . . . .	1209
6.219.4 Member Data Documentation . . . . .	1209
6.219.4.1 values . . . . .	1209
6.220multiscale::verification::TimeseriesMeasureAttribute Class Reference . . . . .	1209
6.220.1 Detailed Description . . . . .	1210
6.220.2 Member Data Documentation . . . . .	1210
6.220.2.1 timeseriesMeasure . . . . .	1210
6.221multiscale::verification::TimeseriesMeasureTypeParser Struct Reference . . . . .	1210
6.221.1 Detailed Description . . . . .	1210
6.221.2 Constructor & Destructor Documentation . . . . .	1210
6.221.2.1 TimeseriesMeasureTypeParser . . . . .	1210
6.222multiscale::verification::TimeseriesTimeseriesComponentAttribute - Class Reference . . . . .	1211
6.222.1 Detailed Description . . . . .	1211
6.222.2 Member Data Documentation . . . . .	1211
6.222.2.1 temporalNumericMeasureCollection . . . . .	1211
6.222.2.2 timeseriesComponent . . . . .	1212
6.222.2.3 timeseriesMeasure . . . . .	1212
6.223multiscaletest::TraceEvaluationTest Class Reference . . . . .	1212
6.223.1 Detailed Description . . . . .	1216
6.223.2 Constructor & Destructor Documentation . . . . .	1216
6.223.2.1 TraceEvaluationTest . . . . .	1216
6.223.3 Member Function Documentation . . . . .	1216
6.223.3.1 InitialiseMultiscaleArchitectureGraph . . . . .	1216
6.223.3.2 InitialiseQuery . . . . .	1216
6.223.3.3 InitialiseTrace . . . . .	1217

6.223.3.4 RunEvaluationTest . . . . .	1217
6.223.3.5 RunTest . . . . .	1217
6.223.3.6 ValidateTestResults . . . . .	1217
6.223.4 Member Data Documentation . . . . .	1218
6.223.4.1 a.MaxValue . . . . .	1218
6.223.4.2 a.MinValue . . . . .	1218
6.223.4.3 a.NumericStateVariableId . . . . .	1218
6.223.4.4 aWithTypeNumericStateVariableId . . . . .	1218
6.223.4.5 b.ConstantValue . . . . .	1218
6.223.4.6 b.NumericStateVariableId . . . . .	1218
6.223.4.7 bWithTypeNumericStateVariableId . . . . .	1218
6.223.4.8 c.MaxValue . . . . .	1219
6.223.4.9 c.MinValue . . . . .	1219
6.223.4.10 c.NumericStateVariableId . . . . .	1219
6.223.4.11 d.ConstantValue . . . . .	1219
6.223.4.12 d.NumericStateVariableId . . . . .	1219
6.223.4.13 evaluationResult . . . . .	1219
6.223.4.14 multiscaleArchitectureGraph . . . . .	1220
6.223.4.15 nrOfTimePoints . . . . .	1220
6.223.4.16 query . . . . .	1220
6.223.4.17 SCALE_AND_SUBSYSTEM_ORGAN_HEART . . .	1220
6.223.4.18 SCALE_AND_SUBSYSTEM_ORGAN_KIDNEY . . .	1220
6.223.4.19 SCALE_AND_SUBSYSTEM_ORGAN_LIVER . . .	1220
6.223.4.20 SCALE_AND_SUBSYSTEM_ORGANISM_HUMAN .	1221
6.223.4.21 trace . . . . .	1221
6.224 multiscale::verification::UnaryNumericFilterAttribute Class Reference .	1221
6.224.1 Detailed Description . . . . .	1222
6.224.2 Member Data Documentation . . . . .	1222
6.224.2.1 filterNumericMeasure . . . . .	1222
6.224.2.2 unaryNumericMeasure . . . . .	1223
6.225 multiscale::verification::UnaryNumericMeasureAttribute Class Reference	1223
6.225.1 Detailed Description . . . . .	1223
6.225.2 Member Data Documentation . . . . .	1223
6.225.2.1 unaryNumericMeasureType . . . . .	1223

6.226	multiscale::verification::UnaryNumericMeasureGrammar< Iterator > -	
	Class Template Reference . . . . .	1224
6.226.1	Detailed Description . . . . .	1225
6.226.2	Constructor & Destructor Documentation . . . . .	1226
6.226.2.1	UnaryNumericMeasureGrammar . . . . .	1226
6.226.3	Member Function Documentation . . . . .	1226
6.226.3.1	assignNamesToRules . . . . .	1226
6.226.3.2	initialise . . . . .	1226
6.226.3.3	initialiseDebugSupport . . . . .	1226
6.226.3.4	initialiseGrammar . . . . .	1226
6.226.3.5	initialiseRulesDebugging . . . . .	1227
6.226.4	Member Data Documentation . . . . .	1227
6.226.4.1	unaryNumericMeasureRule . . . . .	1227
6.226.4.2	unaryNumericMeasureTypeParser . . . . .	1227
6.227	multiscale::verification::UnaryNumericMeasureTypeParser Struct -	
	Reference . . . . .	1227
6.227.1	Detailed Description . . . . .	1227
6.227.2	Constructor & Destructor Documentation . . . . .	1228
6.227.2.1	UnaryNumericMeasureTypeParser . . . . .	1228
6.228	multiscale::verification::UnaryNumericNumericAttribute Class Reference	1228
6.228.1	Detailed Description . . . . .	1229
6.228.2	Member Data Documentation . . . . .	1229
6.228.2.1	numericMeasure . . . . .	1229
6.228.2.2	unaryNumericMeasure . . . . .	1230
6.229	multiscale::verification::UnaryNumericSpatialAttribute Class Reference	1230
6.229.1	Detailed Description . . . . .	1230
6.229.2	Member Data Documentation . . . . .	1231
6.229.2.1	spatialMeasureCollection . . . . .	1231
6.229.2.2	unaryNumericMeasure . . . . .	1231
6.230	multiscale::verification::UnaryNumericTemporalAttribute Class Reference	1231
6.230.1	Detailed Description . . . . .	1232
6.230.2	Member Data Documentation . . . . .	1232
6.230.2.1	temporalNumericMeasure . . . . .	1232
6.230.2.2	unaryNumericMeasure . . . . .	1233

6.231 multiscale::verification::UnaryScaleAndSubsystemConstraintAttribute	
Class Reference . . . . .	1233
6.231.1 Detailed Description . . . . .	1234
6.231.2 Member Data Documentation . . . . .	1234
6.231.2.1 comparator . . . . .	1234
6.231.2.2 scaleAndSubsystem . . . . .	1234
6.232 multiscale::verification::UnarySpatialConstraintAttribute	Class Reference
6.232.1 Detailed Description . . . . .	1234
6.232.2 Member Data Documentation . . . . .	1235
6.232.2.1 comparator . . . . .	1235
6.232.2.2 filterNumericMeasure . . . . .	1235
6.232.2.3 spatialMeasure . . . . .	1236
6.233 multiscale::verification::UnaryStatisticalMeasureAttribute	Class Reference
6.233.1 Detailed Description . . . . .	1236
6.233.2 Member Data Documentation . . . . .	1236
6.233.2.1 unaryStatisticalMeasureType . . . . .	1236
6.234 multiscale::verification::UnaryStatisticalMeasureGrammar< Iterator > -	Class Template Reference
6.234.1 Detailed Description . . . . .	1237
6.234.2 Constructor & Destructor Documentation . . . . .	1238
6.234.2.1 UnaryStatisticalMeasureGrammar . . . . .	1239
6.234.3 Member Function Documentation . . . . .	1239
6.234.3.1 assignNamesToRules . . . . .	1239
6.234.3.2 initialise . . . . .	1239
6.234.3.3 initialiseDebugSupport . . . . .	1239
6.234.3.4 initialiseGrammar . . . . .	1239
6.234.3.5 initialiseRulesDebugging . . . . .	1240
6.234.4 Member Data Documentation . . . . .	1240
6.234.4.1 unaryStatisticalMeasureRule . . . . .	1240
6.234.4.2 unaryStatisticalMeasureTypeParser . . . . .	1240
6.235 multiscale::verification::UnaryStatisticalMeasureTypeParser	Struct - Reference
6.235.1 Detailed Description . . . . .	1240
6.235.2 Constructor & Destructor Documentation . . . . .	1241
6.235.2.1 UnaryStatisticalMeasureTypeParser . . . . .	1241

6.236multiscale::verification::UnaryStatisticalNumericAttribute Class Reference	1241
6.236.1 Detailed Description	1242
6.236.2 Member Data Documentation	1242
6.236.2.1 numericMeasureCollection	1242
6.236.2.2 unaryStatisticalMeasure	1242
6.237multiscale::verification::UnaryStatisticalSpatialAttribute Class Reference	1242
6.237.1 Detailed Description	1243
6.237.2 Member Data Documentation	1243
6.237.2.1 spatialMeasureCollection	1243
6.237.2.2 unaryStatisticalMeasure	1243
6.238multiscale::UnexpectedBehaviourException Class Reference	1244
6.238.1 Detailed Description	1246
6.238.2 Constructor & Destructor Documentation	1246
6.238.2.1 UnexpectedBehaviourException	1246
6.238.2.2 UnexpectedBehaviourException	1246
6.238.2.3 UnexpectedBehaviourException	1246
6.239multiscale::verification::UnexpectedErrorHandler Struct Reference	1246
6.239.1 Detailed Description	1247
6.239.2 Member Function Documentation	1247
6.239.2.1 getExpectedTokenAsString	1247
6.239.2.2 operator()	1247
6.240multiscale::verification::TimeseriesComponentEvaluator::Uniform-HomogeneousComponentEvaluator< Relation > Class Template - Reference	1248
6.240.1 Detailed Description	1248
6.240.2 Member Function Documentation	1249
6.240.2.1 computeEndIndex	1249
6.240.2.2 hasValidSuccessors	1249
6.240.2.3 startIndex	1250
6.241multiscale::UnimplementedMethodException Class Reference	1250
6.241.1 Detailed Description	1253
6.241.2 Constructor & Destructor Documentation	1253
6.241.2.1 UnimplementedMethodException	1253
6.241.2.2 UnimplementedMethodException	1253

6.241.2.3 UnimplementedMethodException . . . . .	1253
6.242 multiscale::verification::UntilLogicPropertyAttribute Class Reference . . .	1253
6.242.1 Detailed Description . . . . .	1254
6.242.2 Member Data Documentation . . . . .	1254
6.242.2.1 endTimepoint . . . . .	1254
6.242.2.2 logicProperty . . . . .	1254
6.242.2.3 startTimepoint . . . . .	1254
6.243 multiscale::UserDefinedTypeName< T > Class Template Reference . .	1254
6.243.1 Detailed Description . . . . .	1255
6.243.2 Member Function Documentation . . . . .	1255
6.243.2.1 name . . . . .	1255
6.244 multiscale::XmlValidator::XmlValidationErrorHandler Class Reference .	1255
6.244.1 Detailed Description . . . . .	1257
6.244.2 Member Function Documentation . . . . .	1257
6.244.2.1 constructExceptionMessage . . . . .	1257
6.244.2.2 error . . . . .	1258
6.244.2.3 fatalError . . . . .	1258
6.244.2.4 handleValidationException . . . . .	1258
6.244.2.5 resetErrors . . . . .	1258
6.244.2.6 warning . . . . .	1259
6.244.3 Member Data Documentation . . . . .	1259
6.244.3.1 ERR_EXCEPTION_BEGIN_MSG . . . . .	1259
6.244.3.2 ERR_EXCEPTION_COLUMN_MSG . . . . .	1259
6.244.3.3 ERR_EXCEPTION_END_MSG . . . . .	1259
6.244.3.4 ERR_EXCEPTION_LINE_MSG . . . . .	1259
6.244.3.5 ERR_EXCEPTION_MIDDLE_MSG . . . . .	1259
6.245 multiscale::XmlValidator Class Reference . . . . .	1260
6.245.1 Detailed Description . . . . .	1263
6.245.2 Member Function Documentation . . . . .	1263
6.245.2.1 checkIfValidXmlFile . . . . .	1263
6.245.2.2 configureParser . . . . .	1263
6.245.2.3 isValidXmlFile . . . . .	1263
6.245.2.4 isValidXmlPathAndFile . . . . .	1264
6.245.2.5 isValidXmlPathAndFile . . . . .	1264

---

6.245.2.6 loadParserSchema . . . . .	1265
6.245.2.7 validateXmlFilepath . . . . .	1265
6.245.2.8 validateXmlSchemaPath . . . . .	1265
6.245.2.9 verifyIfValidXmlFile . . . . .	1266
6.245.3 Member Data Documentation . . . . .	1266
6.245.3.1 ERR_INVALID_SCHEMA_FILEPATH . . . . .	1266
6.245.3.2 ERR_INVALID_XML_FILEPATH . . . . .	1266
6.245.3.3 ERR_SCHEMA_CONTENTS . . . . .	1266



# **Chapter 1**

## **Multiscale**

### **1.1 Brief description**

The "Multiscale" software is a multiscale model checker implemented during the Ph-D research project carried out by Ovidiu Parvu, Brunel University, London, United - Kingdom, October 2012 - present.

### **1.2 Contact**

For more information, comments, recommendations or suggestions please visit the author's [institutional web page](#), where contact details are provided.



## Chapter 2

# Namespace Index

### 2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

<code>multiscale</code> . . . . .	23
<code>multiscale::analysis</code> . . . . .	28
<code>multiscale::verification</code> . . . . .	30
<code>multiscale::verification::spatialmeasure</code> . . . . .	58
<code>multiscale::verification::subsetspecific</code> . . . . .	61
<code>multiscale::visualisation</code> . . . . .	64
<code>multiscaletest</code> . . . . .	65
<code>multiscaletest::verification</code> . . . . .	66



# Chapter 3

## Class Index

### 3.1 Class Hierarchy

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

multiscale::verification::AbstractSyntaxTree . . . . .	67
multiscale::AdditionOperation . . . . .	71
multiscale::verification::AndConstraintAttribute . . . . .	75
multiscale::verification::AndLogicPropertyAttribute . . . . .	76
multiscale::visualisation::AnnularSector . . . . .	77
multiscale::verification::BinaryNumericFilterAttribute . . . . .	150
multiscale::verification::BinaryNumericMeasureAttribute . . . . .	152
multiscale::verification::BinaryNumericMeasureTypeParser . . . . .	157
multiscale::verification::BinaryNumericNumericAttribute . . . . .	158
multiscale::verification::BinaryNumericSpatialAttribute . . . . .	160
multiscale::verification::BinaryNumericTemporalAttribute . . . . .	161
multiscale::verification::BinaryStatisticalMeasureAttribute . . . . .	163
multiscale::verification::BinaryStatisticalMeasureTypeParser . . . . .	168
multiscale::verification::BinaryStatisticalNumericAttribute . . . . .	169
multiscale::verification::BinaryStatisticalQuantileMeasureAttribute . . . . .	170
multiscale::verification::BinaryStatisticalQuantileMeasureTypeParser . . . . .	175
multiscale::verification::BinaryStatisticalQuantileNumericAttribute . . . . .	176
multiscale::verification::BinaryStatisticalQuantileSpatialAttribute . . . . .	177
multiscale::verification::BinaryStatisticalSpatialAttribute . . . . .	179
multiscale::visualisation::CartesianToConcentrationsConverter . . . . .	186
multiscale::visualisation::CartesianToPolarConverter . . . . .	193
multiscale::verification::ChangeMeasureAttribute . . . . .	200
multiscale::verification::ChangeMeasureEvaluator . . . . .	201
multiscale::verification::ChangeMeasureTypeParser . . . . .	208
multiscale::verification::ChangeTemporalNumericCollectionAttribute . . . . .	209
multiscale::verification::ChangeTemporalNumericMeasureAttribute . . . . .	210
multiscale::analysis::CircularityMeasure< PointType > . . . . .	212
multiscale::verification::CommandLineModelChecking . . . . .	246
multiscale::verification::ComparatorAttribute . . . . .	290

multiscale::verification::ComparatorEvaluator . . . . .	291
multiscale::verification::ComparatorNonEqualTypeParser . . . . .	297
multiscale::verification::ComparatorTypeParser . . . . .	298
multiscale::ConsolePrinter . . . . .	303
multiscale::verification::ConstraintAttribute . . . . .	313
multiscale::verification::ConstraintVisitor . . . . .	315
multiscale::analysis::DBSCAN< PointType > . . . . .	329
multiscale::analysis::Detector . . . . .	341
multiscale::analysis::ClusterDetector . . . . .	232
multiscale::analysis::SimulationClusterDetector . . . . .	940
multiscale::analysis::RegionDetector . . . . .	879
multiscale::Distribution . . . . .	367
multiscale::BetaDistribution . . . . .	146
multiscale::BinomialDistribution . . . . .	180
multiscale::DivisionOperation . . . . .	370
multiscale::analysis::Entity . . . . .	374
multiscale::verification::EquivalenceConstraintAttribute . . . . .	380
multiscale::verification::EquivalenceLogicPropertyAttribute . . . . .	381
EuclideanDataPoint . . . . .	382
multiscale::ExceptionHandler . . . . .	383
multiscale::Filesystem . . . . .	388
multiscale::verification::FilterNumericMeasureAttribute . . . . .	393
multiscale::verification::FilterNumericVisitor . . . . .	393
multiscale::verification::FilterSubsetAttribute . . . . .	398
multiscale::verification::FutureLogicPropertyAttribute . . . . .	400
multiscale::Geometry2D . . . . .	401
multiscale::verification::GlobalLogicPropertyAttribute . . . . .	423
grammar . . . . .	424
multiscale::verification::BinaryNumericMeasureGrammar< Iterator > . . . . .	153
multiscale::verification::BinaryStatisticalMeasureGrammar< Iterator > . . . . .	164
multiscale::verification::BinaryStatisticalQuantileMeasureGrammar< - Iterator > . . . . .	171
multiscale::verification::ChangeMeasureGrammar< Iterator > . . . . .	205
multiscale::verification::ComparatorGrammar< Iterator > . . . . .	293
multiscale::verification::LogicPropertyGrammar< Iterator > . . . . .	462
multiscale::verification::NumericStateVariableGrammar< Iterator > . . . . .	732
multiscale::verification::PrimaryNumericMeasureGrammar< Iterator > . . . . .	801
multiscale::verification::ScaleAndSubsystemGrammar< Iterator > . . . . .	923
multiscale::verification::ScaleAndSubsystemStringGrammar< Iterator > . . . . .	928
multiscale::verification::SpatialMeasureCollectionGrammar< Iterator > . . . . .	986
multiscale::verification::TemporalNumericCollectionGrammar< Iterator > . . . . .	1142
multiscale::verification::TemporalNumericMeasureGrammar< Iterator > . . . . .	1158
multiscale::verification::UnaryNumericMeasureGrammar< Iterator > . . . . .	1224
multiscale::verification::UnaryStatisticalMeasureGrammar< Iterator > . . . . .	1237
multiscale::verification::HeterogeneousTimeseriesComponentAttribute . . . . .	425
multiscale::verification::HeterogeneousTimeseriesComponentTypeParser . . . . .	425
multiscale::verification::TimeseriesComponentEvaluator::Homogeneous- ComponentEvaluator< Relation > . . . . .	426
multiscale::verification::HomogeneousHomogeneousTimeseriesAttribute . . . . .	428

multiscale::verification::HomogeneousTimeseriesComponentAttribute . . . . .	430
multiscale::verification::HomogeneousTimeseriesComponentTypeParser . . . . .	431
multiscale::verification::HomogeneousTimeseriesMeasureAttribute . . . . .	431
multiscale::verification::HomogeneousTimeseriesMeasureTypeParser . . . . .	432
multiscale::verification::ImplicationConstraintAttribute . . . . .	433
multiscale::verification::ImplicationLogicPropertyAttribute . . . . .	433
multiscale::verification::LogicPropertyAttribute . . . . .	453
multiscale::verification::LogicPropertyDataReader . . . . .	456
multiscale::verification::LogicPropertyVisitor . . . . .	477
multiscale::analysis::MatFactory . . . . .	499
multiscale::analysis::CircularMatFactory . . . . .	213
multiscale::analysis::RectangularMatFactory . . . . .	859
multiscale::MinEnclosingTriangleFinder . . . . .	507
multiscale::verification::ModelChecker . . . . .	540
multiscale::verification::ApproximateBayesianModelChecker . . . . .	80
multiscale::verification::ApproximateProbabilisticModelChecker . . . . .	103
multiscale::verification::BayesianModelChecker . . . . .	121
multiscale::verification::ProbabilisticBlackBoxModelChecker . . . . .	812
multiscale::verification::StatisticalModelChecker . . . . .	1069
multiscale::verification::ModelCheckerFactory . . . . .	554
multiscale::verification::ApproximateBayesianModelCheckerFactory . . . . .	95
multiscale::verification::ApproximateProbabilisticModelCheckerFactory . . . . .	113
multiscale::verification::BayesianModelCheckerFactory . . . . .	137
multiscale::verification::ProbabilisticBlackBoxModelCheckerFactory . . . . .	817
multiscale::verification::StatisticalModelCheckerFactory . . . . .	1086
multiscale::verification::ModelCheckingManager . . . . .	568
multiscale::verification::ModelCheckingOutputWriter . . . . .	586
multiscale::verification::MSTMLSubfilesMerger . . . . .	611
multiscale::MultiplicationOperation . . . . .	629
multiscale::verification::MultiscaleArchitectureGraph . . . . .	630
multiscale::MultiscaleException . . . . .	660
multiscale::AlgorithmException . . . . .	72
multiscale::UnexpectedBehaviourException . . . . .	1244
multiscale::verification::ModelCheckingException . . . . .	561
multiscale::verification::ModelCheckingHelpRequestException . . . . .	565
multiscale::verification::SpatialTemporalException . . . . .	1039
multiscale::UnimplementedMethodException . . . . .	1250
multiscale::IOException . . . . .	443
multiscale::FileOpenException . . . . .	385
multiscale::InvalidInputException . . . . .	437
multiscale::InvalidOutputException . . . . .	440
multiscale::NumericException . . . . .	708
multiscale::RuntimeException . . . . .	913
multiscale::IndexOutOfBoundsException . . . . .	434
multiscale::TestException . . . . .	1177
multiscaletest::MultiscaleTest . . . . .	664
multiscaletest::MinEnclosingTriangleFinderTest . . . . .	532
multiscaletest::ModelCheckerTest . . . . .	555

multiscaletest::ApproximateBayesianModelCheckerTest . . . . .	98
multiscaletest::ApproximateProbabilisticModelCheckerTest . . . . .	117
multiscaletest::BayesianModelCheckerTest . . . . .	141
multiscaletest::ProbabilisticBlackBoxModelCheckerTest . . . . .	819
multiscaletest::StatisticalModelCheckerTest . . . . .	1089
multiscaletest::TraceEvaluationTest . . . . .	1212
multiscaletest::CompleteTraceTest . . . . .	299
multiscaletest::EmptyTraceTest . . . . .	371
multiscaletest::NumericStateVariableTraceTest . . . . .	742
multiscaletest::SpatialEntitiesTraceTest . . . . .	950
multiscale::verification::NextKLogicPropertyAttribute . . . . .	666
multiscale::verification::NextLogicPropertyAttribute . . . . .	667
multiscale::verification::Nil . . . . .	668
multiscale::verification::NotConstraintAttribute . . . . .	668
multiscale::verification::NotLogicPropertyAttribute . . . . .	669
multiscale::NumberIterator . . . . .	669
multiscale::LexicographicNumberIterator . . . . .	446
multiscale::StandardNumberIterator . . . . .	1059
multiscale::Numeric . . . . .	674
multiscale::verification::NumericEvaluator . . . . .	705
multiscale::verification::NumericMeasureAttribute . . . . .	711
multiscale::verification::NumericMeasureCollectionAttribute . . . . .	712
multiscale::verification::NumericMeasureCollectionEvaluator . . . . .	713
multiscale::verification::NumericMeasureCollectionVisitor . . . . .	715
multiscale::NumericRangeManipulator . . . . .	726
multiscale::verification::NumericSpatialMeasureAttribute . . . . .	728
multiscale::verification::NumericStateVariableAttribute . . . . .	728
multiscale::verification::NumericStateVariableEvaluator . . . . .	730
multiscale::verification::NumericStatisticalMeasureAttribute . . . . .	745
multiscale::verification::NumericVisitor . . . . .	746
multiscale::OperatingSystem . . . . .	754
multiscale::verification::OrConstraintAttribute . . . . .	758
multiscale::verification::OrLogicPropertyAttribute . . . . .	759
multiscale::verification::Parser . . . . .	759
multiscale::verification::ParserGrammarExceptionHandler . . . . .	764
multiscale::verification::ParserGrammarExtraInputException . . . . .	767
multiscale::verification::ParserGrammarProbabilityException . . . . .	769
multiscale::verification::ParserGrammarUnexpectedTokenException . . . . .	772
multiscale::verification::ParserGrammarUnparseableInputException . . . . .	775
multiscale::visualisation::PolarCsvToInputFilesConverter . . . . .	777
multiscale::visualisation::PolarGnuplotScriptGenerator . . . . .	791
multiscale::verification::PrimaryConstraintAttribute . . . . .	799
multiscale::verification::PrimaryLogicPropertyAttribute . . . . .	799
multiscale::verification::PrimaryNumericMeasureAttribute . . . . .	800
multiscale::verification::PrimarySpatialMeasureCollectionAttribute . . . . .	810
multiscale::verification::ProbabilisticLogicPropertyAttribute . . . . .	822
multiscale::verification::ProbabilityErrorHandler . . . . .	826
multiscale::visualisation::RectangularCsvToInputFilesConverter . . . . .	827
multiscale::visualisation::RectangularEntityCsvToInputFilesConverter . . . . .	840

multiscale::visualisation::RectangularGnuplotScriptGenerator . . . . .	851
multiscale::verification::ProbabilityErrorHandler::result< typename, type- name, typename > . . . . .	908
multiscale::verification::UnexpectedTokenErrorHandler::result< typename, typename, typename > . . . . .	909
multiscale::RGBColourGenerator . . . . .	910
multiscale::verification::ScaleAndSubsystem . . . . .	916
multiscale::verification::ScaleAndSubsystemAttribute . . . . .	918
multiscale::verification::ScaleAndSubsystemEvaluator . . . . .	920
multiscale::analysis::Silhouette . . . . .	933
multiscale::verification::SimilarityMeasureAttribute . . . . .	937
multiscale::verification::SimilarityMeasureTypeParser . . . . .	938
multiscale::verification::SimilarityTemporalNumericCollectionAttribute . . . . .	939
multiscale::analysis::SpatialEntityPseudo3D . . . . .	961
multiscale::analysis::Cluster . . . . .	218
multiscale::analysis::Region . . . . .	867
multiscale::verification::SpatialMeasureAttribute . . . . .	979
multiscale::analysis::SpatialMeasureCalculator . . . . .	979
multiscale::verification::SpatialMeasureCollectionAttribute . . . . .	986
multiscale::verification::SpatialMeasureCollectionVisitor . . . . .	1004
multiscale::verification::SpatialMeasureEvaluator . . . . .	1009
multiscale::verification::SpatialMeasureTypeParser . . . . .	1010
multiscale::verification::SpatialNumericComparisonAttribute . . . . .	1011
multiscale::verification::SpatialTemporalDataReader . . . . .	1012
multiscale::verification::SpatialTemporalDataWriter . . . . .	1029
multiscale::verification::SpatialTemporalTrace . . . . .	1043
multiscale::verification::StateVariable . . . . .	1063
multiscale::verification::NumericStateVariableId . . . . .	737
multiscale::verification::SpatialEntity . . . . .	953
multiscale::verification::Cluster . . . . .	229
multiscale::verification::Region . . . . .	864
multiscale::verification::StateVariableAttribute . . . . .	1067
multiscale::StringManipulator . . . . .	1093
multiscale::verification::SubsetAttribute . . . . .	1100
multiscale::verification::SubsetOperationAttribute . . . . .	1100
multiscale::verification::SubsetOperationTypeParser . . . . .	1101
multiscale::verification::SubsetSpecificAttribute . . . . .	1102
multiscale::verification::SubsetSpecificTypeParser . . . . .	1102
multiscale::verification::SubsetSubsetOperationAttribute . . . . .	1103
multiscale::verification::SubsetVisitor . . . . .	1105
multiscale::SubtractionOperation . . . . .	1111
multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType > . . . . .	1111
multiscale::verification::TemporalDataReader . . . . .	1132
multiscale::verification::TemporalNumericCollectionAttribute . . . . .	1141
multiscale::verification::TemporalNumericComparisonAttribute . . . . .	1154
multiscale::verification::TemporalNumericMeasureAttribute . . . . .	1156
multiscale::verification::TemporalNumericMeasureCollectionAttribute . . . . .	1157
multiscale::verification::TemporalNumericVisitor . . . . .	1169

multiscale::verification::TimePoint . . . . .	1180
multiscale::verification::TimePointEvaluator . . . . .	1201
multiscale::verification::TimeseriesComponentAttribute . . . . .	1203
multiscale::verification::TimeseriesComponentEvaluator . . . . .	1203
multiscale::verification::TimeseriesComponentVisitor . . . . .	1205
multiscale::verification::TimeseriesMeasureAttribute . . . . .	1209
multiscale::verification::TimeseriesMeasureTypeParser . . . . .	1210
multiscale::verification::TimeseriesTimeseriesComponentAttribute . . . . .	1211
multiscale::verification::UnaryNumericFilterAttribute . . . . .	1221
multiscale::verification::UnaryNumericMeasureAttribute . . . . .	1223
multiscale::verification::UnaryNumericMeasureTypeParser . . . . .	1227
multiscale::verification::UnaryNumericNumericAttribute . . . . .	1228
multiscale::verification::UnaryNumericSpatialAttribute . . . . .	1230
multiscale::verification::UnaryNumericTemporalAttribute . . . . .	1231
multiscale::verification::UnaryScaleAndSubsystemConstraintAttribute . . . . .	1233
multiscale::verification::UnarySpatialConstraintAttribute . . . . .	1234
multiscale::verification::UnaryStatisticalMeasureAttribute . . . . .	1236
multiscale::verification::UnaryStatisticalMeasureTypeParser . . . . .	1240
multiscale::verification::UnaryStatisticalNumericAttribute . . . . .	1241
multiscale::verification::UnaryStatisticalSpatialAttribute . . . . .	1242
multiscale::verification::UnexpectedTokenErrorHandler . . . . .	1246
multiscale::verification::TimeseriesComponentEvaluator::UniformHomogeneous- ComponentEvaluator< Relation > . . . . .	1248
multiscale::verification::UntilLogicPropertyAttribute . . . . .	1253
multiscale::UserDefinedTypeName< T > . . . . .	1254
multiscale::XmlValidator::XmlValidationErrorHandler . . . . .	1255
multiscale::XmlValidator . . . . .	1260

## Chapter 4

# Class Index

### 4.1 Class List

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

<code>multiscale::verification::AbstractSyntaxTree</code>	Class used for representing an abstract syntax tree . . . . .	67
<code>multiscale::AdditionOperation</code>	Functor representing an addition operation . . . . .	71
<code>multiscale::AlgorithmException</code>	Class for representing algorithm exceptions . . . . .	72
<code>multiscale::verification::AndConstraintAttribute</code>	Class for representing an "and" constraint attribute . . . . .	75
<code>multiscale::verification::AndLogicPropertyAttribute</code>	Class for representing an "and" logic property attribute . . . . .	76
<code>multiscale::visualisation::AnnularSector</code>	An annular sector is the basic element in the considered circular geometry . . . . .	77
<code>multiscale::verification::ApproximateBayesianModelChecker</code>	Class used to run approximate Bayesian model checking tasks . . . . .	80
<code>multiscale::verification::ApproximateBayesianModelCheckerFactory</code>	Class for creating <code>ApproximateBayesianModelChecker</code> instances . . . . .	95
<code>multiscaletest::ApproximateBayesianModelCheckerTest</code>	Class for testing the approximate Bayesian model checker . . . . .	98
<code>multiscale::verification::ApproximateProbabilisticModelChecker</code>	Class used to run approximate probabilistic model checking tasks . . . . .	103
<code>multiscale::verification::ApproximateProbabilisticModelCheckerFactory</code>	Class for creating <code>ApproximateProbabilisticModelChecker</code> instances . . . . .	113
<code>multiscaletest::ApproximateProbabilisticModelCheckerTest</code>	Class for testing the approximate probabilistic model checker . . . . .	117
<code>multiscale::verification::BayesianModelChecker</code>	Class used to run Bayesian model checking tasks . . . . .	121
<code>multiscale::verification::BayesianModelCheckerFactory</code>	Class for creating <code>BayesianModelChecker</code> instances . . . . .	137

<b>multiscaletest::BayesianModelCheckerTest</b>	
Class for testing the Bayesian model checker . . . . .	141
<b>multiscale::BetaDistribution</b>	
Class for analysing Beta distributed data . . . . .	146
<b>multiscale::verification::BinaryNumericFilterAttribute</b>	
Class for representing a binary numeric filter attribute . . . . .	150
<b>multiscale::verification::BinaryNumericMeasureAttribute</b>	
Class for representing a binary numeric measure attribute . . . . .	152
<b>multiscale::verification::BinaryNumericMeasureGrammar&lt; Iterator &gt;</b>	
The grammar for parsing binary numeric measure statements . . . . .	153
<b>multiscale::verification::BinaryNumericMeasureTypeParser</b>	
Symbol table and parser for the binary numeric measure type . . . . .	157
<b>multiscale::verification::BinaryNumericNumericAttribute</b>	
Class for representing a binary numeric numeric measure attribute . . . . .	158
<b>multiscale::verification::BinaryNumericSpatialAttribute</b>	
Class for representing a binary numeric spatial measure collection attribute . . . . .	160
<b>multiscale::verification::BinaryNumericTemporalAttribute</b>	
Class for representing a binary numeric temporal measure attribute . . . . .	161
<b>multiscale::verification::BinaryStatisticalMeasureAttribute</b>	
Class for representing a binary statistical measure attribute . . . . .	163
<b>multiscale::verification::BinaryStatisticalMeasureGrammar&lt; Iterator &gt;</b>	
The grammar for parsing binary statistical measure statements . . . . .	164
<b>multiscale::verification::BinaryStatisticalMeasureTypeParser</b>	
Symbol table and parser for the binary statistical measure type . . . . .	168
<b>multiscale::verification::BinaryStatisticalNumericAttribute</b>	
Class for representing a binary statistical numeric attribute . . . . .	169
<b>multiscale::verification::BinaryStatisticalQuantileMeasureAttribute</b>	
Class for representing a binary statistical quantile measure attribute . . . . .	170
<b>multiscale::verification::BinaryStatisticalQuantileMeasureGrammar&lt; Iterator &gt;</b>	
The grammar for parsing binary statistical quantile measure statements . . . . .	171
<b>multiscale::verification::BinaryStatisticalQuantileMeasureTypeParser</b>	
Symbol table and parser for the binary statistical quantile measure type . . . . .	175
<b>multiscale::verification::BinaryStatisticalQuantileNumericAttribute</b>	
Class for representing a binary statistical quantile numeric attribute . . . . .	176
<b>multiscale::verification::BinaryStatisticalQuantileSpatialAttribute</b>	
Class for representing a binary statistical quantile spatial attribute . . . . .	177
<b>multiscale::verification::BinaryStatisticalSpatialAttribute</b>	
Class for representing a binary statistical spatial attribute . . . . .	179
<b>multiscale::BinomialDistribution</b>	
Class for analysing Binomial distributed data . . . . .	180
<b>multiscale::visualisation::CartesianToConcentrationsConverter</b>	
Scale the values of the rectangular geometry grid cells . . . . .	186
<b>multiscale::visualisation::CartesianToPolarConverter</b>	
Converter from the rectangular geometry grid cells to annular sectors . . . . .	193
<b>multiscale::verification::ChangeMeasureAttribute</b>	
Class for representing a change measure attribute . . . . .	200

<a href="#">multiscale::verification::ChangeMeasureEvaluator</a>	Class for evaluating change measure expressions . . . . .	201
<a href="#">multiscale::verification::ChangeMeasureGrammar&lt; Iterator &gt;</a>	The grammar for parsing change measure statements . . . . .	205
<a href="#">multiscale::verification::ChangeMeasureTypeParser</a>	Symbol table and parser for the change measure type . . . . .	208
<a href="#">multiscale::verification::ChangeTemporalNumericCollectionAttribute</a>	Class for representing a change temporal numeric collection attribute	209
<a href="#">multiscale::verification::ChangeTemporalNumericMeasureAttribute</a>	Class for representing a change temporal numeric measure attribute	210
<a href="#">multiscale::analysis::CircularityMeasure&lt; PointType &gt;</a>	Class for computing the circularity measure for the given collection of points . . . . .	212
<a href="#">multiscale::analysis::CircularMatFactory</a>	Class for creating a Mat object considering a circular grid . . . . .	213
<a href="#">multiscale::analysis::Cluster</a>	Class for representing a cluster of entities in an image . . . . .	218
<a href="#">multiscale::verification::Cluster</a>	Class for representing a cluster . . . . .	229
<a href="#">multiscale::analysis::ClusterDetector</a>	Class for detecting clusters in 2D images . . . . .	232
<a href="#">multiscale::verification::CommandLineModelChecking</a>	Class for running model checkers from the command line . . . . .	246
<a href="#">multiscale::verification::ComparatorAttribute</a>	Class for representing a comparator attribute . . . . .	290
<a href="#">multiscale::verification::ComparatorEvaluator</a>	Class for evaluating comparison expressions . . . . .	291
<a href="#">multiscale::verification::ComparatorGrammar&lt; Iterator &gt;</a>	The grammar for parsing comparator statements . . . . .	293
<a href="#">multiscale::verification::ComparatorNonEqualTypeParser</a>	Symbol table and parser for the comparator type which does not accept the "=" symbol . . . . .	297
<a href="#">multiscale::verification::ComparatorTypeParser</a>	Symbol table and parser for the comparator type . . . . .	298
<a href="#">multiscaletest::CompleteTraceTest</a>	Class for testing evaluation of complete traces containing both numeric state variables and spatial entities . . . . .	299
<a href="#">multiscale::ConsolePrinter</a>	Class used to print (coloured) messages to the console . . . . .	303
<a href="#">multiscale::verification::ConstraintAttribute</a>	Class for representing a constraint attribute . . . . .	313
<a href="#">multiscale::verification::ConstraintVisitor</a>	Class used to evaluate constraints . . . . .	315
<a href="#">multiscale::analysis::DBSCAN&lt; PointType &gt;</a>	Class which implements an improved version of the DBSCAN algorithm . . . . .	329
<a href="#">multiscale::analysis::Detector</a>	Abstract class for detecting entities of interest in images . . . . .	341
<a href="#">multiscale::Distribution</a>		367

---

<b>multiscale::DivisionOperation</b>	Functor representing a division operation . . . . .	370
<b>multiscaletest::EmptyTraceTest</b>	Class for testing evaluation of empty traces . . . . .	371
<b>multiscale::analysis::Entity</b>	Class for representing an entity in an image (e.g. cell, organism etc.)	374
<b>multiscale::verification::EquivalenceConstraintAttribute</b>	Class for representing an "equivalence" constraint attribute . . . . .	380
<b>multiscale::verification::EquivalenceLogicPropertyAttribute</b>	Class for representing an "equivalence" logic property attribute . . . . .	381
<b>EuclideanDataPoint</b>	Euclidean data point . . . . .	382
<b>multiscale::ExceptionHandler</b>	Exception handler class . . . . .	383
<b>multiscale::FileOpenException</b>	Class for representing exceptions when opening a file . . . . .	385
<b>multiscale::Filesystem</b>	Class containing methods for interacting with the filesystem . . . . .	388
<b>multiscale::verification::FilterNumericMeasureAttribute</b>	Class for representing a filter numeric measure . . . . .	393
<b>multiscale::verification::FilterNumericVisitor</b>	Class for evaluating filter numeric measures . . . . .	393
<b>multiscale::verification::FilterSubsetAttribute</b>	Class for representing a filter subset attribute . . . . .	398
<b>multiscale::verification::FutureLogicPropertyAttribute</b>	Class for representing a "future" logic property attribute . . . . .	400
<b>multiscale::Geometry2D</b>	Two-dimensional geometric operations . . . . .	401
<b>multiscale::verification::GlobalLogicPropertyAttribute</b>	Class for representing a "globally" logic property attribute . . . . .	423
<b>grammar</b>	Grammar . . . . .	424
<b>multiscale::verification::HeterogeneousTimeseriesComponentAttribute</b>	Class for representing a heterogeneous timeseries component attribute . . . . .	425
<b>multiscale::verification::HeterogeneousTimeseriesComponentTypeParser</b>	Symbol table and parser for the heterogeneous timeseries component type . . . . .	425
<b>multiscale::verification::TimeseriesComponentEvaluator::Homogeneous-ComponentEvaluator&lt; Relation &gt;</b>	Symbol table and parser for the homogeneous timeseries component evaluator . . . . .	426
<b>multiscale::verification::HomogeneousHomogeneousTimeseriesAttribute</b>	Class for representing a homogeneous homogeneous timeseries attribute . . . . .	428
<b>multiscale::verification::HomogeneousTimeseriesComponentAttribute</b>	Class for representing a homogeneous timeseries component attribute . . . . .	430
<b>multiscale::verification::HomogeneousTimeseriesComponentTypeParser</b>	Symbol table and parser for the homogeneous timeseries component type . . . . .	431
<b>multiscale::verification::HomogeneousTimeseriesMeasureAttribute</b>	Class for representing a homogeneous timeseries measure attribute . . . . .	431

<a href="#">multiscale::verification::HomogeneousTimeseriesMeasureTypeParser</a>	Symbol table and parser for the homogeneous timeseries measure type . . . . .	432
<a href="#">multiscale::verification::ImplicationConstraintAttribute</a>	Class for representing an "implication" constraint attribute . . . . .	433
<a href="#">multiscale::verification::ImplicationLogicPropertyAttribute</a>	Class for representing an "implication" logic property attribute . . . . .	433
<a href="#">multiscale::IndexOutOfBoundsException</a>	Class for representing an index out of bounds exception . . . . .	434
<a href="#">multiscale::InvalidInputException</a>	Class for representing invalid input exceptions . . . . .	437
<a href="#">multiscale::InvalidOutputException</a>	Class for representing invalid output exceptions . . . . .	440
<a href="#">multiscale::IOException</a>	Class for representing input and output exceptions . . . . .	443
<a href="#">multiscale::LexicographicNumberIterator</a>	Iterator class starting at 1 and ending at the provided upper bound considering that each number is followed by an "_" . . . . .	446
<a href="#">multiscale::verification::LogicPropertyAttribute</a>	Class for representing a logic property attribute . . . . .	453
<a href="#">multiscale::verification::LogicPropertyDataReader</a>	Class used to input logic properties . . . . .	456
<a href="#">multiscale::verification::LogicPropertyGrammar&lt; Iterator &gt;</a>	The grammar for parsing logic properties . . . . .	462
<a href="#">multiscale::verification::LogicPropertyVisitor</a>	Class used to evaluate logic properties . . . . .	477
<a href="#">multiscale::analysis::MatFactory</a>	Class for creating a cv::Mat object . . . . .	499
<a href="#">multiscale::MinEnclosingTriangleFinder</a>	Class for computing the minimum area enclosing triangle for a given polygon . . . . .	507
<a href="#">multiscaletest::MinEnclosingTriangleFinderTest</a>	Class for testing the minimum enclosing triangle algorithm . . . . .	532
<a href="#">multiscale::verification::ModelChecker</a>	Abstract class representing a generic model checker . . . . .	540
<a href="#">multiscale::verification::ModelCheckerFactory</a>	Interface for different model checker factories . . . . .	554
<a href="#">multiscaletest::ModelCheckerTest</a>	Class for testing model checkers . . . . .	555
<a href="#">multiscale::verification::ModelCheckingException</a>	Class for representing a model checking exception . . . . .	561
<a href="#">multiscale::verification::ModelCheckingHelpRequestException</a>	Class for representing a model checking help request exception . . . . .	565
<a href="#">multiscale::verification::ModelCheckingManager</a>	Class for managing the model checking processes . . . . .	568
<a href="#">multiscale::verification::ModelCheckingOutputWriter</a>	Class used to output the model checkers progress . . . . .	586
<a href="#">multiscale::verification::MSTMLSubfilesMerger</a>	Class for merging MSTML subfiles . . . . .	611

<b>multiscale::MultiplicationOperation</b>	Functor representing a multiplication operation . . . . .	629
<b>multiscale::verification::MultiscaleArchitectureGraph</b>	Class for defining a multiscale architecture graph . . . . .	630
<b>multiscale::MultiscaleException</b>	Parent exception class for the project . . . . .	660
<b>multiscaletest::MultiscaleTest</b>	. . . . .	664
<b>multiscale::verification::NextKLogicPropertyAttribute</b>	Class for representing a "next K" logic property attribute . . . . .	666
<b>multiscale::verification::NextLogicPropertyAttribute</b>	Class for representing a "next" logic property attribute . . . . .	667
<b>multiscale::verification::Nil</b>	A class used to avoid run-time errors when defining a variant type . . . . .	668
<b>multiscale::verification::NotConstraintAttribute</b>	Class for representing a "not" constraint attribute . . . . .	668
<b>multiscale::verification::NotLogicPropertyAttribute</b>	Class for representing a "not" logic property attribute . . . . .	669
<b>multiscale::NumberIterator</b>	Abstract class representing a number iterator . . . . .	669
<b>multiscale::Numeric</b>	Class for processing numeric (shorts, ints, floats, doubles etc.) expressions . . . . .	674
<b>multiscale::verification::NumericEvaluator</b>	Class for evaluating numeric expressions . . . . .	705
<b>multiscale::NumericException</b>	Class for representing algorithm exceptions . . . . .	708
<b>multiscale::verification::NumericMeasureAttribute</b>	Class for representing a numeric measure attribute . . . . .	711
<b>multiscale::verification::NumericMeasureCollectionAttribute</b>	Class for representing a numeric measure collection attribute . . . . .	712
<b>multiscale::verification::NumericMeasureCollectionEvaluator</b>	Class used to evaluate numeric measure collections . . . . .	713
<b>multiscale::verification::NumericMeasureCollectionVisitor</b>	Class for evaluating numeric measure collections . . . . .	715
<b>multiscale::NumericRangeManipulator</b>	Operations for ranges of numeric values . . . . .	726
<b>multiscale::verification::NumericSpatialMeasureAttribute</b>	Class for representing a numeric spatial measure attribute . . . . .	728
<b>multiscale::verification::NumericStateVariableAttribute</b>	Class for representing a numeric state variable attribute . . . . .	728
<b>multiscale::verification::NumericStateVariableEvaluator</b>	Class used to evaluate numeric state variables . . . . .	730
<b>multiscale::verification::NumericStateVariableGrammar&lt; Iterator &gt;</b>	The grammar for parsing numeric state variable statements . . . . .	732
<b>multiscale::verification::NumericStateVariableId</b>	Class for representing the identity (name, type) of a numeric state variable . . . . .	737
<b>multiscaletest::NumericStateVariableTraceTest</b>	Class for testing evaluation of numeric state variable-only traces . . . . .	742

<a href="#">multiscale::verification::NumericStatisticalMeasureAttribute</a>	Class for representing a numeric statistical measure attribute . . . . .	745
<a href="#">multiscale::verification::NumericVisitor</a>	Class for evaluating numeric measures . . . . .	746
<a href="#">multiscale::OperatingSystem</a>	Class for executing operating system related functions . . . . .	754
<a href="#">multiscale::verification::OrConstraintAttribute</a>	Class for representing an "or" constraint attribute . . . . .	758
<a href="#">multiscale::verification::OrLogicPropertyAttribute</a>	Class for representing an "or" logic property attribute . . . . .	759
<a href="#">multiscale::verification::Parser</a>	Class used for parsing (P)BLSTL logical queries . . . . .	759
<a href="#">multiscale::verification::ParserGrammarExceptionHandler</a>	Class for handling parser grammar exceptions . . . . .	764
<a href="#">multiscale::verification::ParserGrammarExtraInputException</a>	Class for representing "extra input" exceptions in the parsing process	767
<a href="#">multiscale::verification::ParserGrammarProbabilityException</a>	Class for representing "probability" exceptions in the parsing process	769
<a href="#">multiscale::verification::ParserGrammarUnexpectedTokenException</a>	Class for representing "unexpected token" exceptions in the parsing process . . . . .	772
<a href="#">multiscale::verification::ParserGrammarUnparseableInputException</a>	Class for representing "unparseable input" exceptions in the parsing process . . . . .	775
<a href="#">multiscale::visualisation::PolarCsvToInputFilesConverter</a>	Csv file to input file converter considering polar coordinates . . . . .	777
<a href="#">multiscale::visualisation::PolarGnuplotScriptGenerator</a>	Gnuplot script generator from the provided annular sectors . . . . .	791
<a href="#">multiscale::verification::PrimaryConstraintAttribute</a>	Class for representing a primary constraint attribute . . . . .	799
<a href="#">multiscale::verification::PrimaryLogicPropertyAttribute</a>	Class for representing a primary logic property attribute . . . . .	799
<a href="#">multiscale::verification::PrimaryNumericMeasureAttribute</a>	Class for representing a primary numeric measure attribute . . . . .	800
<a href="#">multiscale::verification::PrimaryNumericMeasureGrammar&lt; Iterator &gt;</a>	The grammar for parsing primary numeric measure statements . . . . .	801
<a href="#">multiscale::verification::PrimarySpatialMeasureCollectionAttribute</a>	Class used to represent a primary spatial measure collection attribute . . . . .	810
<a href="#">multiscale::verification::ProbabilisticBlackBoxModelChecker</a>	Class used to run probabilistic black-box model checking tasks . . . . .	812
<a href="#">multiscale::verification::ProbabilisticBlackBoxModelCheckerFactory</a>	Class for creating <a href="#">ProbabilisticBlackBoxModelChecker</a> instances . . . . .	817
<a href="#">multiscaletest::ProbabilisticBlackBoxModelCheckerTest</a>	Class for testing the probabilistic black-box model checker . . . . .	819
<a href="#">multiscale::verification::ProbabilisticLogicPropertyAttribute</a>	Class for representing a probabilistic logic property attribute . . . . .	822
<a href="#">multiscale::verification::ProbabilityErrorHandler</a>	Structure for defining the error handler for invalid probability errors . . . . .	826

<b>multiscale::visualisation::RectangularCsvToInputFilesConverter</b>	Csv file to input file converter considering cartesian coordinates . . . . .	827
<b>multiscale::visualisation::RectangularEntityCsvToInputFilesConverter</b>	Csv entity file to input file converter considering cartesian coordinates . . . . .	840
<b>multiscale::visualisation::RectangularGnuplotScriptGenerator</b>	Gnuplot script generator from the provided concentrations considering a rectangular geometry . . . . .	851
<b>multiscale::analysis::RectangularMatFactory</b>	Class for creating a cv::Mat object considering a rectangular grid . . . . .	859
<b>multiscale::verification::Region</b>	Class for representing a region . . . . .	864
<b>multiscale::analysis::Region</b>	Class for representing a region . . . . .	867
<b>multiscale::analysis::RegionDetector</b>	Class for detecting regions of high intensity in grayscale images . . . . .	879
<b>multiscale::verification::ProbabilityErrorHandler::result&lt; typename, typename, typename &gt;</b>	Structure for specifying the type of the result . . . . .	908
<b>multiscale::verification::UnexpectedTokenErrorHandler::result&lt; typename, typename, typename &gt;</b>	Structure for specifying the type of the result . . . . .	909
<b>multiscale::RGBColourGenerator</b>	Generate a RGB colour . . . . .	910
<b>multiscale::RuntimeException</b>	Class for representing runtime exceptions . . . . .	913
<b>multiscale::verification::ScaleAndSubsystem</b>	Class for representing a scale and subsystem . . . . .	916
<b>multiscale::verification::ScaleAndSubsystemAttribute</b>	Class for representing a scale and subsystem attribute . . . . .	918
<b>multiscale::verification::ScaleAndSubsystemEvaluator</b>		920
<b>multiscale::verification::ScaleAndSubsystemGrammar&lt; Iterator &gt;</b>	The grammar for parsing scale and subsystem statements . . . . .	923
<b>multiscale::verification::ScaleAndSubsystemStringGrammar&lt; Iterator &gt;</b>	The grammar for parsing scale and subsystem string statements . . . . .	928
<b>multiscale::analysis::Silhouette</b>	Class for computing the "Silhouette" clustering index . . . . .	933
<b>multiscale::verification::SimilarityMeasureAttribute</b>	Class for representing a similarity measure attribute . . . . .	937
<b>multiscale::verification::SimilarityMeasureTypeParser</b>	Symbol table and parser for the similarity measure type . . . . .	938
<b>multiscale::verification::SimilarityTemporalNumericCollectionAttribute</b>	Class for representing a similarity temporal numeric collection attribute . . . . .	939
<b>multiscale::analysis::SimulationClusterDetector</b>	Class for detecting clusters in 2D images obtained from simulations . . . . .	940
<b>multiscaletest::SpatialEntitiesTraceTest</b>	Class for testing evaluation of spatial entities-only traces . . . . .	950
<b>multiscale::verification::SpatialEntity</b>	Class for representing a spatial entity . . . . .	953

<a href="#">multiscale::analysis::SpatialEntityPseudo3D</a>	
Class for representing a pseudo-3D (explicit 2D + implicit height) object . . . . .	961
<a href="#">multiscale::verification::SpatialMeasureAttribute</a>	
Class for representing a spatial measure attribute . . . . .	979
<a href="#">multiscale::analysis::SpatialMeasureCalculator</a>	
Class for computing spatial measures . . . . .	979
<a href="#">multiscale::verification::SpatialMeasureCollectionAttribute</a>	
Class used to represent a spatial measure collection . . . . .	986
<a href="#">multiscale::verification::SpatialMeasureCollectionGrammar&lt; Iterator &gt;</a>	
The grammar for parsing spatial measure collection statements . . . . .	986
<a href="#">multiscale::verification::SpatialMeasureCollectionVisitor</a>	
Class for evaluating spatial measure collections . . . . .	1004
<a href="#">multiscale::verification::SpatialMeasureEvaluator</a>	
Class for evaluating spatial measures . . . . .	1009
<a href="#">multiscale::verification::SpatialMeasureTypeParser</a>	
Symbol table and parser for the spatial measure type . . . . .	1010
<a href="#">multiscale::verification::SpatialNumericComparisonAttribute</a>	
Class for representing a spatial numeric comparison attribute . . . . .	1011
<a href="#">multiscale::verification::SpatialTemporalDataReader</a>	
Class for reading spatial temporal trace data from input files . . . . .	1012
<a href="#">multiscale::verification::SpatialTemporalDataWriter</a>	
Class for writing spatial temporal traces to output files . . . . .	1029
<a href="#">multiscale::verification::SpatialTemporalException</a>	
Class for representing a spatial temporal exception . . . . .	1039
<a href="#">multiscale::verification::SpatialTemporalTrace</a>	
Class for representing a spatial temporal trace . . . . .	1043
<a href="#">multiscale::StandardNumberIterator</a>	
Iterator class starting at 1 and iterating over all natural numbers until the provided upper bound is reached . . . . .	1059
<a href="#">multiscale::verification::StateVariable</a>	
Class for representing a state variable . . . . .	1063
<a href="#">multiscale::verification::StateVariableAttribute</a>	
Class for representing a state variable attribute . . . . .	1067
<a href="#">multiscale::verification::StatisticalModelChecker</a>	
Class used to run statistical model checking tasks . . . . .	1069
<a href="#">multiscale::verification::StatisticalModelCheckerFactory</a>	
Class for creating StatisticalModelChecker instances . . . . .	1086
<a href="#">multiscaletest::StatisticalModelCheckerTest</a>	
Class for testing the statistical model checker . . . . .	1089
<a href="#">multiscale::StringManipulator</a>	
Class for manipulating strings . . . . .	1093
<a href="#">multiscale::verification::SubsetAttribute</a>	
Class for representing a subset attribute . . . . .	1100
<a href="#">multiscale::verification::SubsetOperationAttribute</a>	
Class for representing a subset operation attribute . . . . .	1100
<a href="#">multiscale::verification::SubsetOperationTypeParser</a>	
Symbol table and parser for the subset operation type . . . . .	1101
<a href="#">multiscale::verification::SubsetSpecificAttribute</a>	
Class for representing a subset specific attribute . . . . .	1102

<b>multiscale::verification::SubsetSpecificTypeParser</b>	Symbol table and parser for a specific subset type . . . . .	1102
<b>multiscale::verification::SubsetSubsetOperationAttribute</b>	Class for representing a subset subset operation attribute . . . . .	1103
<b>multiscale::verification::SubsetVisitor</b>	Class used to evaluate subsets . . . . .	1105
<b>multiscale::SubtractionOperation</b>	Functor representing a subtraction operation . . . . .	1111
<b>multiscale::TangentsFromPointToPolygonFinder&lt; PolygonPointsType, ReferencePointType &gt;</b>	Class for finding the tangents from a point to a polygon . . . . .	1111
<b>multiscale::verification::TemporalDataReader</b>	Class for reading (non-spatial) timeseries data from a .csv file . . . . .	1132
<b>multiscale::verification::TemporalNumericCollectionAttribute</b>	Class for representing a temporal numeric collection attribute . . . . .	1141
<b>multiscale::verification::TemporalNumericCollectionGrammar&lt; Iterator &gt;</b>	The grammar for parsing temporal numeric collection statements . . . . .	1142
<b>multiscale::verification::TemporalNumericComparisonAttribute</b>	Class for representing a temporal numeric comparison attribute . . . . .	1154
<b>multiscale::verification::TemporalNumericMeasureAttribute</b>	Class for representing a temporal numeric measure attribute . . . . .	1156
<b>multiscale::verification::TemporalNumericMeasureCollectionAttribute</b>	Class for representing temporal numeric measure collection attributes . . . . .	1157
<b>multiscale::verification::TemporalNumericMeasureGrammar&lt; Iterator &gt;</b>	The grammar for parsing temporal numeric measure statements . . . . .	1158
<b>multiscale::verification::TemporalNumericVisitor</b>	Class for evaluating temporal numeric measures . . . . .	1169
<b>multiscale::TestException</b>	Class for representing testing exceptions . . . . .	1177
<b>multiscale::verification::TimePoint</b>	Class for representing a timepoint . . . . .	1180
<b>multiscale::verification::TimePointEvaluator</b>	Class used to evaluate timepoints . . . . .	1201
<b>multiscale::verification::TimeseriesComponentAttribute</b>	Class for representing a timeseries component attribute . . . . .	1203
<b>multiscale::verification::TimeseriesComponentEvaluator</b>	Class for evaluating timeseries components . . . . .	1203
<b>multiscale::verification::TimeseriesComponentVisitor</b>	Class for evaluating timeseries components . . . . .	1205
<b>multiscale::verification::TimeseriesMeasureAttribute</b>	Class for representing a timeseries measure attribute . . . . .	1209
<b>multiscale::verification::TimeseriesMeasureTypeParser</b>	Symbol table and parser for the timeseries measure type . . . . .	1210
<b>multiscale::verification::TimeseriesTimeseriesComponentAttribute</b>	Class for representing a timeseries timeseries component attribute . . . . .	1211
<b>multiscaletest::TraceEvaluationTest</b>	Class for testing evaluation of traces . . . . .	1212
<b>multiscale::verification::UnaryNumericFilterAttribute</b>	Class for representing a unary numeric filter attribute . . . . .	1221

<b>multiscale::verification::UnaryNumericMeasureAttribute</b>	
Class for representing a unary numeric measure attribute . . . . .	1223
<b>multiscale::verification::UnaryNumericMeasureGrammar&lt; Iterator &gt;</b>	
The grammar for parsing unary numeric measure statements . . . . .	1224
<b>multiscale::verification::UnaryNumericMeasureTypeParser</b>	
Symbol table and parser for the unary numeric measure type . . . . .	1227
<b>multiscale::verification::UnaryNumericNumericAttribute</b>	
Class for representing a unary numeric numeric measure attribute . . . . .	1228
<b>multiscale::verification::UnaryNumericSpatialAttribute</b>	
Class for representing a unary numeric spatial measure collection attribute . . . . .	1230
<b>multiscale::verification::UnaryNumericTemporalAttribute</b>	
Class for representing a unary numeric temporal measure attribute . . . . .	1231
<b>multiscale::verification::UnaryScaleAndSubsystemConstraintAttribute</b>	
Class for representing a "unary" scale and subsystem constraint at- tribute . . . . .	1233
<b>multiscale::verification::UnarySpatialConstraintAttribute</b>	
Class for representing a "unary" spatial constraint attribute . . . . .	1234
<b>multiscale::verification::UnaryStatisticalMeasureAttribute</b>	
Class for representing a unary statistical measure attribute . . . . .	1236
<b>multiscale::verification::UnaryStatisticalMeasureGrammar&lt; Iterator &gt;</b>	
The grammar for parsing unary statistical measure statements . . . . .	1237
<b>multiscale::verification::UnaryStatisticalMeasureTypeParser</b>	
Symbol table and parser for the unary statistical measure type . . . . .	1240
<b>multiscale::verification::UnaryStatisticalNumericAttribute</b>	
Class for representing a unary statistical numeric attribute . . . . .	1241
<b>multiscale::verification::UnaryStatisticalSpatialAttribute</b>	
Class for representing a unary statistical spatial attribute . . . . .	1242
<b>multiscale::UnexpectedBehaviourException</b>	
Class for representing unexpected behaviour exceptions . . . . .	1244
<b>multiscale::verification::UnexpectedTokenErrorHandler</b>	
Structure for defining the error handler for unexpected token errors . . . . .	1246
<b>multiscale::verification::TimeseriesComponentEvaluator::UniformHomogeneous-     ComponentEvaluator&lt; Relation &gt;</b>	
. . . . .	1248
<b>multiscale::UnimplementedMethodException</b>	
Class for representing unimplemented method exceptions . . . . .	1250
<b>multiscale::verification::UntilLogicPropertyAttribute</b>	
Class for representing an "until" logic property attribute . . . . .	1253
<b>multiscale::UserDefinedTypeName&lt; T &gt;</b>	
Class for representing a user defined type name . . . . .	1254
<b>multiscale::XmlValidator::XmlValidationErrorHandler</b>	
Class used for handling errors during the xml file validation process	1255
<b>multiscale::XmlValidator</b>	
Class used to validate xml files . . . . .	1260



# Chapter 5

## Namespace Documentation

### 5.1 multiscale Namespace Reference

#### Namespaces

- namespace [analysis](#)
- namespace [verification](#)
- namespace [visualisation](#)

#### Classes

- class [UserDefinedTypeName](#)  
*Class for representing a user defined type name.*
- class [AlgorithmException](#)  
*Class for representing algorithm exceptions.*
- class [ExceptionHandler](#)  
*Exception handler class.*
- class [FileOpenException](#)  
*Class for representing exceptions when opening a file.*
- class [IndexOutOfBoundsException](#)  
*Class for representing an index out of bounds exception.*
- class [InvalidInputException](#)  
*Class for representing invalid input exceptions.*
- class [InvalidOutputException](#)  
*Class for representing invalid output exceptions.*
- class [IOException](#)  
*Class for representing input and output exceptions.*
- class [MultiscaleException](#)  
*Parent exception class for the project.*
- class [NumericException](#)

- class [RuntimeException](#)

*Class for representing algorithm exceptions.*
- class [TestException](#)

*Class for representing runtime exceptions.*
- class [UnexpectedBehaviourException](#)

*Class for representing testing exceptions.*
- class [UnimplementedMethodException](#)

*Class for representing unexpected behaviour exceptions.*
- class [ConsolePrinter](#)

*Class for representing unimplemented method exceptions.*
- class [Filesystem](#)

*Class used to print (coloured) messages to the console.*
- class [MinEnclosingTriangleFinder](#)

*Class containing methods for interacting with the filesystem.*
- class [TangentsFromPointToPolygonFinder](#)

*Class for computing the minimum area enclosing triangle for a given polygon.*
- class [Geometry2D](#)

*Class for finding the tangents from a point to a polygon.*
- class [LexicographicNumberIterator](#)

*Two-dimensional geometric operations.*
- class [StandardNumberIterator](#)

*Iterator class starting at 1 and ending at the provided upper bound considering that each number is followed by an "\_" .*
- class [NumberIterator](#)

*Iterator class starting at 1 and iterating over all natural numbers until the provided upper bound is reached.*
- class [Numeric](#)

*Abstract class representing a number iterator.*
- class [AdditionOperation](#)

*Class for processing numeric (shorts, ints, floats, doubles etc.) expressions.*
- class [DivisionOperation](#)

*Functor representing an addition operation.*
- class [MultiplicationOperation](#)

*Functor representing a division operation.*
- class [SubtractionOperation](#)

*Functor representing a multiplication operation.*
- class [NumericRangeManipulator](#)

*Functor representing a subtraction operation.*
- class [OperatingSystem](#)

*Operations for ranges of numeric values.*
- class [RGBColourGenerator](#)

*Class for executing operating system related functions.*
- class [BetaDistribution](#)

*Generate a RGB colour.*

- class [BinomialDistribution](#)  
*Class for analysing Beta distributed data.*
- class [Distribution](#)  
*Class for analysing Binomial distributed data.*
- class [StringManipulator](#)  
*Class for manipulating strings.*
- class [XmlValidator](#)  
*Class used to validate xml files.*

## Enumerations

- enum [UnixColourCode](#) { **BLACK** = 0, **RED** = 1, **GREEN** = 2, **YELLOW** = 3, **BLUE** = 4, **MAGENTA** = 5, **CYAN** = 6, **WHITE** = 7 }
- enum [WindowsColourCode](#) { **BLACK** = 0, **DARK\_BLUE** = 1, **DARK\_GREEN** = 2, **DARK\_CYAN** = 3, **DARK\_RED** = 4, **DARK\_MAGENTA** = 5, **DARK\_YELLOW** = 6, **DARK\_WHITE** = 7, **GRAY** = 8, **BLUE** = 4, **GREEN** = 2, **CYAN** = 6, **RED** = 1, **MAGENTA** = 5, **YELLOW** = 3, **WHITE** = 7 }
- enum [ColourCode](#) { **BLACK** = 0, **RED** = 1, **GREEN** = 2, **YELLOW** = 3, **BLUE** = 4, **MAGENTA** = 5, **CYAN** = 6, **WHITE** = 7 }
- enum [NumberIteratorType](#) { **STANDARD** = 1, **LEXICOGRAPHIC** = 2 }

*The type of the number iterator.*

## Variables

- const int [EXEC\\_SUCCESS\\_CODE](#) = 0
- const int [EXEC\\_ERR\\_CODE](#) = 1
- const std::string [ERR\\_MSG](#) = "An error occurred."
- const std::string [ERR\\_UNDEFINED\\_ENUM\\_VALUE](#) = "The provided enumeration value is invalid. Please use one of the available enumeration values instead."
- const std::string [ERR\\_INDEX\\_OUT\\_OF\\_BOUNDS\\_BEGIN](#) = "The provided index value ("
- const std::string [ERR\\_INDEX\\_OUT\\_OF\\_BOUNDS\\_END](#) = ") is invalid. Please change."
- const std::string [ERR\\_UNIMPLEMENTED\\_METHOD](#) = "The method you tried to call is not implemented. Please change."

### 5.1.1 Enumeration Type Documentation

#### 5.1.1.1 enum multiscale::ColourCode

Enumerator:

- BLACK** Black non-colour
- RED** Red colour
- GREEN** Green colour

**YELLOW** Yellow colour

**BLUE** Blue colour

**MAGENTA** Magenta colour

**CYAN** Cyan colour

**WHITE** White non-colour

Faint white non-colour

Definition at line 44 of file ConsolePrinter.hpp.

#### 5.1.1.2 enum multiscale::NumberIteratorType

The type of the number iterator.

Enumerator:

**STANDARD** Standard number iterator

**LEXICOGRAPHIC** Lexicographic number iterator

Definition at line 7 of file NumberIteratorType.hpp.

#### 5.1.1.3 enum multiscale::UnixColourCode

Enumerator:

**BLACK** Black non-colour

**RED** Red colour

**GREEN** Green colour

**YELLOW** Yellow colour

**BLUE** Blue colour

**MAGENTA** Magenta colour

**CYAN** Cyan colour

**WHITE** White non-colour

Faint white non-colour

Definition at line 12 of file ConsolePrinter.hpp.

#### 5.1.1.4 enum multiscale::WindowsColourCode

Enumerator:

**BLACK** Black non-colour

**DARK\_BLUE** Dark blue colour

**DARK\_GREEN** Dark green colour

**DARK\_CYAN** Dark cyan colour

**DARK\_RED** Dark red colour  
**DARK\_MAGENTA** Dark magenta colour  
**DARK\_YELLOW** Dark yellow colour  
**DARK\_WHITE** White non-colour  
**GRAY** Gray non-colour  
**BLUE** Blue colour  
**GREEN** Green colour  
**CYAN** Cyan colour  
**RED** Red colour  
**MAGENTA** Magenta colour  
**YELLOW** Yellow colour  
**WHITE** White non-colour  
Faint white non-colour

Definition at line 24 of file ConsolePrinter.hpp.

### 5.1.2 Variable Documentation

5.1.2.1 `const std::string multiscale::ERR_INDEX_OUT_OF_BOUNDS_BEGIN = "The provided index value ("`

Definition at line 30 of file Multiscale.hpp.

Referenced by multiscale::IndexOutOfBoundsException::IndexOutOfBoundsException().

5.1.2.2 `const std::string multiscale::ERR_INDEX_OUT_OF_BOUNDS_END = ") is invalid. Please change."`

Definition at line 31 of file Multiscale.hpp.

Referenced by multiscale::IndexOutOfBoundsException::IndexOutOfBoundsException().

5.1.2.3 `const std::string multiscale::ERR_MSG = "An error occurred: "`

Definition at line 25 of file Multiscale.hpp.

Referenced by multiscale::ExceptionHandler::printDetailedErrorMessage(), and multiscale::ExceptionHandler::printRawErrorMessage().

5.1.2.4 `const std::string multiscale::ERR_UNDEFINED_ENUM_VALUE = "The provided enumeration value is invalid. Please use one of the available enumeration values instead."`

Definition at line 28 of file Multiscale.hpp.

Referenced by multiscale::verification::LogicPropertyVisitor::computeDissimilarityValue(), multiscale::verification::ChangeMeasureEvaluator::computeNumericMeasureValueChange(), multiscale::verification::NumericEvaluator::evaluate(), multiscale::verification::ComparatorEvaluator::evaluate(), multiscale::verification::TimeseriesComponentEvaluator::evaluate(), multiscale::verification::NumericMeasureCollectionVisitor::evaluateHomogeneousHomogeneousTimeseries(), multiscale::verification::SubsetVisitor::evaluateSubsetOperation(), multiscale::verification::NumericMeasureCollectionVisitor::evaluateTimeseriesTimeseriesComponent(), multiscale::verification::spatialmeasure::getMaxValidSpatialMeasureValue(), multiscale::verification::spatialmeasure::getMinValidSpatialMeasureValue(), multiscale::verification::TimeseriesComponentVisitor::operator()(), multiscale::verification::subsetspecific::toString(), multiscale::verification::spatialmeasure::toString(), multiscale::verification::spatialmeasure::validateSpatialMeasureType(), multiscale::verification::spatialmeasure::validateSpatialMeasureTypeIndex(), multiscale::verification::subsetspecific::validateSubsetSpecificType(), and multiscale::verification::subsetspecific::validateSubsetSpecificTypeIndex().

**5.1.2.5 const std::string multiscale::ERR\_UNIMPLEMENTED\_METHOD = "The method you tried to call is not implemented. Please change."**

Definition at line 31 of file UnimplementedMethodException.hpp.

**5.1.2.6 const int multiscale::EXEC\_ERR\_CODE = 1**

Definition at line 22 of file Multiscale.hpp.

**5.1.2.7 const int multiscale::EXEC\_SUCCESS\_CODE = 0**

Definition at line 21 of file Multiscale.hpp.

## 5.2 multiscale::analysis Namespace Reference

### Classes

- class [ClusterDetector](#)  
*Class for detecting clusters in 2D images.*
- class [Detector](#)  
*Abstract class for detecting entities of interest in images.*
- class [RegionDetector](#)  
*Class for detecting regions of high intensity in grayscale images.*
- class [SimulationClusterDetector](#)  
*Class for detecting clusters in 2D images obtained from simulations.*
- class [CircularMatFactory](#)  
*Class for creating a Mat object considering a circular grid.*

- class [MatFactory](#)  
*Class for creating a cv::Mat object.*
- class [RectangularMatFactory](#)  
*Class for creating a cv::Mat object considering a rectangular grid.*
- class [Cluster](#)  
*Class for representing a cluster of entities in an image.*
- class [Entity](#)  
*Class for representing an entity in an image (e.g. cell, organism etc.)*
- class [Region](#)  
*Class for representing a region.*
- class [SpatialEntityPseudo3D](#)  
*Class for representing a pseudo-3D (explicit 2D + implicit height) object.*
- class [CircularityMeasure](#)  
*Class for computing the circularity measure for the given collection of points.*
- class [DBSCAN](#)  
*Class which implements an improved version of the [DBSCAN](#) algorithm.*
- class [Silhouette](#)  
*Class for computing the "Silhouette" clustering index.*
- class [SpatialMeasureCalculator](#)  
*Class for computing spatial measures.*

## Typedefs

- `typedef std::pair< std::vector < cv::Point >, std::vector < std::vector < cv::Point > > > > Polygon`

## Enumerations

- enum [Shape2D](#) { `Triangle` = 0, `Rectangle`, `Circle`, `Undefined` }  
*Enumeration for determining the type of a 2D shape.*
- enum [SpatialEntityPseudo3DType](#) { `Cluster` = 0, `Region` }  
*Enumeration for determining the type of a pseudo 3D entity.*
- enum [DBSCANPointClusteringTag](#) { `UNCLASSIFIED` = -2, `BORDER` = -1, `NOISE` = 0 }  
*Enumeration for storing the DBSCAN point clustering tags.*

### 5.2.1 Typedef Documentation

#### 5.2.1.1 `typedef std::pair<std::vector<cv::Point>, std::vector<std::vector<cv::Point>> > > multiscale::analysis::Polygon`

Define a wrapper for polygons i.e. pairs (o, i) where o = outer contour and i = collection of inner contours/holes

Definition at line 20 of file RegionDetector.hpp.

### 5.2.2 Enumeration Type Documentation

#### 5.2.2.1 enum multiscale::analysis::DBSCANPointClusteringTag

Enumeration for storing the [DBSCAN](#) point clustering tags.

Enumerator:

**UNCLASSIFIED** The point was not attributed to a cluster

**BORDER** The point is a border point

**NOISE** The point is noise

Definition at line 15 of file DBSCAN.hpp.

#### 5.2.2.2 enum multiscale::analysis::Shape2D

Enumeration for determining the type of a 2D shape.

Enumerator:

**Triangle** Triangular 2D shape

**Rectangle** Rectangular 2D shape

**Circle** Circular 2D shape

**Undefined** Undefined 2D shape

Definition at line 10 of file Shape2D.hpp.

#### 5.2.2.3 enum multiscale::analysis::SpatialEntityPseudo3DType

Enumeration for determining the type of a pseudo 3D entity.

Enumerator:

**Cluster** Cluster

**Region** Region

Definition at line 10 of file SpatialEntityPseudo3DType.hpp.

## 5.3 multiscale::verification Namespace Reference

### Namespaces

- namespace [spatialmeasure](#)
- namespace [subsetsspecific](#)

## Classes

- class [AndConstraintAttribute](#)  
*Class for representing an "and" constraint attribute.*
- class [AndLogicPropertyAttribute](#)  
*Class for representing an "and" logic property attribute.*
- class [BinaryNumericFilterAttribute](#)  
*Class for representing a binary numeric filter attribute.*
- class [BinaryNumericMeasureAttribute](#)  
*Class for representing a binary numeric measure attribute.*
- class [BinaryNumericNumericAttribute](#)  
*Class for representing a binary numeric numeric measure attribute.*
- class [BinaryNumericSpatialAttribute](#)  
*Class for representing a binary numeric spatial measure collection attribute.*
- class [BinaryNumericTemporalAttribute](#)  
*Class for representing a binary numeric temporal measure attribute.*
- class [BinaryStatisticalMeasureAttribute](#)  
*Class for representing a binary statistical measure attribute.*
- class [BinaryStatisticalNumericAttribute](#)  
*Class for representing a binary statistical numeric attribute.*
- class [BinaryStatisticalQuantileMeasureAttribute](#)  
*Class for representing a binary statistical quantile measure attribute.*
- class [BinaryStatisticalQuantileNumericAttribute](#)  
*Class for representing a binary statistical quantile numeric attribute.*
- class [BinaryStatisticalQuantileSpatialAttribute](#)  
*Class for representing a binary statistical quantile spatial attribute.*
- class [BinaryStatisticalSpatialAttribute](#)  
*Class for representing a binary statistical spatial attribute.*
- class [ChangeMeasureAttribute](#)  
*Class for representing a change measure attribute.*
- class [ChangeTemporalNumericCollectionAttribute](#)  
*Class for representing a change temporal numeric collection attribute.*
- class [ChangeTemporalNumericMeasureAttribute](#)  
*Class for representing a change temporal numeric measure attribute.*
- class [ComparatorAttribute](#)  
*Class for representing a comparator attribute.*
- class [ConstraintAttribute](#)  
*Class for representing a constraint attribute.*
- class [EquivalenceConstraintAttribute](#)  
*Class for representing an "equivalence" constraint attribute.*
- class [EquivalenceLogicPropertyAttribute](#)  
*Class for representing an "equivalence" logic property attribute.*
- class [FilterNumericMeasureAttribute](#)

- class [FilterSubsetAttribute](#)

*Class for representing a filter numeric measure.*
- class [FutureLogicPropertyAttribute](#)

*Class for representing a filter subset attribute.*
- class [GlobalLogicPropertyAttribute](#)

*Class for representing a "future" logic property attribute.*
- class [HeterogeneousTimeseriesComponentAttribute](#)

*Class for representing a "globally" logic property attribute.*
- class [HomogeneousHomogeneousTimeseriesAttribute](#)

*Class for representing a heterogeneous timeseries component attribute.*
- class [HomogeneousTimeseriesComponentAttribute](#)

*Class for representing a homogeneous homogeneous timeseries attribute.*
- class [HomogeneousTimeseriesMeasureAttribute](#)

*Class for representing a homogeneous timeseries component attribute.*
- class [ImplicationConstraintAttribute](#)

*Class for representing an "implication" constraint attribute.*
- class [ImplicationLogicPropertyAttribute](#)

*Class for representing an "implication" logic property attribute.*
- class [LogicPropertyAttribute](#)

*Class for representing a logic property attribute.*
- class [NextKLogicPropertyAttribute](#)

*Class for representing a "next K" logic property attribute.*
- class [NextLogicPropertyAttribute](#)

*Class for representing a "next" logic property attribute.*
- class [Nil](#)

*A class used to avoid run-time errors when defining a variant type.*
- class [NotConstraintAttribute](#)

*Class for representing a "not" constraint attribute.*
- class [NotLogicPropertyAttribute](#)

*Class for representing a "not" logic property attribute.*
- class [NumericMeasureAttribute](#)

*Class for representing a numeric measure attribute.*
- class [NumericMeasureCollectionAttribute](#)

*Class for representing a numeric measure collection attribute.*
- class [NumericSpatialMeasureAttribute](#)

*Class for representing a numeric spatial measure attribute.*
- class [NumericStateVariableAttribute](#)

*Class for representing a numeric state variable attribute.*
- class [NumericStatisticalMeasureAttribute](#)

*Class for representing a numeric statistical measure attribute.*
- class [OrConstraintAttribute](#)

*Class for representing an "or" constraint attribute.*

- class [OrLogicPropertyAttribute](#)  
*Class for representing an "or" logic property attribute.*
- class [PrimaryConstraintAttribute](#)  
*Class for representing a primary constraint attribute.*
- class [PrimaryLogicPropertyAttribute](#)  
*Class for representing a primary logic property attribute.*
- class [PrimaryNumericMeasureAttribute](#)  
*Class for representing a primary numeric measure attribute.*
- class [PrimarySpatialMeasureCollectionAttribute](#)  
*Class used to represent a primary spatial measure collection attribute.*
- class [ProbabilisticLogicPropertyAttribute](#)  
*Class for representing a probabilistic logic property attribute.*
- class [ScaleAndSubsystemAttribute](#)  
*Class for representing a scale and subsystem attribute.*
- class [SimilarityMeasureAttribute](#)  
*Class for representing a similarity measure attribute.*
- class [SimilarityTemporalNumericCollectionAttribute](#)  
*Class for representing a similarity temporal numeric collection attribute.*
- class [SpatialMeasureAttribute](#)  
*Class for representing a spatial measure attribute.*
- class [SpatialMeasureCollectionAttribute](#)  
*Class used to represent a spatial measure collection.*
- class [SpatialNumericComparisonAttribute](#)  
*Class for representing a spatial numeric comparison attribute.*
- class [StateVariableAttribute](#)  
*Class for representing a state variable attribute.*
- class [SubsetAttribute](#)  
*Class for representing a subset attribute.*
- class [SubsetOperationAttribute](#)  
*Class for representing a subset operation attribute.*
- class [SubsetSpecificAttribute](#)  
*Class for representing a subset specific attribute.*
- class [SubsetSubsetOperationAttribute](#)  
*Class for representing a subset subset operation attribute.*
- class [TemporalNumericCollectionAttribute](#)  
*Class for representing a temporal numeric collection attribute.*
- class [TemporalNumericComparisonAttribute](#)  
*Class for representing a temporal numeric comparison attribute.*
- class [TemporalNumericMeasureAttribute](#)  
*Class for representing a temporal numeric measure attribute.*
- class [TemporalNumericMeasureCollectionAttribute](#)  
*Class for representing temporal numeric measure collection attributes.*
- class [TimeseriesComponentAttribute](#)

- class [TimeseriesMeasureAttribute](#)  
*Class for representing a timeseries component attribute.*
- class [TimeseriesTimeseriesComponentAttribute](#)  
*Class for representing a timeseries measure attribute.*
- class [UnaryNumericFilterAttribute](#)  
*Class for representing a timeseries timeseries component attribute.*
- class [UnaryNumericMeasureAttribute](#)  
*Class for representing a unary numeric filter attribute.*
- class [UnaryNumericNumericAttribute](#)  
*Class for representing a unary numeric measure attribute.*
- class [UnaryNumericSpatialAttribute](#)  
*Class for representing a unary numeric spatial measure collection attribute.*
- class [UnaryNumericTemporalAttribute](#)  
*Class for representing a unary numeric temporal measure attribute.*
- class [UnaryScaleAndSubsystemConstraintAttribute](#)  
*Class for representing a "unary" scale and subsystem constraint attribute.*
- class [UnarySpatialConstraintAttribute](#)  
*Class for representing a "unary" spatial constraint attribute.*
- class [UnaryStatisticalMeasureAttribute](#)  
*Class for representing a unary statistical measure attribute.*
- class [UnaryStatisticalNumericAttribute](#)  
*Class for representing a unary statistical numeric attribute.*
- class [UnaryStatisticalSpatialAttribute](#)  
*Class for representing a unary statistical spatial attribute.*
- class [UntilLogicPropertyAttribute](#)  
*Class for representing an "until" logic property attribute.*
- class [ApproximateBayesianModelChecker](#)  
*Class used to run approximate Bayesian model checking tasks.*
- class [ApproximateBayesianModelCheckerFactory](#)  
*Class for creating `ApproximateBayesianModelChecker` instances.*
- class [ApproximateProbabilisticModelChecker](#)  
*Class used to run approximate probabilistic model checking tasks.*
- class [ApproximateProbabilisticModelCheckerFactory](#)  
*Class for creating `ApproximateProbabilisticModelChecker` instances.*
- class [BayesianModelChecker](#)  
*Class used to run Bayesian model checking tasks.*
- class [BayesianModelCheckerFactory](#)  
*Class for creating `BayesianModelChecker` instances.*
- class [ModelChecker](#)  
*Abstract class representing a generic model checker.*
- class [ModelCheckerFactory](#)  
*Interface for different model checker factories.*

- class [ModelCheckingManager](#)  
*Class for managing the model checking processes.*
- class [ModelCheckingOutputWriter](#)  
*Class used to output the model checkers progress.*
- class [ProbabilisticBlackBoxModelChecker](#)  
*Class used to run probabilistic black-box model checking tasks.*
- class [ProbabilisticBlackBoxModelCheckerFactory](#)  
*Class for creating ProbabilisticBlackBoxModelChecker instances.*
- class [StatisticalModelChecker](#)  
*Class used to run statistical model checking tasks.*
- class [StatisticalModelCheckerFactory](#)  
*Class for creating StatisticalModelChecker instances.*
- class [LogicPropertyDataReader](#)  
*Class used to input logic properties.*
- class [MSTMLSubfilesMerger](#)  
*Class for merging MSTML subfiles.*
- class [SpatialTemporalDataReader](#)  
*Class for reading spatial temporal trace data from input files.*
- class [SpatialTemporalDataWriter](#)  
*Class for writing spatial temporal traces to output files.*
- class [TemporalDataReader](#)  
*Class for reading (non-spatial) timeseries data from a .csv file.*
- class [ModelCheckingException](#)  
*Class for representing a model checking exception.*
- class [ModelCheckingHelpRequestException](#)  
*Class for representing a model checking help request exception.*
- class [ParserGrammarExceptionHandler](#)  
*Class for handling parser grammar exceptions.*
- class [ParserGrammarExtraInputException](#)  
*Class for representing "extra input" exceptions in the parsing process.*
- class [ParserGrammarProbabilityException](#)  
*Class for representing "probability" exceptions in the parsing process.*
- class [ParserGrammarUnexpectedTokenException](#)  
*Class for representing "unexpected token" exceptions in the parsing process.*
- class [ParserGrammarUnparseableInputException](#)  
*Class for representing "unparseable input" exceptions in the parsing process.*
- class [SpatialTemporalException](#)  
*Class for representing a spatial temporal exception.*
- class [CommandLineModelChecking](#)  
*Class for running model checkers from the command line.*
- struct [ProbabilityErrorHandler](#)  
*Structure for defining the error handler for invalid probability errors.*
- struct [UnexpectedTokenErrorHandler](#)

*Structure for defining the error handler for unexpected token errors.*

- class [AbstractSyntaxTree](#)  
*Class used for representing an abstract syntax tree.*
- class [Cluster](#)  
*Class for representing a cluster.*
- class [MultiscaleArchitectureGraph](#)  
*Class for defining a multiscale architecture graph.*
- class [NumericStateVariableId](#)  
*Class for representing the identity (name, type) of a numeric state variable.*
- class [Region](#)  
*Class for representing a region.*
- class [ScaleAndSubsystem](#)  
*Class for representing a scale and subsystem.*
- class [SpatialEntity](#)  
*Class for representing a spatial entity.*
- class [SpatialTemporalTrace](#)  
*Class for representing a spatial temporal trace.*
- class [StateVariable](#)  
*Class for representing a state variable.*
- class [TimePoint](#)  
*Class for representing a timepoint.*
- class [BinaryNumericMeasureGrammar](#)  
*The grammar for parsing binary numeric measure statements.*
- class [BinaryStatisticalMeasureGrammar](#)  
*The grammar for parsing binary statistical measure statements.*
- class [BinaryStatisticalQuantileMeasureGrammar](#)  
*The grammar for parsing binary statistical quantile measure statements.*
- class [ChangeMeasureGrammar](#)  
*The grammar for parsing change measure statements.*
- class [ComparatorGrammar](#)  
*The grammar for parsing comparator statements.*
- class [LogicPropertyGrammar](#)  
*The grammar for parsing logic properties.*
- class [NumericStateVariableGrammar](#)  
*The grammar for parsing numeric state variable statements.*
- class [Parser](#)  
*Class used for parsing (P)BLSTL logical queries.*
- class [PrimaryNumericMeasureGrammar](#)  
*The grammar for parsing primary numeric measure statements.*
- class [ScaleAndSubsystemGrammar](#)  
*The grammar for parsing scale and subsystem statements.*
- class [ScaleAndSubsystemStringGrammar](#)  
*The grammar for parsing scale and subsystem string statements.*

- class [SpatialMeasureCollectionGrammar](#)  
*The grammar for parsing spatial measure collection statements.*
- struct [BinaryNumericMeasureTypeParser](#)  
*Symbol table and parser for the binary numeric measure type.*
- struct [BinaryStatisticalMeasureTypeParser](#)  
*Symbol table and parser for the binary statistical measure type.*
- struct [BinaryStatisticalQuantileMeasureTypeParser](#)  
*Symbol table and parser for the binary statistical quantile measure type.*
- struct [ChangeMeasureTypeParser](#)  
*Symbol table and parser for the change measure type.*
- struct [ComparatorNonEqualTypeParser](#)  
*Symbol table and parser for the comparator type which does not accept the "=" symbol.*
- struct [ComparatorTypeParser](#)  
*Symbol table and parser for the comparator type.*
- struct [HeterogeneousTimeseriesComponentTypeParser](#)  
*Symbol table and parser for the heterogeneous timeseries component type.*
- struct [HomogeneousTimeseriesComponentTypeParser](#)  
*Symbol table and parser for the homogeneous timeseries component type.*
- struct [HomogeneousTimeseriesMeasureTypeParser](#)  
*Symbol table and parser for the homogeneous timeseries measure type.*
- struct [SimilarityMeasureTypeParser](#)  
*Symbol table and parser for the similarity measure type.*
- struct [SubsetOperationTypeParser](#)  
*Symbol table and parser for the subset operation type.*
- struct [TimeseriesMeasureTypeParser](#)  
*Symbol table and parser for the timeseries measure type.*
- struct [UnaryNumericMeasureTypeParser](#)  
*Symbol table and parser for the unary numeric measure type.*
- struct [UnaryStatisticalMeasureTypeParser](#)  
*Symbol table and parser for the unary statistical measure type.*
- struct [SpatialMeasureTypeParser](#)  
*Symbol table and parser for the spatial measure type.*
- struct [SubsetSpecificTypeParser](#)  
*Symbol table and parser for a specific subset type.*
- class [TemporalNumericCollectionGrammar](#)  
*The grammar for parsing temporal numeric collection statements.*
- class [TemporalNumericMeasureGrammar](#)  
*The grammar for parsing temporal numeric measure statements.*
- class [UnaryNumericMeasureGrammar](#)  
*The grammar for parsing unary numeric measure statements.*
- class [UnaryStatisticalMeasureGrammar](#)  
*The grammar for parsing unary statistical measure statements.*
- class [ChangeMeasureEvaluator](#)

- class [ComparatorEvaluator](#)

*Class for evaluating change measure expressions.*
- class [ConstraintVisitor](#)

*Class for evaluating comparison expressions.*
- class [FilterNumericVisitor](#)

*Class used to evaluate constraints.*
- class [LogicPropertyVisitor](#)

*Class for evaluating filter numeric measures.*
- class [NumericEvaluator](#)

*Class used to evaluate logic properties.*
- class [NumericMeasureCollectionEvaluator](#)

*Class used to evaluate numeric measure collections.*
- class [NumericMeasureCollectionVisitor](#)

*Class for evaluating numeric measure collections.*
- class [NumericStateVariableEvaluator](#)

*Class used to evaluate numeric state variables.*
- class [NumericVisitor](#)

*Class for evaluating numeric measures.*
- class [ScaleAndSubsystemEvaluator](#)
- class [SpatialMeasureCollectionVisitor](#)

*Class for evaluating spatial measure collections.*
- class [SpatialMeasureEvaluator](#)

*Class for evaluating spatial measures.*
- class [SubsetVisitor](#)

*Class used to evaluate subsets.*
- class [TemporalNumericVisitor](#)

*Class for evaluating temporal numeric measures.*
- class [TimePointEvaluator](#)

*Class used to evaluate timepoints.*
- class [TimeseriesComponentEvaluator](#)

*Class for evaluating timeseries components.*
- class [TimeseriesComponentVisitor](#)

*Class for evaluating timeseries components.*

## Typedefs

- **typedef boost::variant< Nil, boost::recursive\_wrapper < ConstraintAttribute > , boost::recursive\_wrapper < OrConstraintAttribute > , boost::recursive\_wrapper < AndConstraintAttribute > , boost::recursive\_wrapper < ImplicationConstraintAttribute > , boost::recursive\_wrapper < EquivalenceConstraintAttribute > , boost::recursive\_wrapper < PrimaryConstraintAttribute > > - ConstraintAttributeType**

*Variant for a constraint attribute type.*

- `typedef boost::variant < SpatialMeasureAttribute, boost::recursive_wrapper < PrimaryNumericMeasureAttribute > , boost::recursive_wrapper < UnaryNumericFilterAttribute > , boost::recursive_wrapper < BinaryNumericFilterAttribute > , boost::recursive_wrapper < FilterNumericMeasureAttribute > > FilterNumericMeasureAttributeType`

*Variant for a filter numeric measure attribute.*

- `typedef boost::variant< Nil, boost::recursive_wrapper < LogicPropertyAttribute > , boost::recursive_wrapper < OrLogicPropertyAttribute > , boost::recursive_wrapper < AndLogicPropertyAttribute > , boost::recursive_wrapper < -ImplicationLogicPropertyAttribute > , boost::recursive_wrapper < EquivalenceLogicPropertyAttribute > , boost::recursive_wrapper < UntilLogicPropertyAttribute > , boost::recursive_wrapper < PrimaryLogicPropertyAttribute > > LogicPropertyAttributeType`

*Variant for the logic property attribute.*

- `typedef boost::variant< double, NumericStateVariableAttribute, boost::recursive_wrapper < NumericSpatialMeasureAttribute > , boost::recursive_wrapper < PrimaryNumericMeasureAttribute > , boost::recursive_wrapper < UnaryNumericAttribute > , boost::recursive_wrapper < BinaryNumericAttribute > , boost::recursive_wrapper < NumericMeasureAttribute > > NumericMeasureType`

*Variant for the numeric measure attribute.*

- `typedef boost::variant < SpatialMeasureCollectionAttribute, TemporalNumericCollectionAttribute > NumericMeasureCollectionType`

*Variant for the numeric measure collection attribute.*

- `typedef boost::variant < UnaryStatisticalSpatialAttribute, BinaryStatisticalSpatialAttribute, BinaryStatisticalQuantileSpatialAttribute, boost::recursive_wrapper < NumericSpatialMeasureAttribute > > NumericSpatialMeasureType`

*Variant for a numeric spatial measure attribute.*

- `typedef boost::variant < UnaryStatisticalNumericAttribute, BinaryStatisticalNumericAttribute, BinaryStatisticalQuantileNumericAttribute > NumericStatisticalMeasureType`

*Variant for the numeric statistical measure attribute.*

- `typedef boost::variant< Nil, boost::recursive_wrapper < ConstraintAttribute > , boost::recursive_wrapper < NotConstraintAttribute > , boost::recursive_wrapper < UnarySpatialConstraintAttribute > , boost::recursive_wrapper < -UnaryScaleAndSubsystemConstraintAttribute > > PrimaryConstraintAttributeType`

*Variant for a primary constraint attribute.*

- `typedef boost::variant < TemporalNumericComparisonAttribute, ChangeTemporalNumericMeasureAttribute, SimilarityTemporalNumericCollectionAttribute, boost::recursive_wrapper < NotLogicPropertyAttribute > , boost::recursive_wrapper < FutureLogicPropertyAttribute > , boost::recursive_wrapper < GlobalLogicPropertyAttribute > , boost::recursive_wrapper < NextLogicPropertyAttribute > , boost::recursive_wrapper < NextKLogicPropertyAttribute > , boost::recursive_wrapper < LogicPropertyAttribute > > PrimaryLogicPropertyAttributeType`

*Variant for representing a primary logic property type.*

- `typedef boost::variant< double, NumericStateVariableAttribute, boost::recursive_wrapper < NumericSpatialMeasureAttribute >, boost::recursive_wrapper < PrimaryNumericMeasureAttribute >> PrimaryNumericMeasureAttributeType`  
*Variant for the primary numeric measure attribute.*
- `typedef boost::variant < PrimarySpatialMeasureCollectionAttribute, boost::recursive_wrapper < UnaryNumericSpatialAttribute >, boost::recursive_wrapper < BinaryNumericSpatialAttribute >> SpatialMeasureCollectionType`
- `typedef boost::variant < SubsetSpecificAttribute, FilterSubsetAttribute, boost::recursive_wrapper < SubsetSubsetOperationAttribute >, boost::recursive_wrapper < SubsetAttribute >> SubsetAttributeType`  
*Variant for a subset attribute.*
- `typedef boost::variant < TemporalNumericMeasureCollectionAttribute, boost::recursive_wrapper < ChangeTemporalNumericCollectionAttribute >, boost::recursive_wrapper < TimeseriesTimeseriesComponentAttribute >, boost::recursive_wrapper < HomogeneousHomogeneousTimeseriesAttribute >> TemporalNumericCollectionType`  
*Variant for the temporal numeric collection attribute.*
- `typedef boost::variant< double, NumericStateVariableAttribute, NumericStatisticalMeasureAttribute, boost::recursive_wrapper < UnaryNumericTemporalAttribute >, boost::recursive_wrapper < BinaryNumericTemporalAttribute >, boost::recursive_wrapper < TemporalNumericMeasureAttribute >> TemporalNumericMeasureType`  
*Variant for the temporal numeric measure attribute.*
- `typedef boost::variant < HeterogeneousTimeseriesComponentAttribute, × HomogeneousTimeseriesComponentAttribute > TimeseriesComponentType`  
*Variant for the timeseries component attribute.*

## Enumerations

- `enum BinaryNumericMeasureType { Add = 0, Div, Log, Mod, Multiply, Power, Subtract }`  
*Enumeration for representing a binary numeric measure type.*
- `enum BinaryStatisticalMeasureType { Covar = 0 }`  
*Enumeration for representing a binary statistical measure type.*
- `enum BinaryStatisticalQuantileMeasureType { Percentile = 0, Quartile }`  
*Enumeration for representing a binary statistical quantile measure type.*
- `enum ChangeMeasureType { Derivative = 0, Ratio }`  
*Enumeration for representing a change measure type.*
- `enum ComparatorType { GreaterThan = 0, GreaterThanOrEqual, LessThan, - LessThanOrEqual, Equal }`  
*Enumeration for representing a comparator type.*
- `enum HeterogeneousTimeseriesComponentType { Peak = 0, Valley }`  
*Enumeration for representing a heterogeneous timeseries component type.*
- `enum HomogeneousTimeseriesComponentType { Ascent = 0, Descent, Plateau, UniformAscent, UniformDescent }`

- enum **HomogeneousTimeseriesMeasureType** { **TimeSpan** = 0, **Values** }
  - Enumeration for representing a homogeneous timeseries component type.*
- enum **SimilarityMeasureType** { **Opposite** = 0, **Similar** }
  - Enumeration for representing a homogeneous timeseries measure type.*
- enum **SpatialMeasureType** { **Area** = 0, **Perimeter**, **Clusteredness**, **Density**, **DistanceFromOrigin**, **Angle**, **TriangleMeasure**, **RectangleMeasure**, **CircleMeasure**, **CentroidX**, **CentroidY**, **NrOfSpatialMeasureTypeEntries** }
  - Enumeration for representing the types of spatial measures.*
- enum **SubsetOperationType** { **Difference** = 0, **Intersection**, **Union** }
  - Enumeration for representing the types of subset operations.*
- enum **SubsetSpecificType** { **Clusters** = 0, **Regions**, **NrOfSubsetSpecificTypeEntries** }
  - Enumeration for representing a specific subset type.*
- enum **TimeseriesMeasureType** { **EnteringTime** = 0, **EnteringValue** }
  - Enumeration for representing a timeseries measure type.*
- enum **UnaryNumericMeasureType** { **Abs** = 0, **Ceil**, **Floor**, **Round**, **Sign**, **Sqrt**, **Trunc** }
  - Enumeration for representing a unary numeric measure type.*
- enum **UnaryStatisticalMeasureType** { **Avg** = 0, **Count**, **Geomean**, **Harmean**, **Kurt**, **Max**, **Median**, **Min**, **Mode**, **Product**, **Skew**, **Stdev**, **Sum**, **Var** }
  - Enumeration for representing a unary statistical measure type.*
- enum **ApproximateBayesianModelCheckingResult** { **TRUE** = 0, **FALSE**, **MORE\_TRACES\_REQUIRED** }
  - Enumeration for representing the model checking result.*
- enum **BayesianModelCheckingResult** { **TRUE** = 0, **FALSE**, **MORE\_TRACES\_REQUIRED** }
  - Enumeration for representing the model checking result.*
- enum **StatisticalModelCheckingResult** { **TRUE** = 0, **FALSE**, **UNDECIDED**, **MORE\_TRACES\_REQUIRED** }
  - Enumeration for representing the model checking result.*

## Functions

- std::ostream & **operator<<** (std::ostream &out, const **BinaryNumericMeasureType** &binaryNumericMeasureType)
  - Overload the output stream operator for the enumeration.*
- std::ostream & **operator<<** (std::ostream &out, const **BinaryStatisticalMeasureType** &binaryStatisticalMeasureType)
  - Overload the output stream operator for the enumeration.*
- std::ostream & **operator<<** (std::ostream &out, const **BinaryStatisticalQuantileMeasureType** &binaryStatisticalQuantileMeasureType)
  - Overload the output stream operator for the enumeration.*
- std::ostream & **operator<<** (std::ostream &out, const **ChangeMeasureType** &changeMeasureType)
  - Overload the output stream operator for the enumeration.*

- std::ostream & `operator<<` (std::ostream &out, const ComparatorType &comparatorType)
 

Overload the output stream operator for the enumeration.
  - std::ostream & `operator<<` (std::ostream &out, const HeterogeneousTimeseries-ComponentType &heterogeneousTimeseriesComponentType)
 

Overload the output stream operator for the enumeration.
  - std::ostream & `operator<<` (std::ostream &out, const HomogeneousTimeseries-ComponentType &homogeneousTimeseriesComponentType)
 

Overload the output stream operator for the enumeration.
  - std::ostream & `operator<<` (std::ostream &out, const HomogeneousTimeseries-MeasureType &homogeneousTimeseriesMeasureType)
 

Overload the output stream operator for the enumeration.
  - std::ostream & `operator<<` (std::ostream &out, const SimilarityMeasureType &similarityMeasureType)
 

Overload the output stream operator for the enumeration.
  - std::ostream & `operator<<` (std::ostream &out, const SpatialMeasureType &spatialMeasureType)
 

Overload the output stream operator for the enumeration.
  - std::ostream & `operator<<` (std::ostream &out, const SubsetOperationType &subsetOperationType)
 

Overload the output stream operator for the enumeration.
  - std::ostream & `operator<<` (std::ostream &out, const SubsetSpecificType &subsetSpecificType)
 

Overload the output stream operator for the enumeration.
  - std::ostream & `operator<<` (std::ostream &out, const TimeseriesMeasureType &timeseriesMeasureType)
 

Overload the output stream operator for the enumeration.
  - std::ostream & `operator<<` (std::ostream &out, const UnaryNumericMeasure-Type &unaryNumericMeasureType)
 

Overload the output stream operator for the enumeration.
  - std::ostream & `operator<<` (std::ostream &out, const UnaryStatisticalMeasure-Type &unaryStatisticalMeasureType)
 

Overload the output stream operator for the enumeration.
- Overload the output stream operator for the enumeration.*

## Variables

- static const std::size\_t `NR_SPATIAL_MEASURE_TYPES` = static\_cast<std::size\_t>(SpatialMeasureType::NrOfSpatialMeasureTypeEntries)
 

An size\_t constant which stores the number of spatial measure type entries.
- static const std::size\_t `NR_SUBSET_SPECIFIC_TYPES` = static\_cast<std::size\_t>(SubsetSpecificType::NrOfSubsetSpecificTypeEntries)
 

An size\_t constant which stores the number of subset specific type entries.
- phoenix::function <`UnexpectedTokenErrorHandler`> const handleUnexpected-  
`TokenError` = `UnexpectedTokenErrorHandler()`

- phoenix::function < ProbabilityErrorHandler > const handleProbabilityError = - ProbabilityErrorHandler()
- static const std::string WRN\_LOGIC\_PROPERTY\_EVAL\_FALSE = "The enclosing logic property was evaluated to the default value \"false\"."
- static const std::string WRN\_OUTPUT\_SEPARATOR = " "

### 5.3.1 Typedef Documentation

5.3.1.1 `typedef boost::variant< Nil, boost::recursive_wrapper<ConstraintAttribute>, boost::recursive_wrapper<OrConstraintAttribute>, boost::recursive_wrapper<AndConstraintAttribute>, boost::recursive_wrapper<ImplicationConstraintAttribute>, boost::recursive_wrapper<EquivalenceConstraintAttribute>, boost::recursive_wrapper<PrimaryConstraintAttribute> > multiscale::verification::ConstraintAttributeType`

Variant for a constraint attribute type.

Definition at line 20 of file ConstraintAttribute.hpp.

5.3.1.2 `typedef boost::variant< SpatialMeasureAttribute, boost::recursive_wrapper<PrimaryNumericMeasureAttribute>, boost::recursive_wrapper<UnaryNumericFilterAttribute>, boost::recursive_wrapper<BinaryNumericFilterAttribute>, boost::recursive_wrapper<FilterNumericMeasureAttribute> > multiscale::verification::FilterNumericMeasureAttributeType`

Variant for a filter numeric measure attribute.

Definition at line 18 of file FilterNumericMeasureAttribute.hpp.

5.3.1.3 `typedef boost::variant< Nil, boost::recursive_wrapper<LogicPropertyAttribute>, boost::recursive_wrapper<OrLogicPropertyAttribute>, boost::recursive_wrapper<AndLogicPropertyAttribute>, boost::recursive_wrapper<ImplicationLogicPropertyAttribute>, boost::recursive_wrapper<EquivalenceLogicPropertyAttribute>, boost::recursive_wrapper<UntilLogicPropertyAttribute>, boost::recursive_wrapper<PrimaryLogicPropertyAttribute> > multiscale::verification::LogicPropertyAttributeType`

Variant for the logic property attribute.

Definition at line 23 of file LogicPropertyAttribute.hpp.

```
5.3.1.4  typedef boost::variant< SpatialMeasureCollection-
Attribute, TemporalNumericCollectionAttribute >
multiscale::verification::NumericMeasureCollectionType
```

Variant for the numeric measure collection attribute.

Definition at line 18 of file NumericMeasureCollectionAttribute.hpp.

```
5.3.1.5  typedef boost::variant< double, NumericStateVariableAttribute,
boost::recursive_wrapper<NumericSpatialMeasureAttribute>,
boost::recursive_wrapper<PrimaryNumericMeasureAttribute>,
boost::recursive_wrapper<UnaryNumericNumericAttribute>,
boost::recursive_wrapper<BinaryNumericNumericAttribute>,
boost::recursive_wrapper<NumericMeasureAttribute> >
multiscale::verification::NumericMeasureType
```

Variant for the numeric measure attribute.

Definition at line 19 of file NumericMeasureAttribute.hpp.

```
5.3.1.6  typedef boost::variant< UnaryStatisticalSpatialAttribute, Binary-
StatisticalSpatialAttribute, BinaryStatisticalQuantileSpatialAttribute,
boost::recursive_wrapper<NumericSpatialMeasureAttribute> >
multiscale::verification::NumericSpatialMeasureType
```

Variant for a numeric spatial measure attribute.

Definition at line 16 of file NumericSpatialMeasureAttribute.hpp.

```
5.3.1.7  typedef boost::variant< UnaryStatisticalNumericAttribute, BinaryStatistical-
NumericAttribute, BinaryStatisticalQuantileNumericAttribute >
multiscale::verification::NumericStatisticalMeasureType
```

Variant for the numeric statistical measure attribute.

Definition at line 20 of file NumericStatisticalMeasureAttribute.hpp.

```
5.3.1.8  typedef boost::variant< Nil, boost::recursive_wrapper<ConstraintAttribute>,
boost::recursive_wrapper<NotConstraintAttribute>,
boost::recursive_wrapper<UnarySpatialConstraintAttribute>,
boost::recursive_wrapper<UnaryScaleAndSubsystemConstraintAttribute> >
multiscale::verification::PrimaryConstraintAttributeType
```

Variant for a primary constraint attribute.

Definition at line 18 of file PrimaryConstraintAttribute.hpp.

```
5.3.1.9  typedef boost::variant< TemporalNumericComparisonAttribute,
                           ChangeTemporalNumericMeasureAttribute, SimilarityTemporalNumeric-
                           CollectionAttribute, boost::recursive_wrapper<NotLogicPropertyAttribute>,
                           boost::recursive_wrapper<FutureLogicPropertyAttribute>,
                           boost::recursive_wrapper<GlobalLogicPropertyAttribute>,
                           boost::recursive_wrapper<NextLogicPropertyAttribute>,
                           boost::recursive_wrapper<NextKLogicPropertyAttribute>,
                           boost::recursive_wrapper<LogicPropertyAttribute> >
                           multiscale::verification::PrimaryLogicPropertyAttributeType
```

Variant for representing a primary logic property type.

Definition at line 22 of file PrimaryLogicPropertyAttribute.hpp.

```
5.3.1.10  typedef boost::variant< double, NumericStateVariableAttribute,
                           boost::recursive_wrapper<NumericSpatialMeasureAttribute>,
                           boost::recursive_wrapper<PrimaryNumericMeasureAttribute> >
                           multiscale::verification::PrimaryNumericMeasureAttributeType
```

Variant for the primary numeric measure attribute.

Definition at line 15 of file PrimaryNumericMeasureAttribute.hpp.

```
5.3.1.11  typedef boost::variant< PrimarySpatialMeasureCollectionAttribute,
                           boost::recursive_wrapper<UnaryNumericSpatialAttribute>,
                           boost::recursive_wrapper<BinaryNumericSpatialAttribute> >
                           multiscale::verification::SpatialMeasureCollectionType
```

Definition at line 16 of file SpatialMeasureCollectionAttribute.hpp.

```
5.3.1.12  typedef boost::variant< SubsetSpecificAttribute, FilterSubsetAttribute,
                           boost::recursive_wrapper<SubsetSubsetOperation-
                           Attribute>, boost::recursive_wrapper<SubsetAttribute> >
                           multiscale::verification::SubsetAttributeType
```

Variant for a subset attribute.

Definition at line 16 of file SubsetAttribute.hpp.

```
5.3.1.13  typedef boost::variant< TemporalNumericMeasureCollectionAttribute,
                           boost::recursive_wrapper<ChangeTemporalNumericCollectionAttribute>,
                           boost::recursive_wrapper<TimeseriesTimeseriesComponentAttribute>,
                           boost::recursive_wrapper<HomogeneousHomogeneousTimeseries-
                           Attribute> > multiscale::verification::TemporalNumericCollectionType
```

Variant for the temporal numeric collection attribute.

Definition at line 16 of file TemporalNumericCollectionAttribute.hpp.

---

---

```
5.3.1.14 typedef boost::variant< double, NumericStateVariable-
Attribute, NumericStatisticalMeasureAttribute,
boost::recursive_wrapper<UnaryNumericTemporalAttribute>,
boost::recursive_wrapper<BinaryNumericTemporalAttribute>,
boost::recursive_wrapper<TemporalNumericMeasureAttribute> >
multiscale::verification::TemporalNumericMeasureType
```

Variant for the temporal numeric measure attribute.

Definition at line 17 of file TemporalNumericMeasureAttribute.hpp.

```
5.3.1.15 typedef boost::variant< HeterogeneousTimeseriesComponentAttribute,
HomogeneousTimeseriesComponentAttribute >
multiscale::verification::TimeseriesComponentType
```

Variant for the timeseries component attribute.

Definition at line 19 of file TimeseriesComponentAttribute.hpp.

### 5.3.2 Enumeration Type Documentation

**5.3.2.1 enum multiscale::verification::ApproximateBayesianModelCheckingResult**

Enumeration for representing the model checking result.

Enumerator:

**TRUE** The logic property was evaluated to true

**FALSE** The logic property was evaluated to false

**MORE\_TRACES\_REQUIRED** More traces are required to determine the truth value of the logic property

Definition at line 15 of file ApproximateBayesianModelChecker.hpp.

**5.3.2.2 enum multiscale::verification::BayesianModelCheckingResult**

Enumeration for representing the model checking result.

Enumerator:

**TRUE** The logic property was evaluated to true

**FALSE** The logic property was evaluated to false

**MORE\_TRACES\_REQUIRED** More traces are required to determine the truth value of the logic property

Definition at line 15 of file BayesianModelChecker.hpp.

### 5.3.2.3 enum multiscale::verification::BinaryNumericMeasureType

Enumeration for representing a binary numeric measure type.

Enumerator:

- Add** Addition
- Div** Division
- Log** Logarithm
- Mod** Remainder of division
- Multiply** Multiplication
- Power** Raise to power
- Subtract** Subtraction

Definition at line 13 of file BinaryNumericMeasureAttribute.hpp.

### 5.3.2.4 enum multiscale::verification::BinaryStatisticalMeasureType

Enumeration for representing a binary statistical measure type.

Enumerator:

- Covar** Covariance

Definition at line 13 of file BinaryStatisticalMeasureAttribute.hpp.

### 5.3.2.5 enum multiscale::verification::BinaryStatisticalQuantileMeasureType

Enumeration for representing a binary statistical quantile measure type.

Enumerator:

- Percentile** The percentile
- Quartile** The quartile

Definition at line 13 of file BinaryStatisticalQuantileMeasureAttribute.hpp.

### 5.3.2.6 enum multiscale::verification::ChangeMeasureType

Enumeration for representing a change measure type.

Enumerator:

- Derivative** Derivative representing rate of change
- Ratio** Ratio of value change over difference in time change

Definition at line 13 of file ChangeMeasureAttribute.hpp.

---

### 5.3.2.7 enum multiscale::verification::ComparatorType

Enumeration for representing a comparator type.

Enumerator:

- GreaterThan** Greater than
- GreaterThanOrEqual** Greater than or equal
- LessThan** Less than
- LessThanOrEqual** Less than or equal
- Equal** Equal

Definition at line 13 of file ComparatorAttribute.hpp.

### 5.3.2.8 enum multiscale::verification::HeterogeneousTimeseriesComponentType

Enumeration for representing a heterogeneous timeseries component type.

Enumerator:

- Peak** The peak of the timeseries
- Valley** The value of the timeseries

Definition at line 13 of file HeterogeneousTimeseriesComponentAttribute.hpp.

### 5.3.2.9 enum multiscale::verification::HomogeneousTimeseriesComponentType

Enumeration for representing a homogeneous timeseries component type.

Enumerator:

- Ascent** The ascending timeseries type
- Descent** The descending timeseries type
- Plateau** The plateau timeseries type
- UniformAscent** The uniformly (constantly) ascending timeseries type
- UniformDescent** The uniformly (constantly) descending timeseries type

Definition at line 13 of file HomogeneousTimeseriesComponentAttribute.hpp.

### 5.3.2.10 enum multiscale::verification::HomogeneousTimeseriesMeasureType

Enumeration for representing a homogeneous timeseries measure type.

Enumerator:

- TimeSpan** The time span covered by the homogeneous timeseries
- Values** The values defining the homogeneous timeseries

Definition at line 13 of file HomogeneousTimeseriesMeasureAttribute.hpp.

### 5.3.2.11 enum multiscale::verification::SimilarityMeasureType

Enumeration for representing a similarity measure type.

Enumerator:

**Opposite** The opposite type

**Similar** The similar type

Definition at line 13 of file SimilarityMeasureAttribute.hpp.

### 5.3.2.12 enum multiscale::verification::SpatialMeasureType

Enumeration for representing the types of spatial measures.

Enumerator:

**Area** The area of a spatial entity

**Perimeter** The perimeter of a spatial entity

**Clusteredness** The clusteredness of a spatial entity

**Density** The density of a spatial entity

**DistanceFromOrigin** The distance of a spatial entity from origin

**Angle** The angle

**TriangleMeasure** The measure indicating how triangular is the shape of the spatial entity

**RectangleMeasure** The measure indicating how rectangular is the shape of the spatial entity

**CircleMeasure** The measure indicating how circular is the shape of the spatial entity

**CentroidX** The x coordinate of the spatial entity centroid

**CentroidY** The y coordinate of the spatial entity centroid

**NrOfSpatialMeasureTypeEntries** Enumeration type used to store the number of elements in the enumeration. Always leave it last!

Definition at line 19 of file SpatialMeasureType.hpp.

### 5.3.2.13 enum multiscale::verification::StatisticalModelCheckingResult

Enumeration for representing the model checking result.

Enumerator:

**TRUE** The logic property was evaluated to true

**FALSE** The logic property was evaluated to false

**UNDECIDED** The truth value of the logic property is undecided

**MORE\_TRACES\_REQUIRED** More traces are required to determine the truth value of the logic property

Definition at line 15 of file StatisticalModelChecker.hpp.

#### 5.3.2.14 enum multiscale::verification::SubsetOperationType

Enumeration for representing the types of subset operations.

Enumerator:

**Difference** Difference of two subsets

**Intersection** Intersection of two subsets

**Union** Union of two subsets

Definition at line 13 of file SubsetOperationAttribute.hpp.

#### 5.3.2.15 enum multiscale::verification::SubsetSpecificType

Enumeration for representing a specific subset type.

Enumerator:

**Clusters** Clusters

**Regions** Regions

**NrOfSubsetSpecificTypeEntries** Enumeration type used to store the number of elements in the enumeration. Always leave it last!

Definition at line 19 of file SubsetSpecificType.hpp.

#### 5.3.2.16 enum multiscale::verification::TimeseriesMeasureType

Enumeration for representing a timeseries measure type.

Enumerator:

**EnteringTime** The entering time

**EnteringValue** The entering value

Definition at line 13 of file TimeseriesMeasureAttribute.hpp.

#### 5.3.2.17 enum multiscale::verification::UnaryNumericMeasureType

Enumeration for representing a unary numeric measure type.

Enumerator:

**Abs** Absolute value  
**Ceil** Ceiling  
**Floor** Floor  
**Round** Round  
**Sign** Sign: -1 (-), +1 (+) or 0 (0)  
**Sqrt** Square root  
**Trunc** Truncation

Definition at line 13 of file UnaryNumericMeasureAttribute.hpp.

### 5.3.2.18 enum multiscale::verification::UnaryStatisticalMeasureType

Enumeration for representing a unary statistical measure type.

Enumerator:

**Avg** The average (arithmetic mean)  
**Count** The cardinality of a collection  
**Geomean** The geometric mean  
**Harmean** The harmonic mean  
**Kurt** The kurtosis  
**Max** The maximum  
**Median** The median  
**Min** The minimum  
**Mode** The mode  
**Product** The product  
**Skew** The skew  
**Stdev** The standard deviation  
**Sum** The sum  
**Var** The variance

Definition at line 13 of file UnaryStatisticalMeasureAttribute.hpp.

### 5.3.3 Function Documentation

#### 5.3.3.1 std::ostream & multiscale::verification::operator<< ( std::ostream & out, const BinaryStatisticalMeasureType & binaryStatisticalMeasureType )

Overload the output stream operator for the enumeration.

##### Parameters

---

<i>out</i>	Output stream
<i>binary-Statistical-Measure-Type</i>	The binary statistical measure type to be printed out

Definition at line 8 of file BinaryStatisticalMeasureAttribute.cpp.

**5.3.3.2 std::ostream & multiscale::verification::operator<< ( std::ostream & *out*, const ChangeMeasureType & *changeMeasureType* )**

Overload the output stream operator for the enumeration.

#### Parameters

<i>out</i>	Output stream
<i>change-Measure-Type</i>	The change measure type to be printed out

Definition at line 5 of file ChangeMeasureAttribute.cpp.

**5.3.3.3 std::ostream & multiscale::verification::operator<< ( std::ostream & *out*, const HomogeneousTimeseriesMeasureType & *homogeneousTimeseriesMeasureType* )**

Overload the output stream operator for the enumeration.

#### Parameters

<i>out</i>	Output stream
<i>homogeneous-Timeseries-Measure-Type</i>	The homogeneous timeseries measure type to be printed out

Definition at line 8 of file HomogeneousTimeseriesMeasureAttribute.cpp.

**5.3.3.4 std::ostream & multiscale::verification::operator<< ( std::ostream & *out*, const SimilarityMeasureType & *similarityMeasureType* )**

Overload the output stream operator for the enumeration.

**Parameters**

<i>out</i>	Output stream
<i>similarity-Measure-Type</i>	The similarity measure type to be printed out

Definition at line 8 of file SimilarityMeasureAttribute.cpp.

**5.3.3.5 std::ostream & multiscale::verification::operator<< ( std::ostream & *out*, const HeterogeneousTimeseriesComponentType & *heterogeneousTimeseriesComponentType* )**

Overload the output stream operator for the enumeration.

**Parameters**

<i>out</i>	Output stream
<i>heterogeneous-Timeseries-Component-Type</i>	The heterogeneous timeseries component type to be printed out

Definition at line 8 of file HeterogeneousTimeseriesComponentAttribute.cpp.

**5.3.3.6 std::ostream & multiscale::verification::operator<< ( std::ostream & *out*, const TimeseriesMeasureType & *timeseriesMeasureType* )**

Overload the output stream operator for the enumeration.

**Parameters**

<i>out</i>	Output stream
<i>timeseries-Measure-Type</i>	The timeseries measure type to be printed out

Definition at line 8 of file TimeseriesMeasureAttribute.cpp.

**5.3.3.7 std::ostream & multiscale::verification::operator<< ( std::ostream & *out*, const BinaryStatisticalQuantileMeasureType & *binaryStatisticalQuantileMeasureType* )**

Overload the output stream operator for the enumeration.

**Parameters**

<i>out</i>	Output stream
------------	---------------

<i>binary-Statistical-Quantile-Measure-Type</i>	The binary statistical quantile measure type to be printed out
---	--

Definition at line 8 of file BinaryStatisticalQuantileMeasureAttribute.cpp.

**5.3.3.8 std::ostream & multiscale::verification::operator<< ( std::ostream & *out*, const SubsetOperationType & *subsetOperationType* )**

Overload the output stream operator for the enumeration.

#### Parameters

<i>out</i>	Output stream
<i>subset-Operation-Type</i>	The subset operation type to be printed out

Definition at line 7 of file SubsetOperationAttribute.cpp.

**5.3.3.9 std::ostream & multiscale::verification::operator<< ( std::ostream & *out*, const ComparatorType & *comparatorType* )**

Overload the output stream operator for the enumeration.

#### Parameters

<i>out</i>	Output stream
<i>comparator-Type</i>	The comparator type to be printed out

Definition at line 7 of file ComparatorAttribute.cpp.

**5.3.3.10 std::ostream & multiscale::verification::operator<< ( std::ostream & *out*, const HomogeneousTimeseriesComponentType & *homogeneousTimeseriesComponentType* )**

Overload the output stream operator for the enumeration.

**Parameters**

<i>out</i>	Output stream
<i>homogeneous-Timeseries-Component-Type</i>	The homogeneous timeseries component type to be printed out

Definition at line 8 of file HomogeneousTimeseriesComponentAttribute.cpp.

**5.3.3.11 std::ostream & multiscale::verification::operator<< ( std::ostream & *out*, const UnaryNumericMeasureType & *unaryNumericMeasureType* )**

Overload the output stream operator for the enumeration.

**Parameters**

<i>out</i>	Output stream
<i>unary-Numeric-Measure-Type</i>	The unary numeric measure type to be printed out

Definition at line 7 of file UnaryNumericMeasureAttribute.cpp.

**5.3.3.12 std::ostream & multiscale::verification::operator<< ( std::ostream & *out*, const BinaryNumericMeasureType & *binaryNumericMeasureType* )**

Overload the output stream operator for the enumeration.

**Parameters**

<i>out</i>	Output stream
<i>binary-Numeric-Measure-Type</i>	The binary numeric measure type to be printed out

Definition at line 7 of file BinaryNumericMeasureAttribute.cpp.

**5.3.3.13 std::ostream & multiscale::verification::operator<< ( std::ostream & *out*, const UnaryStatisticalMeasureType & *unaryStatisticalMeasureType* )**

Overload the output stream operator for the enumeration.

**Parameters**

<i>out</i>	Output stream
<i>unary- Statistical- Measure- Type</i>	The unary statistical measure type to be printed out

Definition at line 7 of file UnaryStatisticalMeasureAttribute.cpp.

**5.3.3.14 std::ostream & multiscale::verification::operator<< ( std::ostream & *out*, const SubsetSpecificType & *subsetSpecificType* )**

Overload the output stream operator for the enumeration.

Overload the output stream operator for the SubsetSpecificType enumeration.

**Parameters**

<i>out</i>	Output stream
<i>subset- SpecificType</i>	The specific subset type to be printed out

Definition at line 52 of file SubsetSpecificAttributeAutoGenerated.cpp.

**5.3.3.15 std::ostream & multiscale::verification::operator<< ( std::ostream & *out*, const SpatialMeasureType & *spatialMeasureType* )**

Overload the output stream operator for the enumeration.

Overload the output stream operator for the SpatialMeasureType enumeration.

**Parameters**

<i>out</i>	Output stream
<i>spatial- Measure- Type</i>	The spatial measure type to be printed out

Definition at line 219 of file SpatialMeasureAttributeAutoGenerated.cpp.

### 5.3.4 Variable Documentation

**5.3.4.1 phoenix::function<ProbabilityErrorHandler> const multiscale::verification::handleProbabilityError = ProbabilityErrorHandler()**

Definition at line 21 of file LogicPropertyGrammarDefinition.hpp.

Referenced by multiscale::verification::LogicPropertyGrammar< Iterator >::initialiseProbabilisticLogicPropertyErrorHandlerSupport().

**5.3.4.2 phoenix::function< UnexpectedTokenErrorHandler > const multiscale::verification::handleUnexpectedTokenError = UnexpectedTokenErrorHandler()**

Definition at line 20 of file ChangeMeasureGrammarDefinition.hpp.

Referenced by multiscale::verification::SpatialMeasureCollectionGrammar< Iterator >::initialiseComposedConstraintErrorHandlingSupport(), multiscale::verification::LogicPropertyGrammar< Iterator >::initialiseComposedLogicPropertyErrorHandlerSupport(), multiscale::verification::ScaleAndSubsystemStringGrammar< Iterator >::initialiseErrorHandlingSupport(), multiscale::verification::NumericStateVariableGrammar< Iterator >::initialiseErrorHandlingSupport(), multiscale::verification::SpatialMeasureCollectionGrammar< Iterator >::initialiseFilterNumericMeasureErrorHandlerSupport(), multiscale::verification::TemporalNumericCollectionGrammar< Iterator >::initialiseNumericMeasureErrorHandlerSupport(), multiscale::verification::PrimaryNumericMeasureGrammar< Iterator >::initialiseNumericSpatialMeasureErrorHandlerSupport(), multiscale::verification::TemporalNumericMeasureGrammar< Iterator >::initialiseNumericStatisticalMeasureErrorHandlerSupport(), multiscale::verification::SpatialMeasureCollectionGrammar< Iterator >::initialisePrimaryConstraintErrorHandlingSupport(), multiscale::verification::LogicPropertyGrammar< Iterator >::initialisePrimaryLogicPropertyErrorHandlerSupport(), multiscale::verification::LogicPropertyGrammar< Iterator >::initialiseProbabilisticLogicPropertyErrorHandlerSupport(), multiscale::verification::SpatialMeasureCollectionGrammar< Iterator >::initialiseSpatialMeasureCollectionErrorHandlerSupport(), multiscale::verification::SpatialMeasureCollectionGrammar< Iterator >::initialiseSubsetErrorHandlerSupport(), multiscale::verification::TemporalNumericCollectionGrammar< Iterator >::initialiseTemporalNumericCollectionErrorHandlerSupport(), and multiscale::verification::TemporalNumericMeasureGrammar< Iterator >::initialiseTemporalNumericMeasureErrorHandlerSupport().

**5.3.4.3 const std::size\_t multiscale::verification::NR\_SPATIAL\_MEASURE\_TYPES = static\_cast<std::size\_t>(SpatialMeasureType::NrOfSpatialMeasureTypeEntries) [static]**

An size\_t constant which stores the number of spatial measure type entries.

Definition at line 16 of file SpatialMeasureAttribute.hpp.

Referenced by multiscale::verification::spatialmeasure::validateSpatialMeasureTypeIndex().

**5.3.4.4 const std::size\_t multiscale::verification::NR\_SUBSET\_SPECIFIC\_TYPES = static\_cast<std::size\_t>(SubsetSpecificType::NrOfSubsetSpecificTypeEntries) [static]**

An size\_t constant which stores the number of subset specific type entries.

Definition at line 16 of file SubsetSpecificAttribute.hpp.

Referenced by multiscale::verification::MSTMLSubfilesMerger::addSpatialEntitiesToResultingTraceTimepoint(), multiscale::verification::TimePoint::areEqualSpatialEntities(), multiscale::verification::ConstraintVisitor::evaluateScaleAndSubsystemConstraint(), multiscale::verification::ConstraintVisitor::evaluateSpatialMeasureConstraint(), multiscale::verification::TimePoint::getConsideredSpatialEntities(), multiscale::verification::TimePoint::numberOfSpatialEntities(), multiscale::verification::TimePoint::updateSpatialEntities(), and multiscale::verification::subsetsspecific::validateSubsetSpecificTypeIndex().

**5.3.4.5 const std::string multiscale::verification::WRN\_LOGIC\_PROPERTY\_EVAL\_FALSE = "The enclosing logic property was evaluated to the default value \"false\"."**  
`[static]`

Definition at line 19 of file LogicPropertyVisitor.hpp.

Referenced by multiscale::verification::LogicPropertyVisitor::printExceptionMessage().

**5.3.4.6 const std::string multiscale::verification::WRN\_OUTPUT\_SEPARATOR = ""**  
`[static]`

Definition at line 20 of file LogicPropertyVisitor.hpp.

Referenced by multiscale::verification::LogicPropertyVisitor::printExceptionMessage().

## 5.4 multiscale::verification::spatialmeasure Namespace Reference

### Functions

- void [validateSpatialMeasureType](#) (const [SpatialMeasureType](#) &spatialMeasureType)  
*Check if the given spatial measure type is valid.*
- void [validateSpatialMeasureTypeIndex](#) (const std::size\_t &spatialMeasureTypeIndex)  
*Check if the given spatial measure type index is valid.*
- size\_t [computeSpatialMeasureTypeIndex](#) (const [SpatialMeasureType](#) &spatialMeasureType)  
*Compute the index of the spatial measure type.*
- [SpatialMeasureType](#) [computeSpatialMeasureType](#) (const std::size\_t &spatialMeasureTypeIndex)  
*Compute the spatial measure type from the given index.*
- double [getMinValidSpatialMeasureValue](#) (const [SpatialMeasureType](#) &spatialMeasureType)  
*Get the minimum valid value for the given spatial measure type.*
- double [getMaxValidSpatialMeasureValue](#) (const [SpatialMeasureType](#) &spatialMeasureType)

*Get the maximum valid value for the given spatial measure type.*

- std::string **toString** (const [SpatialMeasureType](#) &spatialMeasureType)

*Get the string representation of the given spatial measure type.*

- std::string **toString** (const std::size\_t &spatialMeasureTypeIndex)

*Get the string representation corresponding to the the given spatial measure type index.*

#### 5.4.1 Function Documentation

##### 5.4.1.1 [SpatialMeasureType multiscale::verification::spatialmeasure::computeSpatialMeasureType \( const std::size\\_t & spatialMeasureTypeIndex \)](#)

Compute the spatial measure type from the given index.

###### Parameters

<i>spatial- Measure- TypeIndex</i>	The given spatial measure type index
--	--------------------------------------

Definition at line 33 of file SpatialMeasureAttribute.cpp.

References validateSpatialMeasureTypeIndex().

##### 5.4.1.2 [size\\_t multiscale::verification::spatialmeasure::computeSpatialMeasureTypeIndex \( const \[SpatialMeasureType\]\(#\) & spatialMeasureType \)](#)

Compute the index of the spatial measure type.

###### Parameters

<i>spatial- Measure- Type</i>	The given spatial measure type
---------------------------------------	--------------------------------

Definition at line 26 of file SpatialMeasureAttribute.cpp.

References validateSpatialMeasureType().

Referenced by multiscale::verification::SpatialEntity::getSpatialMeasureValue(), and multiscale::verification::SpatialEntity::setSpatialMeasureValue().

##### 5.4.1.3 [double multiscale::verification::spatialmeasure::getMaxValidSpatialMeasureValue \( const \[SpatialMeasureType\]\(#\) & spatialMeasureType \)](#)

Get the maximum valid value for the given spatial measure type.

**Parameters**

<i>spatial- Measure- Type</i>	The given spatial measure type
---------------------------------------	--------------------------------

Definition at line 75 of file SpatialMeasureAttributeAutoGenerated.cpp.

References multiscale::ERR\_UNDEFINED\_ENUM\_VALUE.

**5.4.1.4 double multiscale::verification::spatialmeasure::getMinValidSpatialMeasureValue ( const SpatialMeasureType & *spatialMeasureType* )**

Get the minimum valid value for the given spatial measure type.

**Parameters**

<i>spatial- Measure- Type</i>	The given spatial measure type
---------------------------------------	--------------------------------

Definition at line 20 of file SpatialMeasureAttributeAutoGenerated.cpp.

References multiscale::ERR\_UNDEFINED\_ENUM\_VALUE.

**5.4.1.5 std::string multiscale::verification::spatialmeasure::toString ( const SpatialMeasureType & *spatialMeasureType* )**

Get the string representation of the given spatial measure type.

**Parameters**

<i>spatial- Measure- Type</i>	The given spatial measure type
---------------------------------------	--------------------------------

Definition at line 130 of file SpatialMeasureAttributeAutoGenerated.cpp.

References multiscale::ERR\_UNDEFINED\_ENUM\_VALUE.

**5.4.1.6 std::string multiscale::verification::spatialmeasure::toString ( const std::size\_t & *spatialMeasureTypeIndex* )**

Get the string representation corresponding to the the given spatial measure type index.

**Parameters**

<i>spatial- Measure- TypeIndex</i>	The given spatial measure type index
--	--------------------------------------

Definition at line 174 of file SpatialMeasureAttributeAutoGenerated.cpp.

References multiscale::ERR\_UNDEFINED\_ENUM\_VALUE.

**5.4.1.7 void multiscale::verification::spatialmeasure::validateSpatialMeasureType ( const SpatialMeasureType & *spatialMeasureType* )**

Check if the given spatial measure type is valid.

**Parameters**

<i>spatialMeasureType</i>	The given spatial measure type
---------------------------	--------------------------------

Definition at line 12 of file SpatialMeasureAttribute.cpp.

References multiscale::ERR\_UNDEFINED\_ENUM\_VALUE.

Referenced by computeSpatialMeasureTypeIndex(), and multiscale::verification::- SpatialMeasureEvaluator::evaluate().

**5.4.1.8 void multiscale::verification::spatialmeasure::validateSpatialMeasureTypeIndex ( const std::size\_t & *spatialMeasureTypeIndex* )**

Check if the given spatial measure type index is valid.

**Parameters**

<i>spatialMeasureTypeIndex</i>	The given spatial measure type index
--------------------------------	--------------------------------------

Definition at line 19 of file SpatialMeasureAttribute.cpp.

References multiscale::ERR\_UNDEFINED\_ENUM\_VALUE, and multiscale::verification::NR\_SPATIAL\_MEASURE\_TYPES.

Referenced by computeSpatialMeasureType().

## 5.5 multiscale::verification::subsetsspecific Namespace Reference

### Functions

- void [validateSubsetSpecificType](#) (const SubsetSpecificType &subsetSpecificType)

*Check if the given subset specific type is valid.*

- void `validateSubsetSpecificTypeIndex` (const std::size\_t &subsetSpecificTypeIndex)
 

*Check if the given subset specific type index is valid.*
- size\_t `computeSubsetSpecificTypeIndex` (const `SubsetSpecificType` &subsetSpecificType)
 

*Compute the index of the subset specific type.*
- `SubsetSpecificType computeSubsetSpecificType` (const std::size\_t &subsetSpecificTypeIndex)
 

*Compute the subset specific type from the given index.*
- std::string `toString` (const `SubsetSpecificType` &subsetSpecificType)
 

*Get the string representation of the given subset specific type.*
- std::string `toString` (const std::size\_t &subsetSpecificTypeIndex)
 

*Get the string representation corresponding to the the given subset specific type index.*

### 5.5.1 Function Documentation

#### 5.5.1.1 `SubsetSpecificType multiscale::verification::subsetspecific::computeSubsetSpecificType` ( const std::size\_t & *subsetSpecificTypeIndex* )

Compute the subset specific type from the given index.

##### Parameters

<i>subset-Specific-TypeIndex</i>	The given subset specific type index
----------------------------------	--------------------------------------

Definition at line 31 of file `SubsetSpecificAttribute.cpp`.

References `validateSubsetSpecificTypeIndex()`.

Referenced by `multiscale::verification::MSTMLSubfilesMerger::addSpatialEntitiesToResultingTraceTimepoint()`, `multiscale::verification::ConstraintVisitor::evaluateScaleAndSubsystemConstraint()`, `multiscale::verification::ConstraintVisitor::evaluateSpatialMeasureConstraint()`, and `multiscale::verification::TimePoint::updateSpatialEntities()`.

#### 5.5.1.2 `size_t multiscale::verification::subsetspecific::computeSubsetSpecificTypeIndex` ( const `SubsetSpecificType` & *subsetSpecificType* )

Compute the index of the subset specific type.

##### Parameters

<i>subset-SpecificType</i>	The given subset specific type
----------------------------	--------------------------------

Definition at line 24 of file SubsetSpecificAttribute.cpp.

References validateSubsetSpecificType().

Referenced by multiscale::verification::TimePoint::addConsideredSpatialEntityType(), multiscale::verification::TimePoint::addSpatialEntity(), multiscale::verification::TimePoint::containsSpatialEntity(), multiscale::verification::TimePoint::getSpatialEntitiesBeginIterator(), multiscale::verification::TimePoint::getSpatialEntitiesEndIterator(), multiscale::verification::TimePoint::removeSpatialEntity(), and multiscale::verification::TimePoint::setConsideredSpatialEntityType().

#### 5.5.1.3 `std::string multiscale::verification::subsetsspecific::toString ( const SubsetSpecificType & subsetSpecificType )`

Get the string representation of the given subset specific type.

##### Parameters

<code>subset- SpecificType</code>	The given subset specific type
---------------------------------------	--------------------------------

Definition at line 17 of file SubsetSpecificAttributeAutoGenerated.cpp.

References multiscale::ERR\_UNDEFINED\_ENUM\_VALUE.

#### 5.5.1.4 `std::string multiscale::verification::subsetsspecific::toString ( const std::size_t & subsetSpecificTypeIndex )`

Get the string representation corresponding to the the given subset specific type index.

##### Parameters

<code>subset- Specific- TypeIndex</code>	The given subset specific type index
--	--------------------------------------

Definition at line 34 of file SubsetSpecificAttributeAutoGenerated.cpp.

References multiscale::ERR\_UNDEFINED\_ENUM\_VALUE.

#### 5.5.1.5 `void multiscale::verification::subsetsspecific::validateSubsetSpecificType ( const SubsetSpecificType & subsetSpecificType )`

Check if the given subset specific type is valid.

##### Parameters

<code>subset- SpecificType</code>	The given subset specific type
---------------------------------------	--------------------------------

Definition at line 10 of file SubsetSpecificAttribute.cpp.

References multiscale::ERR\_UNDEFINED\_ENUM\_VALUE.

Referenced by multiscale::verification::TimePoint::addConsideredSpatialEntityType(), multiscale::verification::TimePoint::addSpatialEntity(), computeSubsetSpecificTypeIndex(), multiscale::verification::TimePoint::getSpatialEntitiesBeginIterator(), multiscale::verification::TimePoint::getSpatialEntitiesEndIterator(), multiscale::verification::TimePoint::removeSpatialEntity(), and multiscale::verification::TimePoint::setConsideredSpatialEntityType().

```
5.5.1.6 void multiscale::verification::subsetsspecific::validateSubset-
    SpecificTypeIndex ( const std::size_t & subsetSpecificTypeIndex
    )
```

Check if the given subset specific type index is valid.

#### Parameters

<i>subset-Specific-TypeIndex</i>	The given subset specific type index
----------------------------------	--------------------------------------

Definition at line 17 of file SubsetSpecificAttribute.cpp.

References multiscale::ERR\_UNDEFINED\_ENUM\_VALUE, and multiscale::verification::NR\_SUBSET\_SPECIFIC\_TYPES.

Referenced by computeSubsetSpecificType(), multiscale::verification::TimePoint::getSpatialEntitiesBeginIterator(), and multiscale::verification::TimePoint::getSpatialEntitiesEndIterator().

## 5.6 multiscale::visualisation Namespace Reference

### Classes

- class [AnnularSector](#)  
*An annular sector is the basic element in the considered circular geometry.*
- class [CartesianToPolarConverter](#)  
*Converter from the rectangular geometry grid cells to annular sectors.*
- class [PolarCsvToInputFilesConverter](#)  
*Csv file to input file converter considering polar coordinates.*
- class [PolarGnuplotScriptGenerator](#)  
*Gnuplot script generator from the provided annular sectors.*
- class [CartesianToConcentrationsConverter](#)  
*Scale the values of the rectangular geometry grid cells.*
- class [RectangularCsvToInputFilesConverter](#)  
*Csv file to input file converter considering cartesian coordinates.*

- class [RectangularEntityCsvToInputFilesConverter](#)  
*Csv entity file to input file converter considering cartesian coordinates.*
- class [RectangularGnuplotScriptGenerator](#)  
*Gnuplot script generator from the provided concentrations considering a rectangular geometry.*

## 5.7 multiscaletest Namespace Reference

### Namespaces

- namespace [verification](#)

### Classes

- class [MultiscaleTest](#)
- class [MinEnclosingTriangleFinderTest](#)  
*Class for testing the minimum enclosing triangle algorithm.*
- class [ApproximateBayesianModelCheckerTest](#)  
*Class for testing the approximate Bayesian model checker.*
- class [ApproximateProbabilisticModelCheckerTest](#)  
*Class for testing the approximate probabilistic model checker.*
- class [BayesianModelCheckerTest](#)  
*Class for testing the Bayesian model checker.*
- class [ModelCheckerTest](#)  
*Class for testing model checkers.*
- class [ProbabilisticBlackBoxModelCheckerTest](#)  
*Class for testing the probabilistic black-box model checker.*
- class [StatisticalModelCheckerTest](#)  
*Class for testing the statistical model checker.*
- class [CompleteTraceTest](#)  
*Class for testing evaluation of complete traces containing both numeric state variables and spatial entities.*
- class [EmptyTraceTest](#)  
*Class for testing evaluation of empty traces.*
- class [NumericStateVariableTraceTest](#)  
*Class for testing evaluation of numeric state variable-only traces.*
- class [SpatialEntitiesTraceTest](#)  
*Class for testing evaluation of spatial entities-only traces.*
- class [TraceEvaluationTest](#)  
*Class for testing evaluation of traces.*

## 5.8 multiscaletest::verification Namespace Reference

### Functions

- bool [parseInputString](#) (const std::string &inputString)  
*Parse the input string and return the result of the parsing.*

#### 5.8.1 Function Documentation

##### 5.8.1.1 bool multiscaletest::verification::parseInputString ( const std::string &inputString )

Parse the input string and return the result of the parsing.

#### Parameters

<i>inputString</i>	The input string
--------------------	------------------

Definition at line 27 of file InputStringParser.hpp.

References multiscale::verification::Parser::parse(), and parseInputString().

Referenced by parseInputString().

## Chapter 6

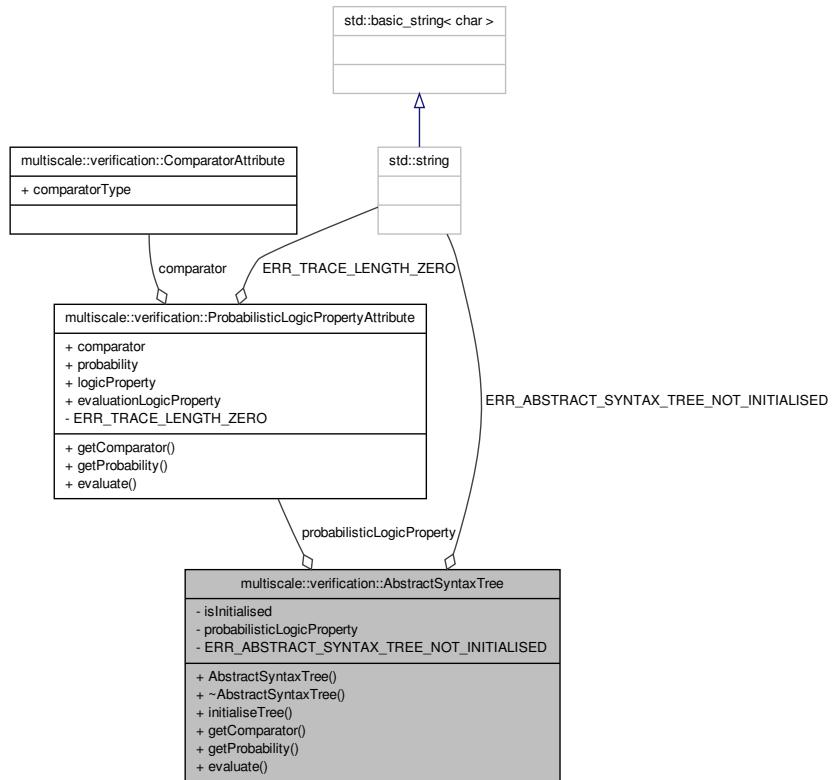
# Class Documentation

### 6.1 multiscale::verification::AbstractSyntaxTree Class Reference

Class used for representing an abstract syntax tree.

```
#include <AbstractSyntaxTree.hpp>
```

Collaboration diagram for multiscale::verification::AbstractSyntaxTree:



## Public Member Functions

- `AbstractSyntaxTree ()`
- `~AbstractSyntaxTree ()`
- void `initialiseTree (const ProbabilisticLogicPropertyAttribute &probabilisticLogicPropertyAttribute)`

*Initialise the abstract syntax tree using the given probabilistic logic property attribute.*
- `ComparatorType getComparator ()`

*Get the type of the comparator used in the probabilistic logical query.*
- double `getProbability ()`

*Get the value of the probability used in the probabilistic logical query.*
- bool `evaluate (const SpatialTemporalTrace &spatialTemporalTrace, const - MultiscaleArchitectureGraph &multiscaleArchitectureGraph)`

*Evaluate the abstract syntax tree considering the given trace and multiscale architecture graph.*

### Private Attributes

- bool `isInitialised`
- ProbabilisticLogicPropertyAttribute `probabilisticLogicProperty`

### Static Private Attributes

- static const std::string `ERR_ABSTRACT_SYNTAX_TREE_NOT_INITIALISED` = "The abstract syntax tree was not initialised before evaluation. Call the method `initialiseTree(...)` before calling the method `evaluate(...)`."

#### 6.1.1 Detailed Description

Class used for representing an abstract syntax tree.

Definition at line 14 of file AbstractSyntaxTree.hpp.

#### 6.1.2 Constructor & Destructor Documentation

##### 6.1.2.1 AbstractSyntaxTree::AbstractSyntaxTree( )

Definition at line 7 of file AbstractSyntaxTree.cpp.

References `isInitialised`.

##### 6.1.2.2 AbstractSyntaxTree::~AbstractSyntaxTree( )

Definition at line 11 of file AbstractSyntaxTree.cpp.

#### 6.1.3 Member Function Documentation

##### 6.1.3.1 bool AbstractSyntaxTree::evaluate ( const SpatialTemporalTrace & *spatialTemporalTrace*, const MultiscaleArchitectureGraph & *multiscaleArchitectureGraph* )

Evaluate the abstract syntax tree considering the given trace and multiscale architecture graph.

#### Parameters

<i>spatial-Temporal-Trace</i>	The given spatial temporal trace
<i>multiscale-Architecture-Graph</i>	The considered multiscale architecture graph

Definition at line 27 of file AbstractSyntaxTree.cpp.

References `ERR_ABSTRACT_SYNTAX_TREE_NOT_INITIALISED`, `multiscale::verification::ProbabilisticLogicPropertyAttribute::evaluate()`, `isInitialised`, and `probabilisticLogicProperty`.

Referenced by `multiscale::verification::ModelChecker::evaluate()`, and `multiscaletest::TraceEvaluationTest::RunTest()`.

#### 6.1.3.2 ComparatorType `AbstractSyntaxTree::getComparator( )`

Get the type of the comparator used in the probabilistic logical query.

Definition at line 19 of file AbstractSyntaxTree.cpp.

References `multiscale::verification::ProbabilisticLogicPropertyAttribute::getComparator()`, and `probabilisticLogicProperty`.

Referenced by `multiscale::verification::ModelChecker::isGreaterThanOrEqualToComparator()`.

#### 6.1.3.3 double `AbstractSyntaxTree::getProbability( )`

Get the value of the probability used in the probabilistic logical query.

Definition at line 23 of file AbstractSyntaxTree.cpp.

References `multiscale::verification::ProbabilisticLogicPropertyAttribute::getProbability()`, and `probabilisticLogicProperty`.

Referenced by `multiscale::verification::ApproximateProbabilisticModelChecker::initialise()`, `multiscale::verification::StatisticalModelChecker::initialise()`, `multiscale::verification::BayesianModelChecker::initialise()`, `multiscale::verification::ApproximateBayesianModelChecker::initialise()`, `multiscale::verification::ModelChecker::updateHypothesesPValuesForGreaterThan()`, and `multiscale::verification::ModelChecker::updateHypothesesPValuesForLessThan()`.

#### 6.1.3.4 void `AbstractSyntaxTree::initialiseTree( const ProbabilisticLogicPropertyAttribute & probabilisticLogicPropertyAttribute )`

Initialise the abstract syntax tree using the given probabilistic logic property attribute.

##### Parameters

<code>probabilisticLogicPropertyAttribute</code>	The probabilistic logic property attribute
--	--

Definition at line 13 of file AbstractSyntaxTree.cpp.

References isInitialised, and probabilisticLogicProperty.

Referenced by multiscale::verification::Parser::parseLogicalQuery().

#### 6.1.4 Member Data Documentation

6.1.4.1 `const std::string AbstractSyntaxTree::ERR_ABSTRACT_SYNTAX_TREE_NOT_INITIALISED = "The abstract syntax tree was not initialised before evaluation. Call the method initialiseTree(...) before calling the method evaluate(...)." [static, private]`

Definition at line 53 of file AbstractSyntaxTree.hpp.

Referenced by evaluate().

6.1.4.2 `bool multiscale::verification::AbstractSyntaxTree::isInitialised [private]`

Flag for indicating if the abstract syntax tree was initialised

Definition at line 19 of file AbstractSyntaxTree.hpp.

Referenced by AbstractSyntaxTree(), evaluate(), and initialiseTree().

6.1.4.3 `ProbabilisticLogicPropertyAttribute multiscale::verification::AbstractSyntaxTree::probabilisticLogicProperty [private]`

The abstract syntax tree represented using a probabilistic logic property attribute

Definition at line 22 of file AbstractSyntaxTree.hpp.

Referenced by evaluate(), getComparator(), getProbability(), and initialiseTree().

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

- AbstractSyntaxTree.hpp
- AbstractSyntaxTree.cpp

## 6.2 multiscale::AdditionOperation Class Reference

Functor representing an addition operation.

```
#include <Numeric.hpp>
```

### Public Member Functions

- template<typename Operand >  
Operand [operator\(\)](#) (Operand operand1, Operand operand2) const

*Add the two operands.*

### 6.2.1 Detailed Description

Functor representing an addition operation.

Definition at line 536 of file Numeric.hpp.

### 6.2.2 Member Function Documentation

6.2.2.1 `template<typename Operand > Operand multiscale::AdditionOperation::operator() ( Operand operand1, Operand operand2 ) const [inline]`

Add the two operands.

#### Parameters

<i>operand1</i>	The first operand
<i>operand2</i>	The second operand

Definition at line 546 of file Numeric.hpp.

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

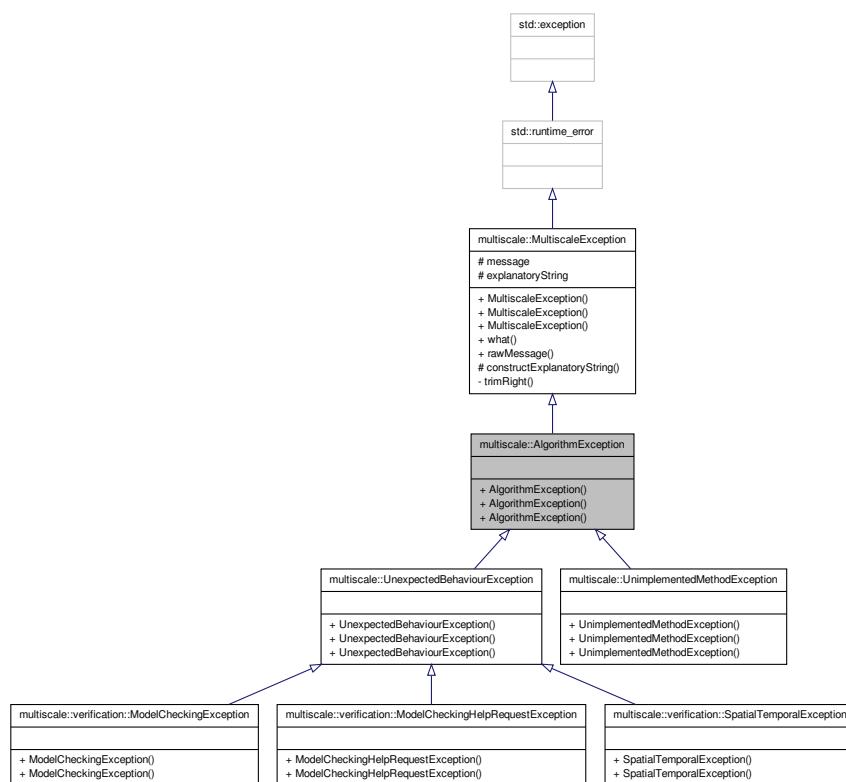
- Numeric.hpp

## 6.3 multiscale::AlgorithmException Class Reference

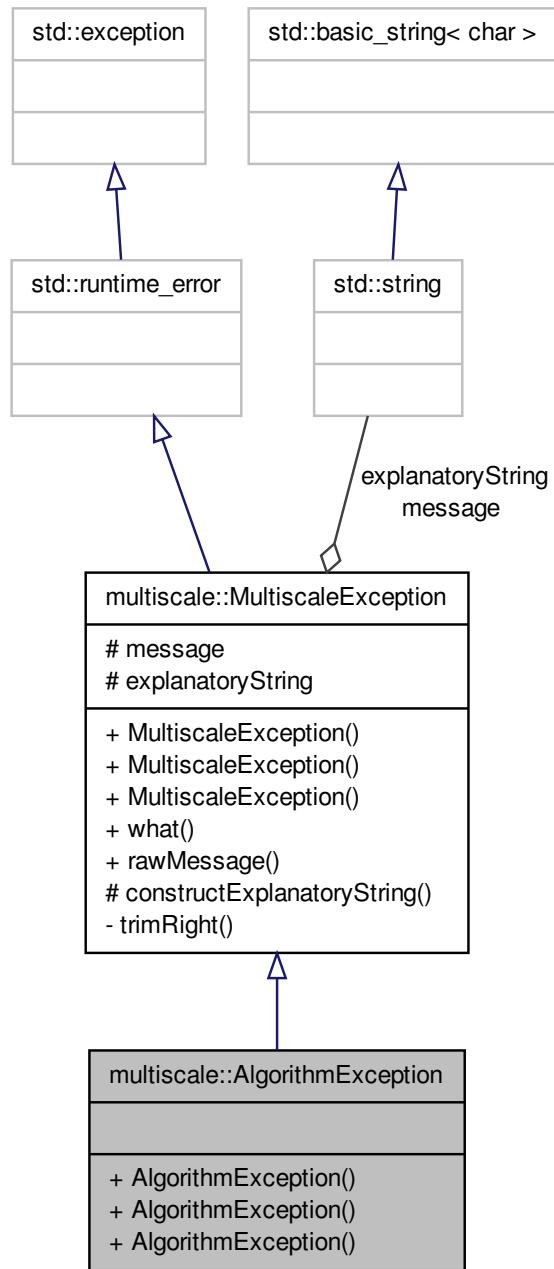
Class for representing algorithm exceptions.

```
#include <AlgorithmException.hpp>
```

Inheritance diagram for multiscale::AlgorithmException:



Collaboration diagram for multiscale::AlgorithmException:



## Public Member Functions

- [AlgorithmException \(\)](#)
- [AlgorithmException \(const std::string &file, int line, const std::string &msg\)](#)
- [AlgorithmException \(const std::string &file, int line, const char \\*msg\)](#)

### 6.3.1 Detailed Description

Class for representing algorithm exceptions.

Definition at line 12 of file AlgorithmException.hpp.

### 6.3.2 Constructor & Destructor Documentation

#### 6.3.2.1 multiscale::AlgorithmException::AlgorithmException( ) [inline]

Definition at line 16 of file AlgorithmException.hpp.

#### 6.3.2.2 multiscale::AlgorithmException::AlgorithmException( const std::string &file, int line, const std::string &msg ) [inline, explicit]

Definition at line 18 of file AlgorithmException.hpp.

#### 6.3.2.3 multiscale::AlgorithmException::AlgorithmException( const std::string &file, int line, const char \* msg ) [inline, explicit]

Definition at line 23 of file AlgorithmException.hpp.

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

- [AlgorithmException.hpp](#)

## 6.4 multiscale::verification::AndConstraintAttribute Class Reference

Class for representing an "and" constraint attribute.

```
#include <AndConstraintAttribute.hpp>
```

## Public Attributes

- [ConstraintAttributeType constraint](#)

### 6.4.1 Detailed Description

Class for representing an "and" constraint attribute.

Definition at line 14 of file AndConstraintAttribute.hpp.

### 6.4.2 Member Data Documentation

#### 6.4.2.1 ConstraintAttributeType multiscale::verification::AndConstraintAttribute::constraint

The constraint following the "and" operator

Definition at line 18 of file AndConstraintAttribute.hpp.

Referenced by multiscale::verification::ConstraintVisitor::operator()().

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

- AndConstraintAttribute.hpp

## 6.5 multiscale::verification::AndLogicPropertyAttribute Class - Reference

Class for representing an "and" logic property attribute.

```
#include <AndLogicPropertyAttribute.hpp>
```

### Public Attributes

- [LogicPropertyAttributeType logicProperty](#)

### 6.5.1 Detailed Description

Class for representing an "and" logic property attribute.

Definition at line 14 of file AndLogicPropertyAttribute.hpp.

### 6.5.2 Member Data Documentation

#### 6.5.2.1 LogicPropertyAttributeType multiscale::verification::AndLogicPropertyAttribute::logicProperty

The logical property following the "and" operator

Definition at line 18 of file AndLogicPropertyAttribute.hpp.

Referenced by multiscale::verification::LogicPropertyVisitor::operator()().

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

- AndLogicPropertyAttribute.hpp

## 6.6 multiscale::visualisation::AnnularSector Class Reference

An annular sector is the basic element in the considered circular geometry.

```
#include <AnnularSector.hpp>
```

### Public Member Functions

- `AnnularSector ()`
- `~AnnularSector ()`
- `void initialise (double startingRadius, double endingRadius, double startingAngle, double endingAngle, double concentration)`  
*Initialise the members of the class.*
- `double getConcentration () const`  
*Get the value of the concentration.*
- `double getEndingAngle () const`  
*Get the value of the ending angle.*
- `double getEndingRadius () const`  
*Get the value of the ending radius.*
- `double getStartingAngle () const`  
*Get the value of the starting angle.*
- `double getStartingRadius () const`  
*Get the value of the starting radius.*
- `std::string toString ()`  
*Get the string representation of the annular sector.*

### Private Attributes

- `double startingRadius`
- `double endingRadius`
- `double startingAngle`
- `double endingAngle`
- `double concentration`

#### 6.6.1 Detailed Description

An annular sector is the basic element in the considered circular geometry.

More information about annuli and sectors of annuli can be found online (e.g. - Wikipedia).

Definition at line 15 of file AnnularSector.hpp.

## 6.6.2 Constructor & Destructor Documentation

### 6.6.2.1 AnnularSector::AnnularSector( )

Definition at line 11 of file AnnularSector.cpp.

References concentration, endingAngle, endingRadius, startingAngle, and startingRadius.

### 6.6.2.2 AnnularSector::~AnnularSector( )

Definition at line 19 of file AnnularSector.cpp.

## 6.6.3 Member Function Documentation

### 6.6.3.1 double AnnularSector::getConcentration( ) const

Get the value of the concentration.

Definition at line 30 of file AnnularSector.cpp.

References concentration.

### 6.6.3.2 double AnnularSector::getEndingAngle( ) const

Get the value of the ending angle.

Definition at line 34 of file AnnularSector.cpp.

References endingAngle.

### 6.6.3.3 double AnnularSector::getEndingRadius( ) const

Get the value of the ending radius.

Definition at line 38 of file AnnularSector.cpp.

References endingRadius.

### 6.6.3.4 double AnnularSector::getStartingAngle( ) const

Get the value of the starting angle.

Definition at line 42 of file AnnularSector.cpp.

References startingAngle.

**6.6.3.5 double AnnularSector::getStartingRadius( ) const**

Get the value of the starting radius.

Definition at line 46 of file AnnularSector.cpp.

References startingRadius.

**6.6.3.6 void AnnularSector::initialise( double *startingRadius*, double *endingRadius*, double *startingAngle*, double *endingAngle*, double *concentration* )**

Initialise the members of the class.

**Parameters**

<i>startingRadius</i>	Starting radius
<i>endingRadius</i>	Ending radius
<i>startingAngle</i>	Starting angle
<i>endingAngle</i>	Ending angle
<i>concentration</i>	Concentration

Definition at line 21 of file AnnularSector.cpp.

References concentration, endingAngle, endingRadius, startingAngle, and startingRadius.

**6.6.3.7 std::string AnnularSector::toString( )**

Get the string representation of the annular sector.

Definition at line 50 of file AnnularSector.cpp.

References concentration, endingAngle, endingRadius, startingAngle, and startingRadius.

**6.6.4 Member Data Documentation****6.6.4.1 double multiscale::visualisation::AnnularSector::concentration  
[private]**

Definition at line 23 of file AnnularSector.hpp.

Referenced by AnnularSector(), getConcentration(), initialise(), and toString().

**6.6.4.2 double multiscale::visualisation::AnnularSector::endingAngle**  
[private]

Definition at line 22 of file AnnularSector.hpp.

Referenced by AnnularSector(), getEndingAngle(), initialise(), and toString().

**6.6.4.3 double multiscale::visualisation::AnnularSector::endingRadius**  
[private]

Definition at line 20 of file AnnularSector.hpp.

Referenced by AnnularSector(), getEndingRadius(), initialise(), and toString().

**6.6.4.4 double multiscale::visualisation::AnnularSector::startingAngle**  
[private]

Definition at line 21 of file AnnularSector.hpp.

Referenced by AnnularSector(), getStartingAngle(), initialise(), and toString().

**6.6.4.5 double multiscale::visualisation::AnnularSector::startingRadius**  
[private]

Definition at line 19 of file AnnularSector.hpp.

Referenced by AnnularSector(), getStartingRadius(), initialise(), and toString().

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

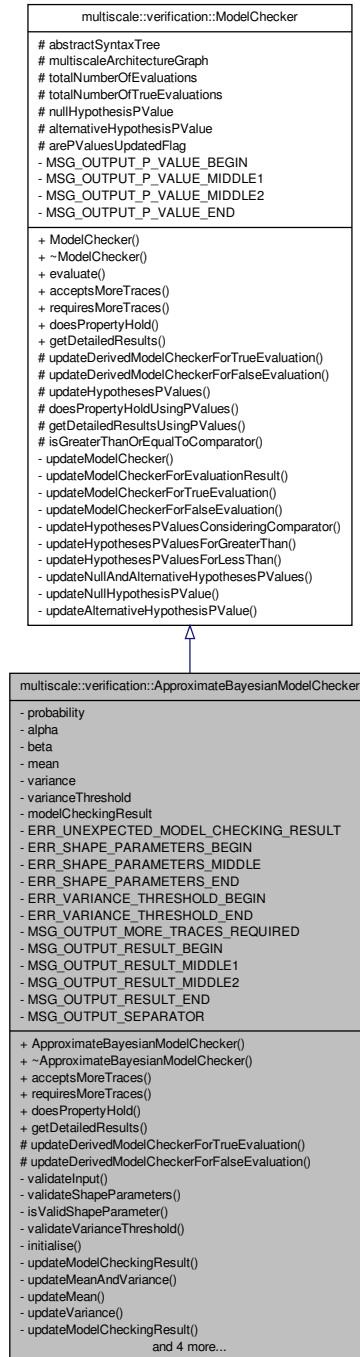
- AnnularSector.hpp
- AnnularSector.cpp

## 6.7 multiscale::verification::ApproximateBayesianModelChecker Class Reference

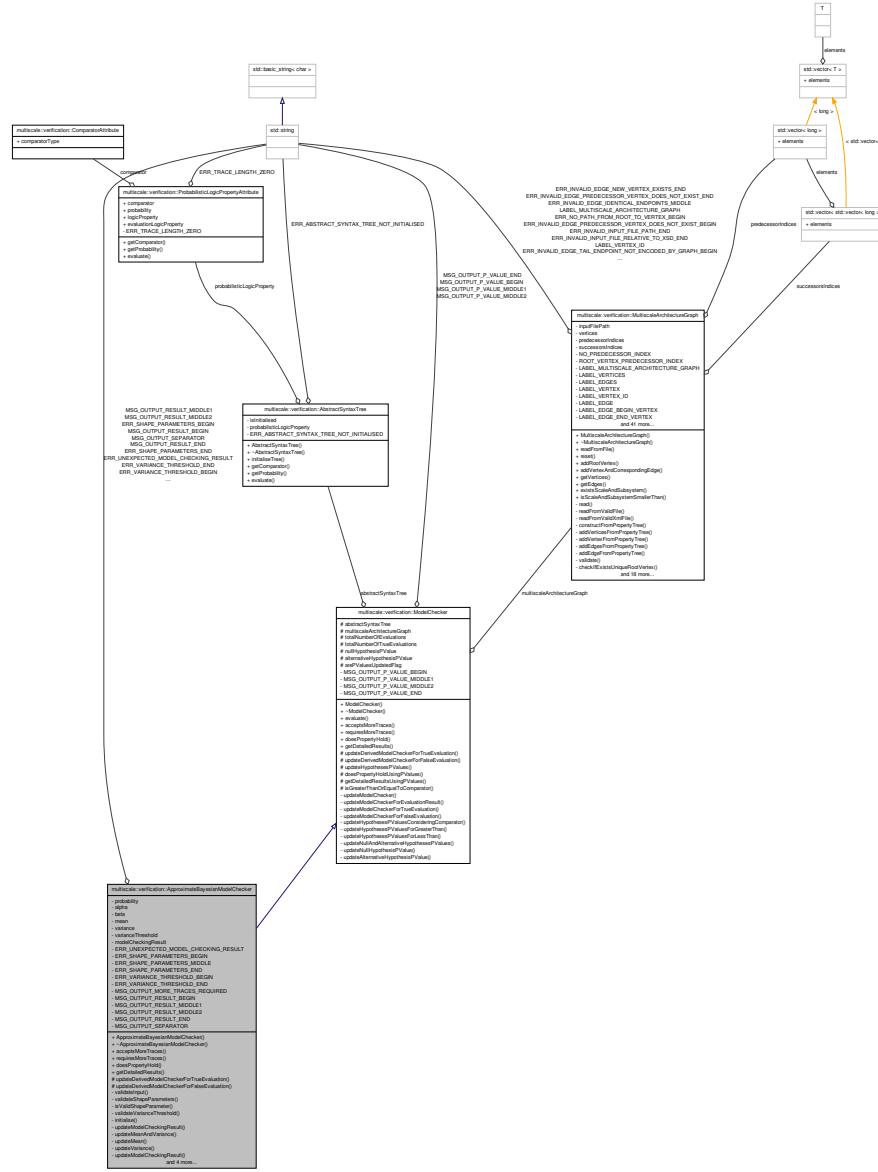
Class used to run approximate Bayesian model checking tasks.

```
#include <ApproximateBayesianModelChecker.hpp>
```

Inheritance diagram for multiscale::verification::ApproximateBayesianModelChecker:



## Collaboration diagram for multiscale::verification::ApproximateBayesianModelChecker:



## Public Member Functions

- `ApproximateBayesianModelChecker` (const `AbstractSyntaxTree` &`abstractSyntaxTree`, const `MultiscaleArchitectureGraph` &`multiscaleArchitectureGraph`, double `alpha`, double `beta`, double `varianceThreshold`)
  - `~ApproximateBayesianModelChecker()`
  - bool `acceptsMoreTraces()` override

*Check if more traces are accepted for evaluating the logic property.*

- bool `requiresMoreTraces ()` override

*Check if more traces are required for evaluating the logic property.*

- bool `doesPropertyHold ()` override

*Check if the given property holds.*

- std::string `getDetailedResults ()` override

*Get the detailed description of the results.*

## Protected Member Functions

- void `updateDerivedModelCheckerForTrueEvaluation ()` override

*Update the results of the derived model checker type considering that the logic property was evaluated to true for the last trace.*

- void `updateDerivedModelCheckerForFalseEvaluation ()` override

*Update the results of the derived model checker type considering that the logic property was evaluated to false for the last trace.*

## Private Member Functions

- void `validateInput (double alpha, double beta, double varianceThreshold)`

*Validate the input parameters  $\alpha$ ,  $\beta$  and the variance threshold.*

- void `validateShapeParameters (double alpha, double beta)`

*Validate the shape parameters  $\alpha$  and  $\beta$ .*

- bool `isValidShapeParameter (double shapeParameter)`

*Check if the given shape parameter value is valid.*

- void `validateVarianceThreshold (double varianceThreshold)`

*Validate the variance threshold.*

- void `initialise ()`

*Initialisation of some of the class members.*

- void `updateModelCheckingResult ()`

*Update the result of the model checking task.*

- void `updateMeanAndVariance ()`

*Update the value of the mean and variance estimates.*

- void `updateMean ()`

*Update the value of the mean estimate.*

- void `updateVariance ()`

*Update the value of the variance estimate.*

- void `updateModelCheckingResult (double variance)`

*Update the result of the model checking task considering the given variance value.*

- void `updateModelCheckingResultEnoughTraces (double variance)`

*Update the result of the model checking task considering that enough traces have been provided.*

- bool `isModelCheckingResultTrueConsideringComparator (double variance)`

*Check if the result of the model checking task is true considering the probabilistic comparator (i.e. <=, >=)*

- void `updateModelCheckingResultNotEnoughTraces ()`

*Update the result of the model checking task considering that not enough traces have been provided.*
- bool `doesPropertyHoldConsideringResult ()`

*Check if the given property holds considering the obtained model checking result.*
- std::string `getDetailedUpdatedResults ()`

*Get the detailed description of the updated results.*

### Private Attributes

- double `probability`
- double `alpha`
- double `beta`
- double `mean`
- double `variance`
- double `varianceThreshold`
- `ApproximateBayesianModelCheckingResult` `modelCheckingResult`

### Static Private Attributes

- static const std::string `ERR_UNEXPECTED_MODEL_CHECKING_RESULT` = "-  
An invalid ApproximateBayesian model checking result was obtained. Please  
check source code."
- static const std::string `ERR_SHAPE_PARAMETERS_BEGIN` = "The provided -  
Beta distribution shape parameters `alpha` and `beta` ("
- static const std::string `ERR_SHAPE_PARAMETERS_MIDDLE` = ", "
- static const std::string `ERR_SHAPE_PARAMETERS_END` = ") should be greater  
than zero. Please change."
- static const std::string `ERR_VARIANCE_THRESHOLD_BEGIN` = "The provided  
`variance` threshold ("
- static const std::string `ERR_VARIANCE_THRESHOLD_END` = ") should be  
greater than zero. Please change."
- static const std::string `MSG_OUTPUT_MORE_TRACES_REQUIRED` = "More  
traces are required to provide a true/false answer assuming the given Beta dis-  
tribution shape parameters and `variance` threshold value. Probabilistic black-box  
model checking was used instead to provide an answer."
- static const std::string `MSG_OUTPUT_RESULT_BEGIN` = "The provided answer  
is given for the Beta distribution shape parameters `alpha` = "
- static const std::string `MSG_OUTPUT_RESULT_MIDDLE1` = " and `beta` = "
- static const std::string `MSG_OUTPUT_RESULT_MIDDLE2` = ", and `variance`  
threshold value = "
- static const std::string `MSG_OUTPUT_RESULT_END` = ""
- static const std::string `MSG_OUTPUT_SEPARATOR` = " "

### 6.7.1 Detailed Description

Class used to run approximate Bayesian model checking tasks.

The implementation of this class is (partially) based on the algorithms described in the following paper:

C. Langmead, 'Generalized Queries and Bayesian Statistical Model Checking in - Dynamic Bayesian Networks: Application to Personalized Medicine', Computer Science Department, Aug. 2009.

In our implementation the variables in the original paper (right hand side of the assignments) have been given the following new names (left hand side of assignments):

probability =  $p$

alpha =  $\alpha$

beta =  $\beta$

mean =  $\hat{\rho}$

variance =  $\hat{v}$

varianceThreshold =  $T$

totalNumberOfEvaluations =  $n$

totalNumberOfTrueEvaluations =  $k$

Definition at line 51 of file ApproximateBayesianModelChecker.hpp.

### 6.7.2 Constructor & Destructor Documentation

6.7.2.1 **ApproximateBayesianModelChecker::ApproximateBayesianModelChecker ( const AbstractSyntaxTree & *abstractSyntaxTree*, const MultiscaleArchitectureGraph & *multiscaleArchitectureGraph*, double *alpha*, double *beta*, double *varianceThreshold* )**

Definition at line 11 of file ApproximateBayesianModelChecker.cpp.

References alpha, beta, initialise(), validateInput(), and varianceThreshold.

6.7.2.2 **ApproximateBayesianModelChecker::~ApproximateBayesianModelChecker ( )**

Definition at line 30 of file ApproximateBayesianModelChecker.cpp.

### 6.7.3 Member Function Documentation

6.7.3.1 **bool ApproximateBayesianModelChecker::acceptsMoreTraces ( ) [override, virtual]**

Check if more traces are accepted for evaluating the logic property.

Implements [multiscale::verification::ModelChecker](#).

Definition at line 32 of file ApproximateBayesianModelChecker.cpp.

References modelCheckingResult, and updateModelCheckingResult().

Referenced by requiresMoreTraces().

#### **6.7.3.2 bool ApproximateBayesianModelChecker::doesPropertyHold( ) [override, virtual]**

Check if the given property holds.

Implements [multiscale::verification::ModelChecker](#).

Definition at line 42 of file ApproximateBayesianModelChecker.cpp.

References doesPropertyHoldConsideringResult(), and updateModelCheckingResult().

#### **6.7.3.3 bool ApproximateBayesianModelChecker::doesPropertyHoldConsideringResult( ) [private]**

Check if the given property holds considering the obtained model checking result.

Definition at line 150 of file ApproximateBayesianModelChecker.cpp.

References multiscale::verification::ModelChecker::doesPropertyHoldUsingPValues(), -ERR\_UNEXPECTED\_MODEL\_CHECKING\_RESULT, and modelCheckingResult.

Referenced by doesPropertyHold().

#### **6.7.3.4 std::string ApproximateBayesianModelChecker::getDetailedResults( ) [override, virtual]**

Get the detailed description of the results.

Implements [multiscale::verification::ModelChecker](#).

Definition at line 48 of file ApproximateBayesianModelChecker.cpp.

References getDetailedUpdatedResults(), and updateModelCheckingResult().

#### **6.7.3.5 std::string ApproximateBayesianModelChecker::getDetailedUpdatedResults( ) [private]**

Get the detailed description of the updated results.

Definition at line 169 of file ApproximateBayesianModelChecker.cpp.

References alpha, beta, multiscale::verification::ModelChecker::getDetailedResultsUsingPValues(), modelCheckingResult, MSG\_OUTPUT\_MORE\_TRACES\_REQUIRED, MSG\_OUTPUT\_RESULT\_BEGIN, MSG\_OUTPUT\_RESULT\_END, MSG\_OUTPUT\_RESULT\_MIDDLE1, MSG\_OUTPUT\_RESULT\_MIDDLE2, MSG\_OUTPUT\_SEPARATOR, multiscale::StringManipulator::toString(), and varianceThreshold.

Referenced by getDetailedResults().

#### 6.7.3.6 void ApproximateBayesianModelChecker::initialise( ) [private]

Initialisation of some of the class members.

Definition at line 89 of file ApproximateBayesianModelChecker.cpp.

References multiscale::verification::ModelChecker::abstractSyntaxTree, multiscale::verification::AbstractSyntaxTree::getProbability(), mean, probability, and variance.

Referenced by ApproximateBayesianModelChecker().

#### 6.7.3.7 bool ApproximateBayesianModelChecker::isModelCheckingResultTrueConsideringComparator( double variance ) [private]

Check if the result of the model checking task is true considering the probabilistic comparator (i.e.  $\leq$ ,  $\geq$ )

For queries of type : a)  $P \geq \theta[\phi]$  the result is (*mean*  $\geq \theta$ ) b)  $P \leq \theta[\phi]$  the result is (*mean*  $\leq \theta$ )

##### Parameters

<i>variance</i>	The given variance value
-----------------	--------------------------

Definition at line 138 of file ApproximateBayesianModelChecker.cpp.

References multiscale::Numeric::greaterOrEqual(), multiscale::verification::ModelChecker::isGreaterThanOrEqualToComparator(), multiscale::Numeric::lessOrEqual(), mean, and probability.

Referenced by updateModelCheckingResultEnoughTraces().

#### 6.7.3.8 bool ApproximateBayesianModelChecker::isValidShapeParameter( double shapeParameter ) [private]

Check if the given shape parameter value is valid.

The shape parameter values should be greater than zero

##### Parameters

<i>shape-Parameter</i>	The given shape parameter
------------------------	---------------------------

Definition at line 74 of file ApproximateBayesianModelChecker.cpp.

Referenced by validateShapeParameters().

**6.7.3.9 bool ApproximateBayesianModelChecker::requiresMoreTraces( )**  
[override, virtual]

Check if more traces are required for evaluating the logic property.

Implements [multiscale::verification::ModelChecker](#).

Definition at line 38 of file ApproximateBayesianModelChecker.cpp.

References [acceptsMoreTraces\(\)](#).

**6.7.3.10 void ApproximateBayesianModelChecker::updateDerivedModelCheckerForFalseEvaluation( )** [override, protected, virtual]

Update the results of the derived model checker type considering that the logic property was evaluated to false for the last trace.

Do not do anything

Implements [multiscale::verification::ModelChecker](#).

Definition at line 56 of file ApproximateBayesianModelChecker.cpp.

**6.7.3.11 void ApproximateBayesianModelChecker::updateDerivedModelCheckerForTrueEvaluation( )** [override, protected, virtual]

Update the results of the derived model checker type considering that the logic property was evaluated to true for the last trace.

Do not do anything

Implements [multiscale::verification::ModelChecker](#).

Definition at line 54 of file ApproximateBayesianModelChecker.cpp.

**6.7.3.12 void ApproximateBayesianModelChecker::updateMean( )** [private]

Update the value of the mean estimate.

Definition at line 106 of file ApproximateBayesianModelChecker.cpp.

References alpha, beta, [multiscale::Numeric::division\(\)](#), mean, [multiscale::verification::ModelChecker::totalNumberOfEvaluations](#), and [multiscale::verification::ModelChecker::totalNumberOfTrueEvaluations](#).

Referenced by [updateMeanAndVariance\(\)](#).

**6.7.3.13 void ApproximateBayesianModelChecker::updateMeanAndVariance( )** [private]

Update the value of the mean and variance estimates.

Definition at line 101 of file ApproximateBayesianModelChecker.cpp.

References updateMean(), and updateVariance().

Referenced by updateModelCheckingResult().

**6.7.3.14 void ApproximateBayesianModelChecker::updateModelCheckingResult ( ) [private]**

Update the result of the model checking task.

Definition at line 96 of file ApproximateBayesianModelChecker.cpp.

References updateMeanAndVariance(), and variance.

Referenced by acceptsMoreTraces(), doesPropertyHold(), and getDetailedResults().

**6.7.3.15 void ApproximateBayesianModelChecker::updateModelCheckingResult ( double variance ) [private]**

Update the result of the model checking task considering the given variance value.

**Parameters**

<b>variance</b>	The given variance value
-----------------	--------------------------

Definition at line 122 of file ApproximateBayesianModelChecker.cpp.

References updateModelCheckingResultEnoughTraces(), updateModelCheckingResultNotEnoughTraces(), and varianceThreshold.

**6.7.3.16 void ApproximateBayesianModelChecker::update- ModelCheckingResultEnoughTraces ( double variance ) [private]**

Update the result of the model checking task considering that enough traces have been provided.

**Parameters**

<b>variance</b>	The given variance value
-----------------	--------------------------

Definition at line 130 of file ApproximateBayesianModelChecker.cpp.

References multiscale::verification::FALSE, isModelCheckingResultTrueConsideringComparator(), modelCheckingResult, and multiscale::verification::TRUE.

Referenced by updateModelCheckingResult().

---

**6.7.3.17 void ApproximateBayesianModelChecker::updateModelCheckingResult-NotEnoughTraces( ) [private]**

Update the result of the model checking task considering that not enough traces have been provided.

Definition at line 146 of file ApproximateBayesianModelChecker.cpp.

References modelCheckingResult, and multiscale::verification::MORE\_TRACES\_REQUIRED.

Referenced by updateModelCheckingResult().

**6.7.3.18 void ApproximateBayesianModelChecker::updateVariance( ) [private]**

Update the value of the variance estimate.

Definition at line 113 of file ApproximateBayesianModelChecker.cpp.

References alpha, beta, multiscale::Numeric::division(), multiscale::verification::ModelChecker::totalNumberOfEvaluations, multiscale::verification::ModelChecker::totalNumberOfTrueEvaluations, and variance.

Referenced by updateMeanAndVariance().

**6.7.3.19 void ApproximateBayesianModelChecker::validateInput( double alpha, double beta, double varianceThreshold ) [private]**

Validate the input parameters  $\alpha$ ,  $\beta$  and the variance threshold.

$\alpha$ ,  $\beta$  and variance threshold should be greater than zero

**Parameters**

<i>alpha</i>	The shape parameter $\alpha$ for the Beta distribution
<i>beta</i>	The shape parameter $\beta$ for the Beta distribution
<i>variance-Threshold</i>	The variance threshold

Definition at line 58 of file ApproximateBayesianModelChecker.cpp.

References validateShapeParameters(), and validateVarianceThreshold().

Referenced by ApproximateBayesianModelChecker().

**6.7.3.20 void ApproximateBayesianModelChecker::validateShapeParameters( double alpha, double beta ) [private]**

Validate the shape parameters  $\alpha$  and  $\beta$ .

$\alpha$  and  $\beta$  should be greater than zero

## Parameters

<i>alpha</i>	The shape parameter $\alpha$ for the Beta distribution
<i>beta</i>	The shape parameter $\beta$ for the Beta distribution

Definition at line 63 of file ApproximateBayesianModelChecker.cpp.

References ERR\_SHAPE\_PARAMETERS\_BEGIN, ERR\_SHAPE\_PARAMETERS\_END, ERR\_SHAPE\_PARAMETERS\_MIDDLE, isValidShapeParameter(), and multiscale::StringManipulator::toString().

Referenced by validateInput().

**6.7.3.21 void ApproximateBayesianModelChecker::validateVarianceThreshold ( double *varianceThreshold* ) [private]**

Validate the variance threshold.

The variance threshold should be greater than 0

## Parameters

<i>variance- Threshold</i>	The variance threshold
--------------------------------	------------------------

Definition at line 78 of file ApproximateBayesianModelChecker.cpp.

References ERR\_VARIANCE\_THRESHOLD\_BEGIN, ERR\_VARIANCE\_THRESHOLD\_END, multiscale::Numeric::lessOrEqual(), and multiscale::StringManipulator::toString().

Referenced by validateInput().

**6.7.4 Member Data Documentation**

**6.7.4.1 double multiscale::verification::ApproximateBayesianModelChecker-  
::alpha [private]**

The shape parameter  $\alpha$  for the Beta distribution prior

Definition at line 58 of file ApproximateBayesianModelChecker.hpp.

Referenced by ApproximateBayesianModelChecker(), getDetailedUpdatedResults(), updateMean(), and updateVariance().

**6.7.4.2 double multiscale::verification::ApproximateBayesianModelChecker::beta  
[private]**

The shape parameter  $\beta$  for the Beta distribution prior

Definition at line 59 of file ApproximateBayesianModelChecker.hpp.

Referenced by ApproximateBayesianModelChecker(), getDetailedUpdatedResults(), updateMean(), and updateVariance().

6.7.4.3 `const std::string ApproximateBayesianModelChecker::ERR_SHAPE_PARAMETERS_BEGIN = "The provided Beta distribution shape parameters alpha and beta (" [static, private]`

Definition at line 185 of file ApproximateBayesianModelChecker.hpp.

Referenced by validateShapeParameters().

6.7.4.4 `const std::string ApproximateBayesianModelChecker::ERR_SHAPE_PARAMETERS_END = ") should be greater than zero. Please change." [static, private]`

Definition at line 187 of file ApproximateBayesianModelChecker.hpp.

Referenced by validateShapeParameters().

6.7.4.5 `const std::string ApproximateBayesianModelChecker::ERR_SHAPE_PARAMETERS_MIDDLE = ", " [static, private]`

Definition at line 186 of file ApproximateBayesianModelChecker.hpp.

Referenced by validateShapeParameters().

6.7.4.6 `const std::string ApproximateBayesianModelChecker::ERR_UNEXPECTED_MODEL_CHECKING_RESULT = "An invalid ApproximateBayesian model checking result was obtained. Please check source code." [static, private]`

Definition at line 183 of file ApproximateBayesianModelChecker.hpp.

Referenced by doesPropertyHoldConsideringResult().

6.7.4.7 `const std::string ApproximateBayesianModelChecker::ERR_VARIANCE_THRESHOLD_BEGIN = "The provided variance threshold (" [static, private]`

Definition at line 189 of file ApproximateBayesianModelChecker.hpp.

Referenced by validateVarianceThreshold().

6.7.4.8 `const std::string ApproximateBayesianModelChecker::ERR_VARIANCE_THRESHOLD_END = ") should be greater than zero. Please change." [static, private]`

Definition at line 190 of file ApproximateBayesianModelChecker.hpp.

Referenced by validateVarianceThreshold().

**6.7.4.9 double multiscale::verification::ApproximateBayesianModelChecker-  
::mean [private]**

The value of the mean

Definition at line 61 of file ApproximateBayesianModelChecker.hpp.

Referenced by initialise(), isModelCheckingResultTrueConsideringComparator(), and updateMean().

**6.7.4.10 ApproximateBayesianModelCheckingResult multiscale::verification-  
::ApproximateBayesianModelChecker::modelCheckingResult  
[private]**

The result of the model checking task

Definition at line 67 of file ApproximateBayesianModelChecker.hpp.

Referenced by acceptsMoreTraces(), doesPropertyHoldConsideringResult(), getDetailedUpdatedResults(), updateModelCheckingResultEnoughTraces(), and updateModelCheckingResultNotEnoughTraces().

**6.7.4.11 const std::string ApproximateBayesianModelChecker::MSG\_OUTPUT\_MO-  
RE\_TRACES\_REQUIRED = "More traces are required to provide a true/false  
answer assuming the given Beta distribution shape parameters and variance  
threshold value. Probabilistic black-box model checking was used instead to provide an  
answer." [static, private]**

Definition at line 192 of file ApproximateBayesianModelChecker.hpp.

Referenced by getDetailedUpdatedResults().

**6.7.4.12 const std::string ApproximateBayesianModelChecker::MSG\_OUTPUT\_R-  
ESULT\_BEGIN = "The provided answer is given for the Beta distribution shape  
parameters alpha = " [static, private]**

Definition at line 194 of file ApproximateBayesianModelChecker.hpp.

Referenced by getDetailedUpdatedResults().

**6.7.4.13 const std::string ApproximateBayesianModelChecker-  
::MSG\_OUTPUT\_RESULT\_END = "" [static,  
private]**

Definition at line 197 of file ApproximateBayesianModelChecker.hpp.

Referenced by getDetailedUpdatedResults().

6.7.4.14 `const std::string ApproximateBayesianModelChecker::MSG_OUTPUT_RESULT_MIDDLE1 = " and beta = " [static, private]`

Definition at line 195 of file ApproximateBayesianModelChecker.hpp.

Referenced by `getDetailedUpdatedResults()`.

6.7.4.15 `const std::string ApproximateBayesianModelChecker::MSG_OUTPUT_RESULT_MIDDLE2 = ", and variance threshold value = " [static, private]`

Definition at line 196 of file ApproximateBayesianModelChecker.hpp.

Referenced by `getDetailedUpdatedResults()`.

6.7.4.16 `const std::string ApproximateBayesianModelChecker::MSG_OUTPUT_SEPARATOR = " " [static, private]`

Definition at line 199 of file ApproximateBayesianModelChecker.hpp.

Referenced by `getDetailedUpdatedResults()`.

6.7.4.17 `double multiscale::verification::ApproximateBayesianModelChecker::probability [private]`

The probability specified by the user for the logic property to be evaluated

Definition at line 55 of file ApproximateBayesianModelChecker.hpp.

Referenced by `initialise()`, and `isModelCheckingResultTrueConsideringComparator()`.

6.7.4.18 `double multiscale::verification::ApproximateBayesianModelChecker::variance [private]`

The value of the variance

Definition at line 62 of file ApproximateBayesianModelChecker.hpp.

Referenced by `initialise()`, `updateModelCheckingResult()`, and `updateVariance()`.

6.7.4.19 `double multiscale::verification::ApproximateBayesianModelChecker::varianceThreshold [private]`

The variance threshold

Definition at line 64 of file ApproximateBayesianModelChecker.hpp.

Referenced by ApproximateBayesianModelChecker(), getDetailedUpdatedResults(), and updateModelCheckingResult().

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

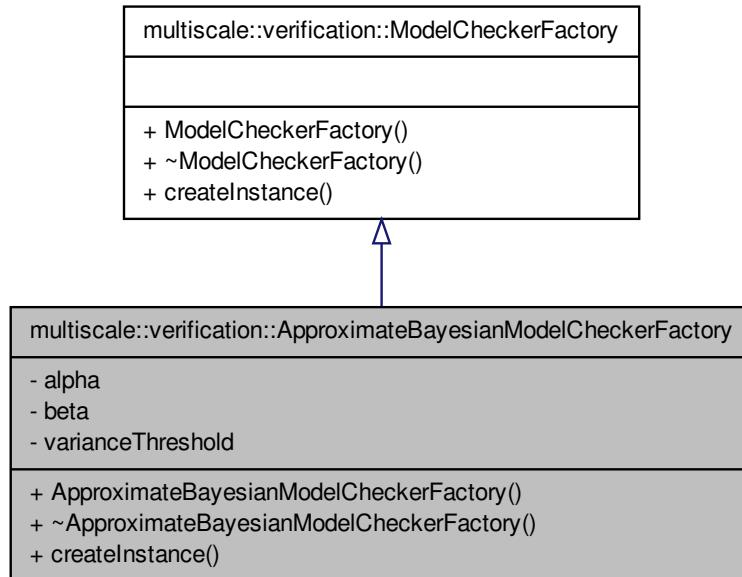
- ApproximateBayesianModelChecker.hpp
- ApproximateBayesianModelChecker.cpp

## 6.8 multiscale::verification::ApproximateBayesianModelChecker-Factory Class Reference

Class for creating [ApproximateBayesianModelChecker](#) instances.

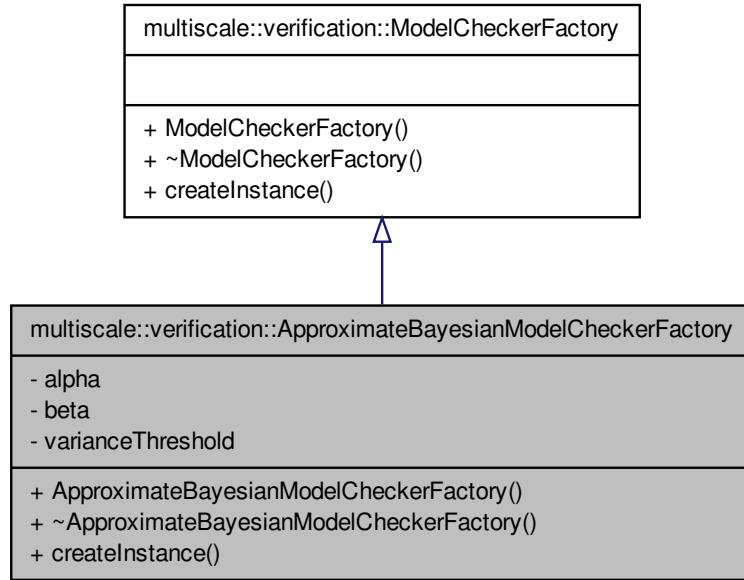
```
#include <ApproximateBayesianModelCheckerFactory.hpp>
```

Inheritance diagram for multiscale::verification::ApproximateBayesianModelChecker-Factory:



Collaboration diagram for multiscale::verification::ApproximateBayesianModelChecker-

Factory:



## Public Member Functions

- `ApproximateBayesianModelCheckerFactory` (double `alpha`, double `beta`, double `varianceThreshold`)
- `~ApproximateBayesianModelCheckerFactory ()`
- `std::shared_ptr< ModelChecker > createInstance (const AbstractSyntaxTree &abstractSyntaxTree, const MultiscaleArchitectureGraph &multiscaleArchitectureGraph) override`

*Create an instance of `ApproximateBayesianModelChecker`.*

## Private Attributes

- double `alpha`
- double `beta`
- double `varianceThreshold`

### 6.8.1 Detailed Description

Class for creating `ApproximateBayesianModelChecker` instances.

Definition at line 12 of file ApproximateBayesianModelCheckerFactory.hpp.

### 6.8.2 Constructor & Destructor Documentation

**6.8.2.1 ApproximateBayesianModelCheckerFactory::ApproximateBayesianModelCheckerFactory ( double *alpha*, double *beta*, double *varianceThreshold* )**

Definition at line 7 of file ApproximateBayesianModelCheckerFactory.cpp.

**6.8.2.2 ApproximateBayesianModelCheckerFactory::~ApproximateBayesianModelCheckerFactory ( )**

Definition at line 12 of file ApproximateBayesianModelCheckerFactory.cpp.

### 6.8.3 Member Function Documentation

**6.8.3.1 std::shared\_ptr< ModelChecker > ApproximateBayesianModelCheckerFactory::createInstance ( const AbstractSyntaxTree & *abstractSyntaxTree*, const MultiscaleArchitectureGraph & *multiscaleArchitectureGraph* ) [override, virtual]**

Create an instance of [ApproximateBayesianModelChecker](#).

#### Parameters

<i>abstract-SyntaxTree</i>	The abstract syntax tree representing the logic property to be checked
<i>multiscale-Architecture-Graph</i>	The multiscale architecture graph encoding the hierarchical organization of scales and subsystems

Implements [multiscale::verification::ModelCheckerFactory](#).

Definition at line 15 of file ApproximateBayesianModelCheckerFactory.cpp.

References alpha, beta, and varianceThreshold.

### 6.8.4 Member Data Documentation

**6.8.4.1 double multiscale::verification::ApproximateBayesianModelCheckerFactory::alpha [private]**

The shape parameter  $\alpha$  for the Beta distribution prior

Definition at line 16 of file ApproximateBayesianModelCheckerFactory.hpp.

Referenced by createInstance().

**6.8.4.2 double multiscale::verification::ApproximateBayesianModelCheckerFactory::beta [private]**

The shape parameter  $\beta$  for the Beta distribution prior

Definition at line 17 of file ApproximateBayesianModelCheckerFactory.hpp.

Referenced by createInstance().

**6.8.4.3 double multiscale::verification::ApproximateBayesianModelCheckerFactory::varianceThreshold [private]**

The variance threshold

Definition at line 19 of file ApproximateBayesianModelCheckerFactory.hpp.

Referenced by createInstance().

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

- ApproximateBayesianModelCheckerFactory.hpp
  
- ApproximateBayesianModelCheckerFactory.cpp

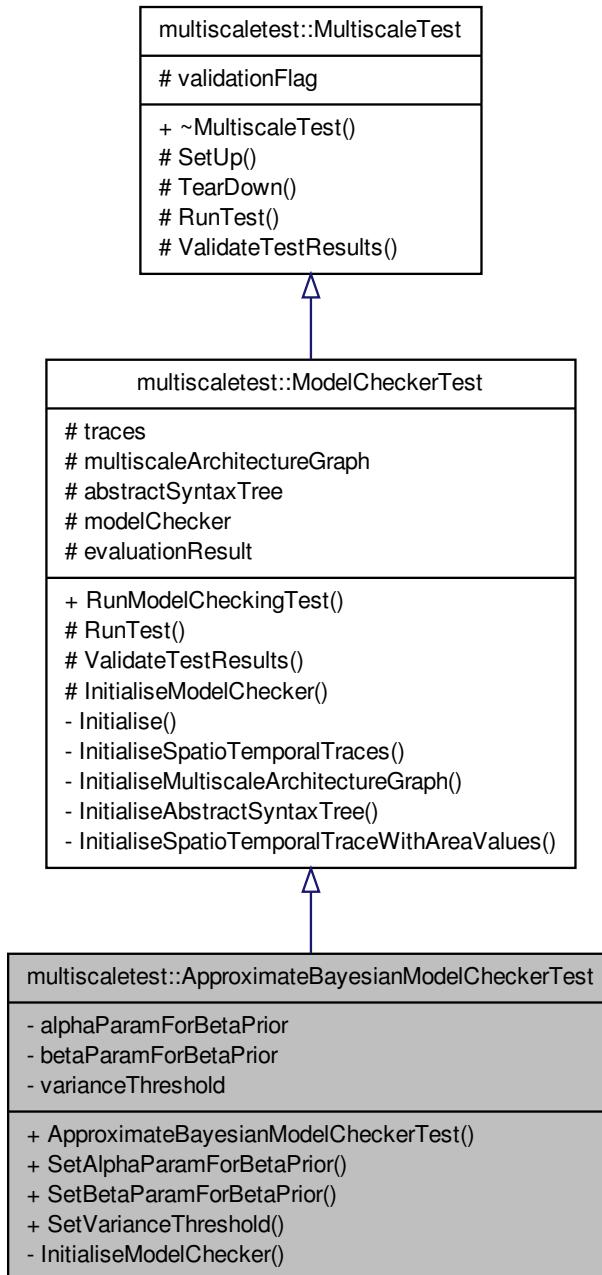
## 6.9 multiscaletest::ApproximateBayesianModelCheckerTest Class Reference

Class for testing the approximate Bayesian model checker.

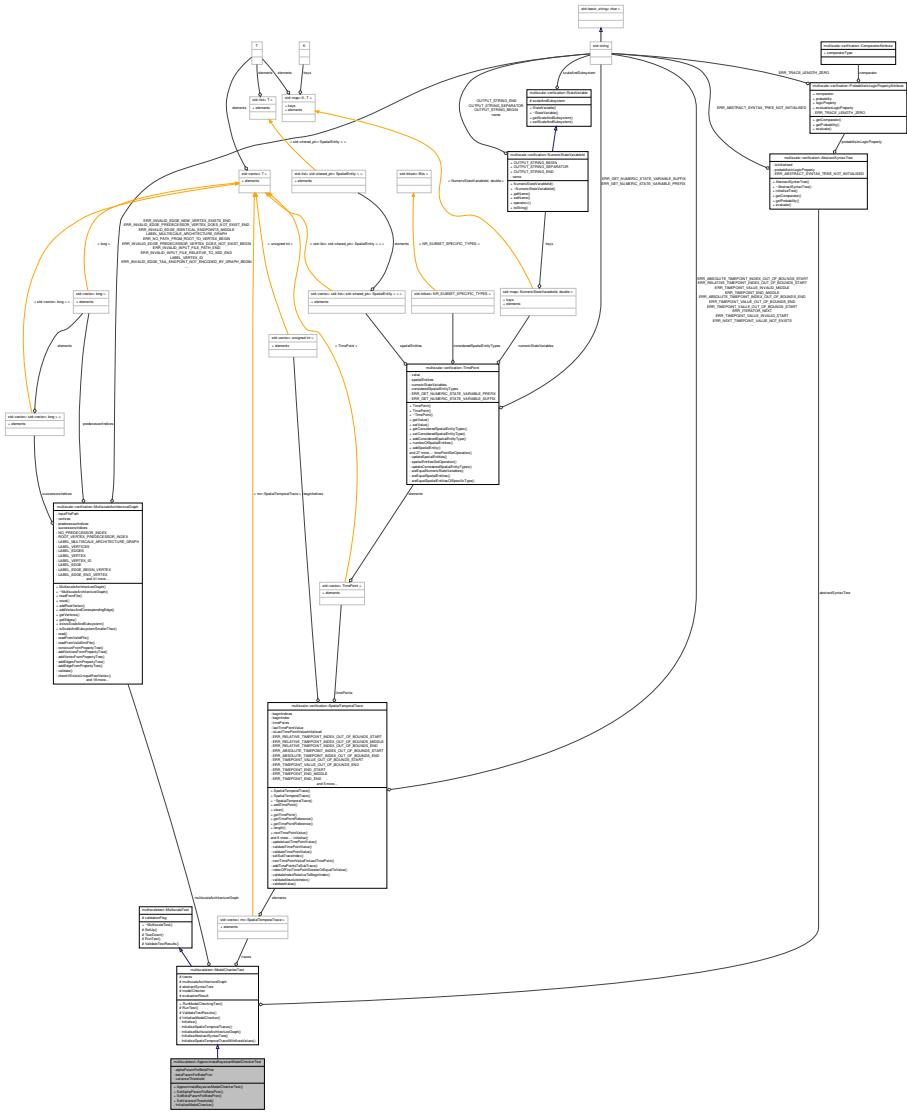
```
#include <ApproximateBayesianModelCheckerTest.hpp>
```

## 6.9 multiscaletest::ApproximateBayesianModelCheckerTest Class Reference 99

Inheritance diagram for multiscaletest::ApproximateBayesianModelCheckerTest:



## Collaboration diagram for multiscaletest::ApproximateBayesianModelCheckerTest:



## Public Member Functions

- `ApproximateBayesianModelCheckerTest ()`
  - void `SetAlphaParamForBetaPrior` (double `alphaParamForBetaPrior`)  
*Set the value of the alpha parameter for the beta prior.*
  - void `SetBetaParamForBetaPrior` (double `betaParamForBetaPrior`)  
*Set the value of the beta parameter for the beta prior.*
  - void `SetVarianceThreshold` (double `varianceThreshold`)  
*Set the value of the variance threshold.*

### Private Member Functions

- void [InitialiseModelChecker \(\) override](#)

*Initialise the model checker.*

### Private Attributes

- double [alphaParamForBetaPrior](#)
- double [betaParamForBetaPrior](#)
- double [varianceThreshold](#)

#### 6.9.1 Detailed Description

Class for testing the approximate Bayesian model checker.

Definition at line 15 of file ApproximateBayesianModelCheckerTest.hpp.

#### 6.9.2 Constructor & Destructor Documentation

##### 6.9.2.1 [multiscaletest::ApproximateBayesianModelCheckerTest::ApproximateBayesianModelCheckerTest \( \) \[inline\]](#)

Definition at line 26 of file ApproximateBayesianModelCheckerTest.hpp.

#### 6.9.3 Member Function Documentation

##### 6.9.3.1 [void multiscaletest::ApproximateBayesianModelCheckerTest::InitialiseModelChecker \( \) \[override, private, virtual\]](#)

Initialise the model checker.

Implements [multiscaletest::ModelCheckerTest](#).

Definition at line 68 of file ApproximateBayesianModelCheckerTest.hpp.

##### 6.9.3.2 [void multiscaletest::ApproximateBayesianModelCheckerTest::SetAlphaParamForBetaPrior \( double alphaParamForBetaPrior \)](#)

Set the value of the alpha parameter for the beta prior.

#### Parameters

<code>alphaParam- ForBetaPrior</code>	The alpha parameter for the beta prior
---	--

Definition at line 56 of file ApproximateBayesianModelCheckerTest.hpp.

```
6.9.3.3 void multiscaletest::ApproximateBayesianModelCheckerTest-
         ::SetBetaParamForBetaPrior ( double betaParamForBetaPrior
         )
```

Set the value of the beta parameter for the beta prior.

Parameters

<i>betaParam-</i> <i>ForBetaPrior</i>	The beta parameter for the beta prior
--	---------------------------------------

Definition at line 60 of file ApproximateBayesianModelCheckerTest.hpp.

```
6.9.3.4 void multiscaletest::ApproximateBayesianModelChecker-
         Test::SetVarianceThreshold ( double varianceThreshold
         )
```

Set the value of the variance threshold.

Parameters

<i>variance-</i> <i>Threshold</i>	The value of the variance threshold
--------------------------------------	-------------------------------------

Definition at line 64 of file ApproximateBayesianModelCheckerTest.hpp.

## 6.9.4 Member Data Documentation

```
6.9.4.1 double multiscaletest::ApproximateBayesianModelCheckerTest::alpha-
         ParamForBetaPrior [private]
```

The alpha parameter for the beta prior

Definition at line 19 of file ApproximateBayesianModelCheckerTest.hpp.

```
6.9.4.2 double multiscaletest::ApproximateBayesianModelCheckerTest::beta-
         ParamForBetaPrior [private]
```

The beta parameter for the beta prior

Definition at line 20 of file ApproximateBayesianModelCheckerTest.hpp.

**6.9.4.3 double multiscaletest::ApproximateBayesianModelCheckerTest::variance-Threshold [private]**

The considered variance threshold T

Definition at line 22 of file ApproximateBayesianModelCheckerTest.hpp.

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

- ApproximateBayesianModelCheckerTest.hpp

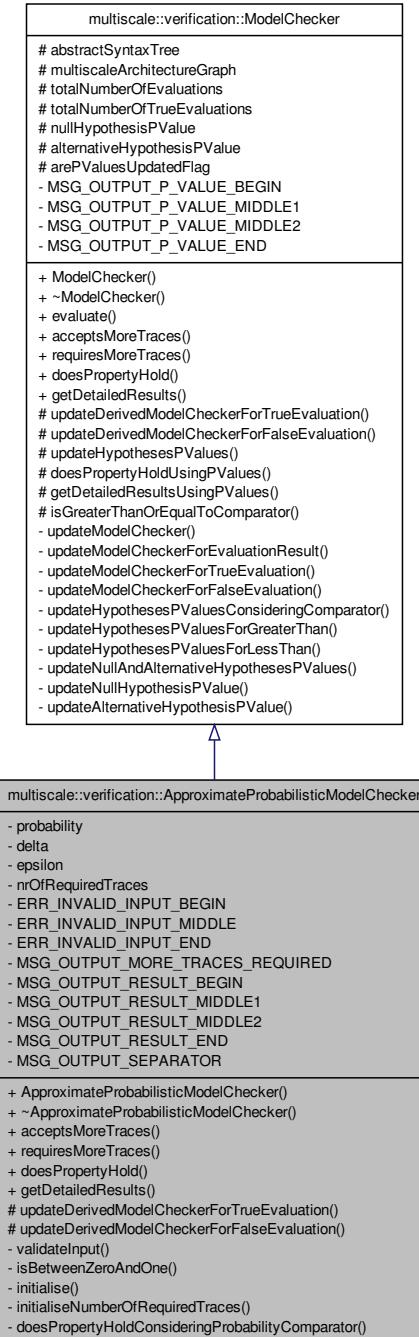
**6.10 multiscale::verification::ApproximateProbabilisticModel-  
Checker Class Reference**

Class used to run approximate probabilistic model checking tasks.

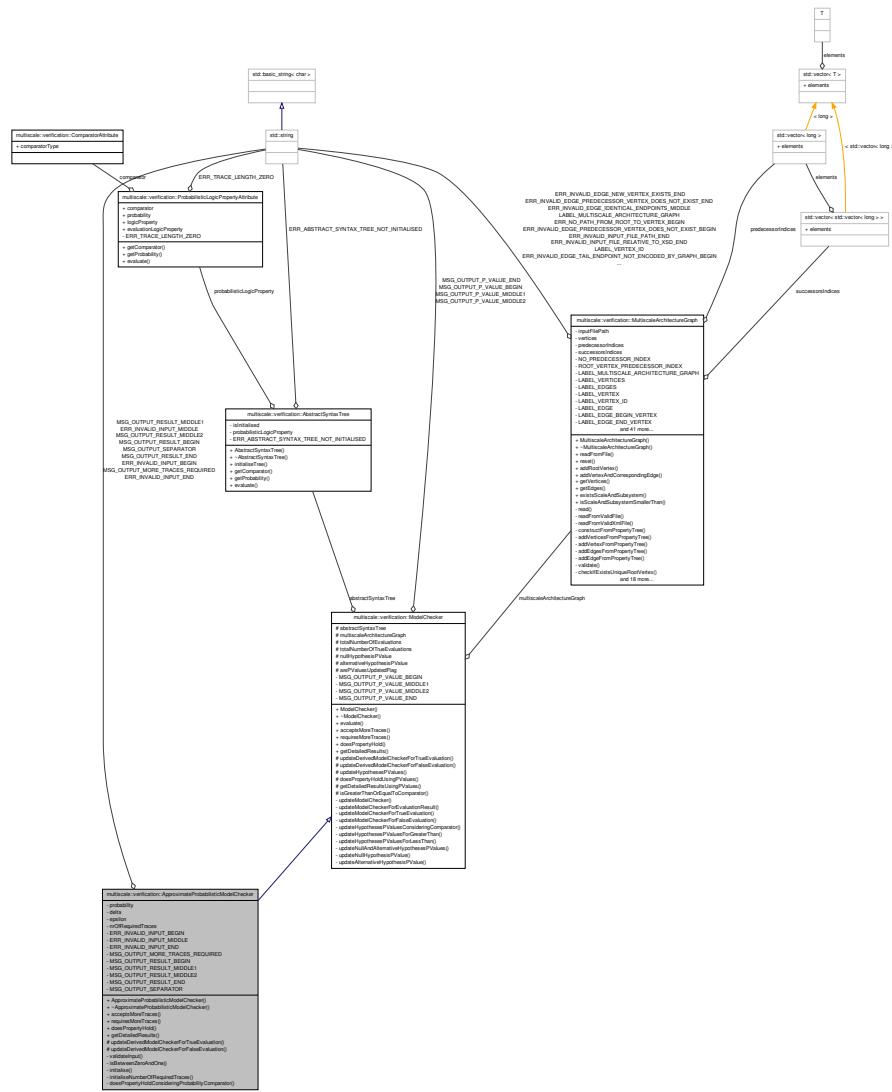
```
#include <ApproximateProbabilisticModelChecker.hpp>
```

Inheritance diagram for multiscale::verification::ApproximateProbabilisticModel-

Checker:



Collaboration diagram for multiscale::verification::ApproximateProbabilisticModelChecker:



## Public Member Functions

- `ApproximateProbabilisticModelChecker` (const `AbstractSyntaxTree` &`abstractSyntaxTree`, const `MultiscaleArchitectureGraph` &`multiscaleArchitectureGraph`, double `delta`, double `epsilon`)
  - `~ApproximateProbabilisticModelChecker` ()
  - bool `acceptsMoreTraces` () override

*Check if more traces are accepted for evaluating the logic property*

- bool `requiresMoreTraces ()` override  
*Check if more traces are required for evaluating the logic property.*
- bool `doesPropertyHold ()` override  
*Check if the given property holds.*
- std::string `getDetailedResults ()` override  
*Get the detailed description of the results.*

### Protected Member Functions

- void `updateDerivedModelCheckerForTrueEvaluation ()` override  
*Update the results of the derived model checker type considering that the logic property was evaluated to true for the last trace.*
- void `updateDerivedModelCheckerForFalseEvaluation ()` override  
*Update the results of the derived model checker type considering that the logic property was evaluated to false for the last trace.*

### Private Member Functions

- void `validateInput (double delta, double epsilon)`  
*Validate the input parameters delta and epsilon.*
- bool `isBetweenZeroAndOne (double value)`  
*Check if the given value is between zero and one (exclusive)*
- void `initialise ()`  
*Initialisation of some of the class members.*
- void `initialiseNumberOfRequiredTraces ()`  
*Initialise the number of required traces.*
- bool `doesPropertyHoldConsideringProbabilityComparator ()`  
*Check if the given property holds considering the probability comparator (i.e. <=, >=)*

### Private Attributes

- double `probability`
- double `delta`
- double `epsilon`
- unsigned int `nrOfRequiredTraces`

### Static Private Attributes

- static const std::string `ERR_INVALID_INPUT_BEGIN` = "The values of the provided input parameters `delta` and `epsilon` ("
- static const std::string `ERR_INVALID_INPUT_MIDDLE` = ", "
- static const std::string `ERR_INVALID_INPUT_END` = ") must be between zero and one (exclusive). Please change."

- static const std::string `MSG_OUTPUT_MORE_TRACES_REQUIRED` = "More traces are required to provide a true/false answer assuming the given upper bound on the `probability` of the computed `probability` to deviate from the true probability. Probabilistic black-box model checking was used instead to provide an answer."
- static const std::string `MSG_OUTPUT_RESULT_BEGIN` = "The provided answer is given assuming the upper bound on the `probability` to deviate more than `epsilon` = "
- static const std::string `MSG_OUTPUT_RESULT_MIDDLE1` = " from the true `probability` is `delta` = "
- static const std::string `MSG_OUTPUT_RESULT_MIDDLE2` = ". The number of required samples was N = "
- static const std::string `MSG_OUTPUT_RESULT_END` = ""
- static const std::string `MSG_OUTPUT_SEPARATOR` = " "

### 6.10.1 Detailed Description

Class used to run approximate probabilistic model checking tasks.

The implementation of this class is based on the algorithm described in the following paper:

T. Héault, R. Lassaigne, F. Magniette, and S. Peyronnet, ‘Approximate Probabilistic - Model Checking’, in Verification, Model Checking, and Abstract Interpretation, B. Steffen and G. Levi, Eds. Springer Berlin Heidelberg, 2004, pp. 73–84.

Definition at line 23 of file ApproximateProbabilisticModelChecker.hpp.

### 6.10.2 Constructor & Destructor Documentation

#### 6.10.2.1 ApproximateProbabilisticModelChecker::ApproximateProbabilisticModelChecker ( const AbstractSyntaxTree & abstractSyntaxTree, const MultiscaleArchitectureGraph & multiscaleArchitectureGraph, double delta, double epsilon )

Definition at line 10 of file ApproximateProbabilisticModelChecker.cpp.

References delta, epsilon, initialise(), and validateInput().

#### 6.10.2.2 ApproximateProbabilisticModelChecker::~ApproximateProbabilisticModelChecker ( )

Definition at line 28 of file ApproximateProbabilisticModelChecker.cpp.

### 6.10.3 Member Function Documentation

---

**6.10.3.1 bool ApproximateProbabilisticModelChecker::acceptsMoreTraces( )**  
 [override, virtual]

Check if more traces are accepted for evaluating the logic property.

Implements [multiscale::verification::ModelChecker](#).

Definition at line 30 of file ApproximateProbabilisticModelChecker.cpp.

References nrOfRequiredTraces, and multiscale::verification::ModelChecker::totalNumberOfEvaluations.

Referenced by requiresMoreTraces().

**6.10.3.2 bool ApproximateProbabilisticModelChecker::doesPropertyHold( )**  
 [override, virtual]

Check if the given property holds.

Implements [multiscale::verification::ModelChecker](#).

Definition at line 38 of file ApproximateProbabilisticModelChecker.cpp.

References doesPropertyHoldConsideringProbabilityComparator(), multiscale::verification::ModelChecker::doesPropertyHoldUsingPValues(), and requiresMoreTraces().

**6.10.3.3 bool ApproximateProbabilisticModelChecker::doesPropertyHoldConsideringProbabilityComparator( )**  
 [private]

Check if the given property holds considering the probability comparator (i.e.  $\leq$ ,  $\geq$ )

For queries of type : a)  $P \geq \theta[\phi]$  result =  $(nr_{true\_races}/nr\_races) - \epsilon \geq \theta$  b)  $P \leq \theta[\phi]$  result =  $(nr_{true\_races}/nr\_races) + \epsilon \leq \theta$

Definition at line 94 of file ApproximateProbabilisticModelChecker.cpp.

References epsilon, multiscale::Numeric::greaterOrEqual(), multiscale::verification::ModelChecker::isGreaterThanOrEqualToComparator(), multiscale::Numeric::lessOrEqual(), probability, multiscale::verification::ModelChecker::totalNumberOfEvaluations, and multiscale::verification::ModelChecker::totalNumberOfTrueEvaluations.

Referenced by doesPropertyHold().

**6.10.3.4 std::string ApproximateProbabilisticModelChecker::getDetailedResults( )**  
 [override, virtual]

Get the detailed description of the results.

Implements [multiscale::verification::ModelChecker](#).

Definition at line 46 of file ApproximateProbabilisticModelChecker.cpp.

References delta, epsilon, multiscale::verification::ModelChecker::getDetailedResultsUsingPValues(), MSG\_OUTPUT\_MORE\_TRACES\_REQUIRED, MSG\_OUTPUT\_RESULT\_BEGIN, MSG\_OUTPUT\_RESULT\_END, MSG\_OUTPUT\_RESULT\_MIDDLE1, MSG\_OUTPUT\_RESULT\_MIDDLE2, MSG\_OUTPUT\_SEPARATOR, nrOfRequiredTraces, requiresMoreTraces(), and multiscale::StringManipulator::toString().

#### 6.10.3.5 void ApproximateProbabilisticModelChecker::initialise( ) [private]

Initialisation of some of the class members.

Definition at line 81 of file ApproximateProbabilisticModelChecker.cpp.

References multiscale::verification::ModelChecker::abstractSyntaxTree, multiscale::verification::AbstractSyntaxTree::getProbability(), initialiseNumberOfRequiredTraces(), and probability.

Referenced by ApproximateProbabilisticModelChecker().

#### 6.10.3.6 void ApproximateProbabilisticModelChecker::initialiseNumberOfRequiredTraces( ) [private]

Initialise the number of required traces.

Precondition: The class members delta and epsilon are correctly initialised.

Definition at line 87 of file ApproximateProbabilisticModelChecker.cpp.

References delta, epsilon, and nrOfRequiredTraces.

Referenced by initialise().

#### 6.10.3.7 bool ApproximateProbabilisticModelChecker::isBetweenZeroAndOne( double value ) [private]

Check if the given value is between zero and one (exclusive)

Parameters

<i>value</i>	The given value
--------------	-----------------

Definition at line 77 of file ApproximateProbabilisticModelChecker.cpp.

Referenced by validateInput().

#### 6.10.3.8 bool ApproximateProbabilisticModelChecker::requiresMoreTraces( ) [override, virtual]

Check if more traces are required for evaluating the logic property.

Implements [multiscale::verification::ModelChecker](#).

Definition at line 34 of file ApproximateProbabilisticModelChecker.cpp.

References `acceptsMoreTraces()`.

Referenced by `doesPropertyHold()`, and `getDetailedResults()`.

```
6.10.3.9 void ApproximateProbabilisticModelChecker::updateDerivedModel-
    CheckerForFalseEvaluation( ) [override, protected,
    virtual]
```

Update the results of the derived model checker type considering that the logic property was evaluated to false for the last trace.

Do not do anything

Implements [multiscale::verification::ModelChecker](#).

Definition at line 64 of file `ApproximateProbabilisticModelChecker.cpp`.

```
6.10.3.10 void ApproximateProbabilisticModelChecker::updateDerivedModel-
    CheckerForTrueEvaluation( ) [override, protected,
    virtual]
```

Update the results of the derived model checker type considering that the logic property was evaluated to true for the last trace.

Do not do anything

Implements [multiscale::verification::ModelChecker](#).

Definition at line 62 of file `ApproximateProbabilisticModelChecker.cpp`.

```
6.10.3.11 void ApproximateProbabilisticModelChecker::validateInput( double delta,
    double epsilon ) [private]
```

Validate the input parameters `delta` and `epsilon`.

Precondition:  $0 < \delta, \epsilon < 1$

#### Parameters

<code>delta</code>	The upper bound on the probability to deviate from the true probability
<code>epsilon</code>	The considered amount by which the probability deviates from the true probability

Definition at line 66 of file `ApproximateProbabilisticModelChecker.cpp`.

References `ERR_INVALID_INPUT_BEGIN`, `ERR_INVALID_INPUT_END`, `ERR_INVALID_INPUT_MIDDLE`, `isBetweenZeroAndOne()`, and `multiscale::StringManipulator::toString()`.

Referenced by `ApproximateProbabilisticModelChecker()`.

#### 6.10.4 Member Data Documentation

6.10.4.1 `double multiscale::verification::ApproximateProbabilisticModelChecker-  
::delta [private]`

The upper bound on the probability for the computed probability to deviate from the true probability

Definition at line 30 of file ApproximateProbabilisticModelChecker.hpp.

Referenced by ApproximateProbabilisticModelChecker(), getDetailedResults(), and initialiseNumberOfRequiredTraces().

6.10.4.2 `double multiscale::verification::ApproximateProbabilisticModelChecker-  
::epsilon [private]`

The considered deviation from the true probability

Definition at line 32 of file ApproximateProbabilisticModelChecker.hpp.

Referenced by ApproximateProbabilisticModelChecker(), doesPropertyHoldConsideringProbabilityComparator(), getDetailedResults(), and initialiseNumberOfRequiredTraces().

6.10.4.3 `const std::string ApproximateProbabilisticModelChecker::ERR_INVALID_I-  
NPUT_BEGIN = "The values of the provided input parameters delta and epsilon ("`  
`[static, private]`

Definition at line 104 of file ApproximateProbabilisticModelChecker.hpp.

Referenced by validateInput().

6.10.4.4 `const std::string ApproximateProbabilisticModelChecker::ERR_INVALID-  
D_INPUT_END = ") must be between zero and one (exclusive). Please change."`  
`[static, private]`

Definition at line 106 of file ApproximateProbabilisticModelChecker.hpp.

Referenced by validateInput().

6.10.4.5 `const std::string ApproximateProbabilisticModelChecker-  
::ERR_INVALID_INPUT_MIDDLE = ", " [static,  
private]`

Definition at line 105 of file ApproximateProbabilisticModelChecker.hpp.

Referenced by validateInput().

6.10.4.6 `const std::string ApproximateProbabilisticModelChecker::MSG_OUTPUT_-  
MORE_TRACES_REQUIRED = "More traces are required to provide a true/false  
answer assuming the given upper bound on the probability of the computed  
probability to deviate from the true probability. Probabilistic black-box model  
checking was used instead to provide an answer." [static, private]`

Definition at line 108 of file ApproximateProbabilisticModelChecker.hpp.

Referenced by `getDetailedResults()`.

6.10.4.7 `const std::string ApproximateProbabilisticModelChecker::MSG_OUTPUT_-  
RESULT_BEGIN = "The provided answer is given assuming the upper bound on the  
probability to deviate more than epsilon =" [static, private]`

Definition at line 110 of file ApproximateProbabilisticModelChecker.hpp.

Referenced by `getDetailedResults()`.

6.10.4.8 `const std::string ApproximateProbabilisticModelChecker-  
::MSG_OUTPUT_RESULT_END = "" [static,  
private]`

Definition at line 113 of file ApproximateProbabilisticModelChecker.hpp.

Referenced by `getDetailedResults()`.

6.10.4.9 `const std::string ApproximateProbabilisticModelChecker::MSG_OUTPUT_-  
RESULT_MIDDLE1 = "from the true probability is delta =" [static,  
private]`

Definition at line 111 of file ApproximateProbabilisticModelChecker.hpp.

Referenced by `getDetailedResults()`.

6.10.4.10 `const std::string ApproximateProbabilisticModelChecker::MSG_OUT-  
PUT_RESULT_MIDDLE2 = ". The number of required samples was N ="  
[static, private]`

Definition at line 112 of file ApproximateProbabilisticModelChecker.hpp.

Referenced by `getDetailedResults()`.

6.10.4.11 `const std::string ApproximateProbabilisticModelChecker-  
::MSG_OUTPUT_SEPARATOR = " " [static,  
private]`

Definition at line 115 of file ApproximateProbabilisticModelChecker.hpp.

Referenced by `getDetailedResults()`.

6.10.4.12 `unsigned int multiscale::verification::ApproximateProbabilisticModelChecker::nrOfRequiredTraces [private]`

The number of required traces

Definition at line 34 of file ApproximateProbabilisticModelChecker.hpp.

Referenced by `acceptsMoreTraces()`, `getDetailedResults()`, and `initialiseNumberOfRequiredTraces()`.

6.10.4.13 `double multiscale::verification::ApproximateProbabilisticModelChecker::probability [private]`

The probability specified by the user for the logic property to be evaluated

Definition at line 27 of file ApproximateProbabilisticModelChecker.hpp.

Referenced by `doesPropertyHoldConsideringProbabilityComparator()`, and `initialise()`.

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

- ApproximateProbabilisticModelChecker.hpp
  
- ApproximateProbabilisticModelChecker.cpp

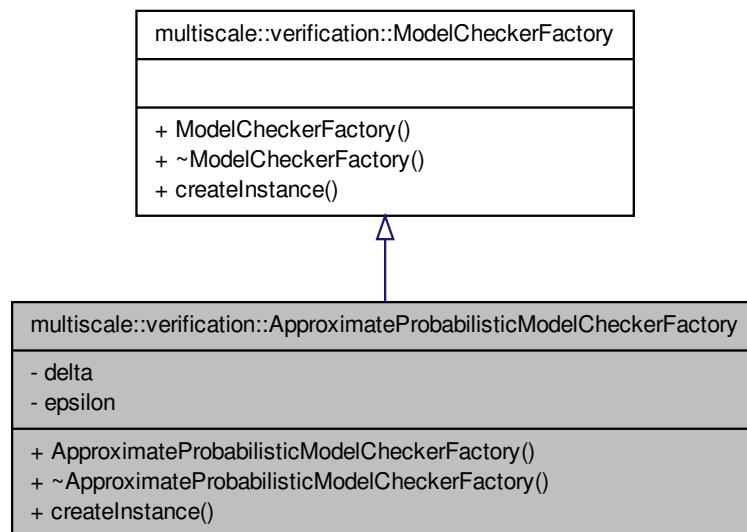
## 6.11 multiscale::verification::ApproximateProbabilisticModelCheckerFactory Class Reference

Class for creating `ApproximateProbabilisticModelChecker` instances.

```
#include <ApproximateProbabilisticModelCheckerFactory.-  
hpp>
```

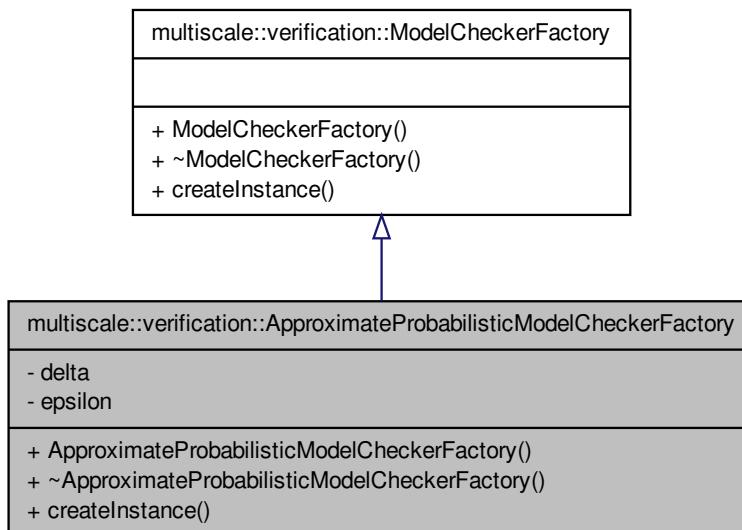
Inheritance diagram for multiscale::verification::ApproximateProbabilisticModel-

CheckerFactory:



Collaboration diagram for `multiscale::verification::ApproximateProbabilisticModel-`

CheckerFactory:



## Public Member Functions

- `ApproximateProbabilisticModelCheckerFactory (double delta, double epsilon)`
- `~ApproximateProbabilisticModelCheckerFactory ()`
- `std::shared_ptr< ModelChecker > createInstance (const AbstractSyntaxTree &abstractSyntaxTree, const MultiscaleArchitectureGraph &multiscaleArchitectureGraph) override`

*Create an instance of `ApproximateProbabilisticModelChecker`.*

## Private Attributes

- double `delta`
- double `epsilon`

### 6.11.1 Detailed Description

Class for creating `ApproximateProbabilisticModelChecker` instances.

Definition at line 12 of file `ApproximateProbabilisticModelCheckerFactory.hpp`.

### 6.11.2 Constructor & Destructor Documentation

6.11.2.1 **ApproximateProbabilisticModelCheckerFactory::ApproximateProbabilisticModelCheckerFactory ( double *delta*, double *epsilon* )**

Definition at line 7 of file ApproximateProbabilisticModelCheckerFactory.cpp.

6.11.2.2 **ApproximateProbabilisticModelCheckerFactory::~ApproximateProbabilisticModelCheckerFactory ( )**

Definition at line 11 of file ApproximateProbabilisticModelCheckerFactory.cpp.

### 6.11.3 Member Function Documentation

6.11.3.1 **std::shared\_ptr< ModelChecker > ApproximateProbabilisticModelCheckerFactory::createInstance ( const AbstractSyntaxTree & *abstractSyntaxTree*, const MultiscaleArchitectureGraph & *multiscaleArchitectureGraph* ) [override, virtual]**

Create an instance of [ApproximateProbabilisticModelChecker](#).

#### Parameters

<i>abstract-SyntaxTree</i>	The abstract syntax tree representing the logic property to be checked
<i>multiscale-Architecture-Graph</i>	The multiscale architecture graph encoding the hierarchical organization of scales and subsystems

Implements [multiscale::verification::ModelCheckerFactory](#).

Definition at line 14 of file ApproximateProbabilisticModelCheckerFactory.cpp.

References delta, and epsilon.

### 6.11.4 Member Data Documentation

6.11.4.1 **double multiscale::verification::ApproximateProbabilisticModelCheckerFactory::delta [private]**

The upper bound on the probability for the computed probability to deviate from the true probability

Definition at line 16 of file ApproximateProbabilisticModelCheckerFactory.hpp.

Referenced by `createInstance()`.

**6.11.4.2 double multiscale::verification::ApproximateProbabilisticModelChecker-Factory::epsilon [private]**

The considered deviation from the true probability

Definition at line 18 of file ApproximateProbabilisticModelCheckerFactory.hpp.

Referenced by createInstance().

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

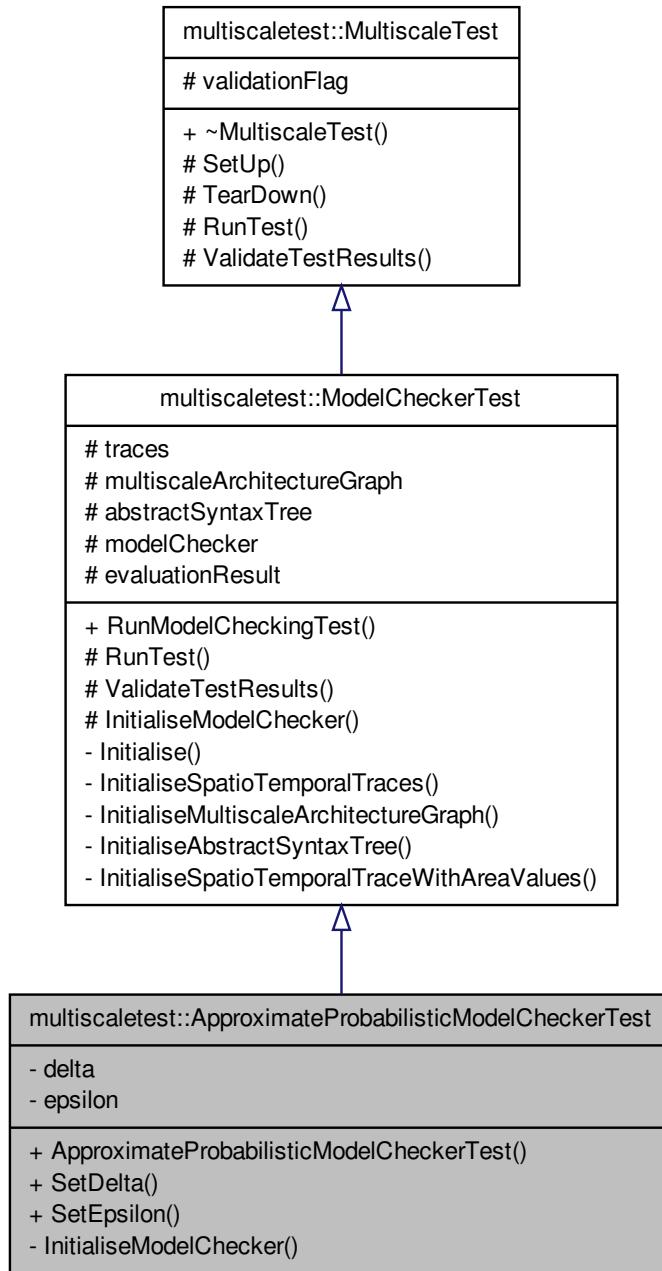
- ApproximateProbabilisticModelCheckerFactory.hpp
  
- ApproximateProbabilisticModelCheckerFactory.cpp

## **6.12 multiscaletest::ApproximateProbabilisticModelCheckerTest Class Reference**

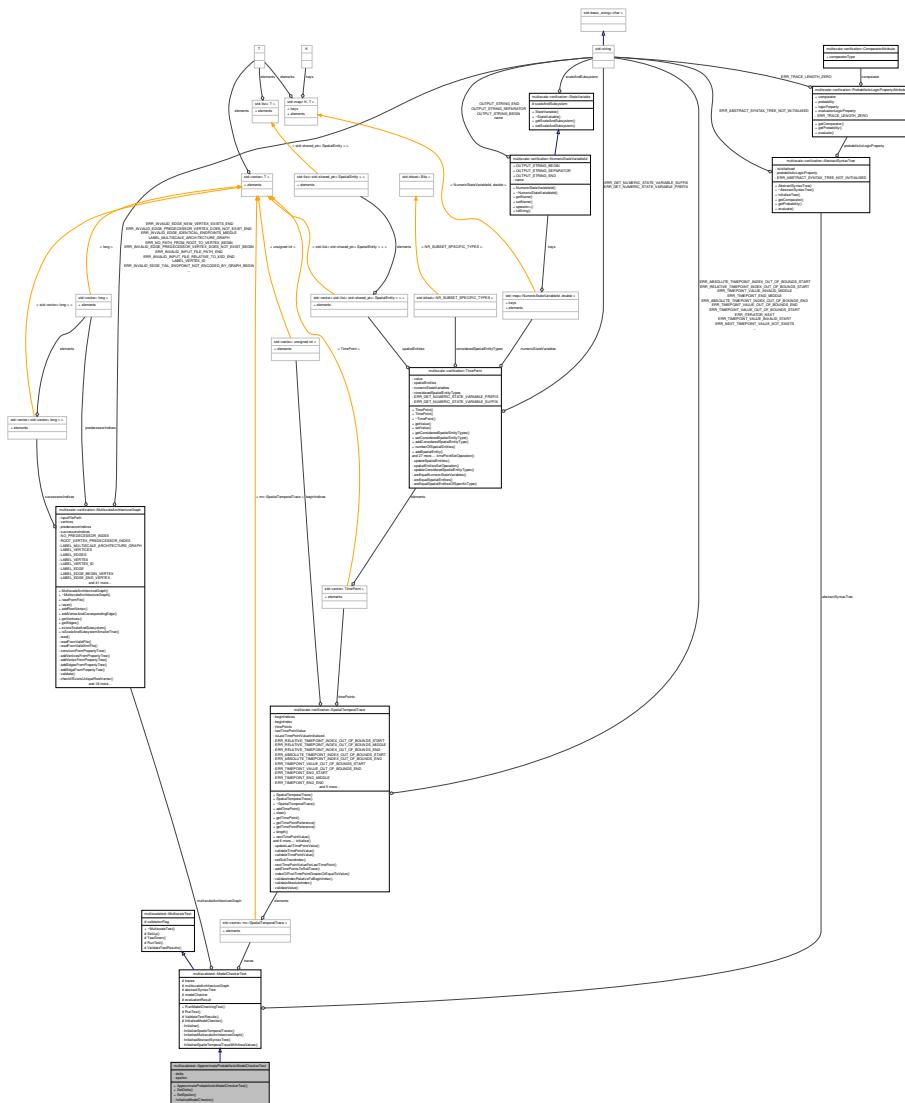
Class for testing the approximate probabilistic model checker.

```
#include <ApproximateProbabilisticModelCheckerTest.hpp>
```

Inheritance diagram for multiscaletest::ApproximateProbabilisticModelCheckerTest:



Collaboration diagram for multiscaletest::ApproximateProbabilisticModelCheckerTest:



# Public Member Functions

- `ApproximateProbabilisticModelCheckerTest ()`
  - `void SetDelta (double delta)`  
Set the value of `delta`.

Set the value of delta.

- void SetEpsilon (double epsilon)

*Set the value of epsilon.*

### Private Member Functions

- void [InitialiseModelChecker \(\) override](#)  
*Initialise the model checker.*

### Private Attributes

- double [delta](#)
- double [epsilon](#)

#### 6.12.1 Detailed Description

Class for testing the approximate probabilistic model checker.

Definition at line 15 of file ApproximateProbabilisticModelCheckerTest.hpp.

#### 6.12.2 Constructor & Destructor Documentation

##### 6.12.2.1 [multiscaletest::ApproximateProbabilisticModelChecker-Test::ApproximateProbabilisticModelCheckerTest \( \)](#) [inline]

Definition at line 24 of file ApproximateProbabilisticModelCheckerTest.hpp.

#### 6.12.3 Member Function Documentation

##### 6.12.3.1 [void multiscaletest::ApproximateProbabilisticModelChecker-Test::InitialiseModelChecker \( \)](#) [override, private, virtual]

Initialise the model checker.

Implements [multiscaletest::ModelCheckerTest](#).

Definition at line 55 of file ApproximateProbabilisticModelCheckerTest.hpp.

##### 6.12.3.2 [void multiscaletest::ApproximateProbabilisticModelCheckerTest::Set-Delta \( double delta \)](#)

Set the value of delta.

#### Parameters

<code>delta</code>	The value of delta
--------------------	--------------------

Definition at line 47 of file ApproximateProbabilisticModelCheckerTest.hpp.

6.12.3.3 void multiscaletest::ApproximateProbabilisticModelCheckerTest::SetEpsilon ( double *epsilon* )

Set the value of epsilon.

Parameters

<i>epsilon</i>	The value of epsilon
----------------	----------------------

Definition at line 51 of file ApproximateProbabilisticModelCheckerTest.hpp.

#### 6.12.4 Member Data Documentation

6.12.4.1 double multiscaletest::ApproximateProbabilisticModelCheckerTest::delta  
[private]

The value of delta in the Chernoff-Hoeffding inequality

Definition at line 19 of file ApproximateProbabilisticModelCheckerTest.hpp.

6.12.4.2 double multiscaletest::ApproximateProbabilisticModelCheckerTest::epsilon [private]

The value of epsilon in the Chernoff-Hoeffding inequality

Definition at line 20 of file ApproximateProbabilisticModelCheckerTest.hpp.

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

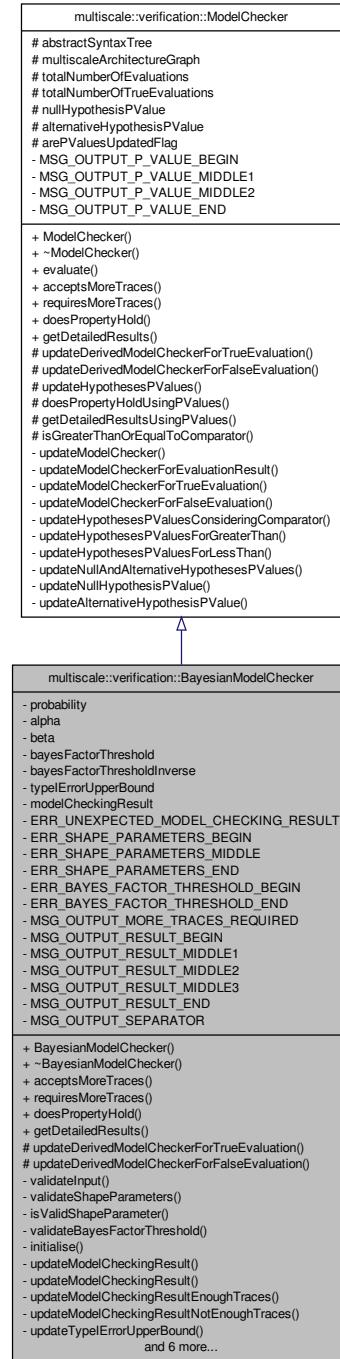
- ApproximateProbabilisticModelCheckerTest.hpp

## 6.13 multiscale::verification::BayesianModelChecker Class Reference

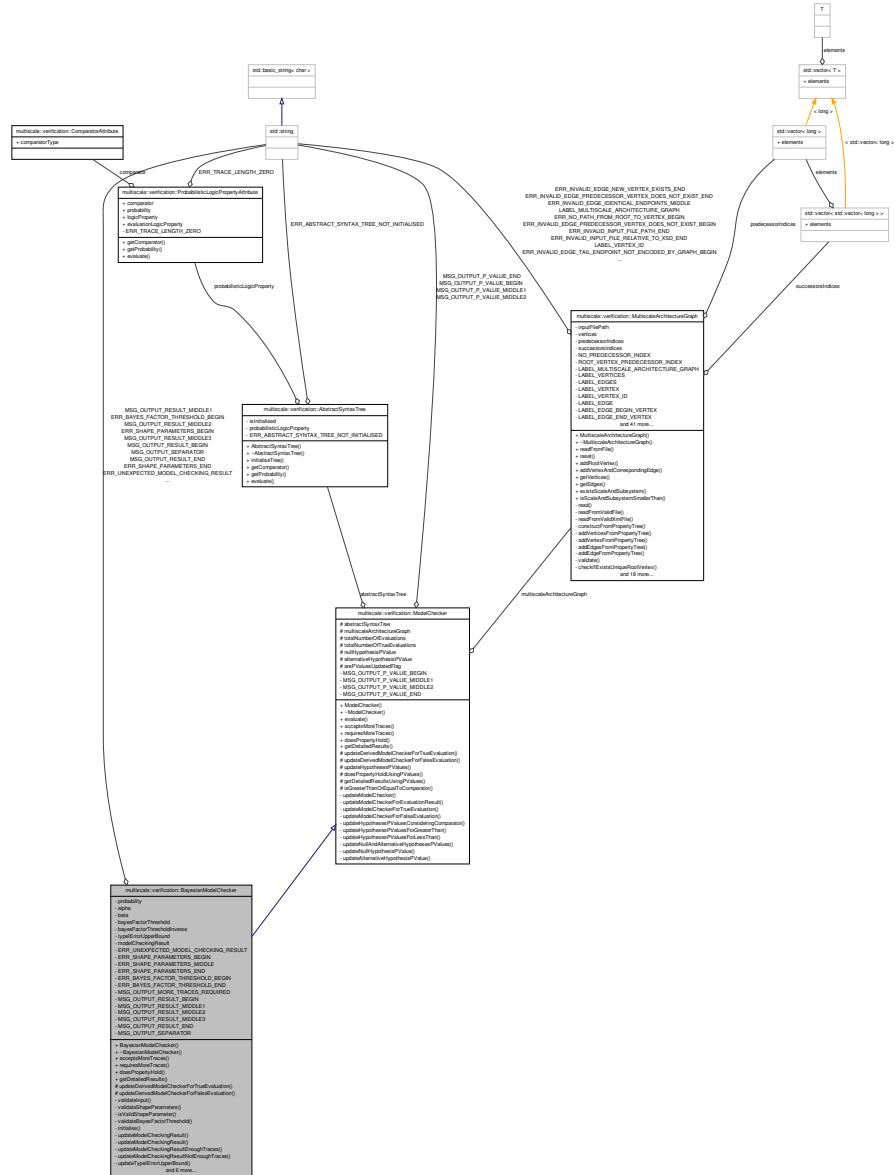
Class used to run Bayesian model checking tasks.

```
#include <BayesianModelChecker.hpp>
```

Inheritance diagram for multiscale::verification::BayesianModelChecker:



## Collaboration diagram for multiscale::verification::BayesianModelChecker:



## Public Member Functions

- `BayesianModelChecker` (const `AbstractSyntaxTree &abstractSyntaxTree`, const `MultiscaleArchitectureGraph &multiscaleArchitectureGraph`, double `alpha`, double `beta`, double `bayesFactorThreshold`)
  - `~BayesianModelChecker ()`
  - bool `acceptsMoreTraces ()` override

*Check if more traces are accepted for evaluating the logic property.*

- bool `requiresMoreTraces ()` override

*Check if more traces are required for evaluating the logic property.*

- bool `doesPropertyHold ()` override

*Check if the given property holds.*

- std::string `getDetailedResults ()` override

*Get the detailed description of the results.*

## Protected Member Functions

- void `updateDerivedModelCheckerForTrueEvaluation ()` override

*Update the results of the derived model checker type considering that the logic property was evaluated to true for the last trace.*

- void `updateDerivedModelCheckerForFalseEvaluation ()` override

*Update the results of the derived model checker type considering that the logic property was evaluated to false for the last trace.*

## Private Member Functions

- void `validateInput (double alpha, double beta, double bayesFactorThreshold)`

*Validate the input parameters  $\alpha$ ,  $\beta$  and the Bayes factor threshold.*

- void `validateShapeParameters (double alpha, double beta)`

*Validate the shape parameters  $\alpha$  and  $\beta$ .*

- bool `isValidShapeParameter (double shapeParameter)`

*Check if the given shape parameter value is valid.*

- void `validateBayesFactorThreshold (double bayesFactorThreshold)`

*Validate the Bayes factor threshold.*

- void `initialise ()`

*Initialisation of some of the class members.*

- void `updateModelCheckingResult ()`

*Update the result of the model checking task.*

- void `updateModelCheckingResult (double bayesFactor)`

*Update the result of the model checking task considering the given Bayes factor value.*

- void `updateModelCheckingResultEnoughTraces (double bayesFactor)`

*Update the result of the model checking task considering that enough traces have been provided.*

- void `updateModelCheckingResultNotEnoughTraces ()`

*Update the result of the model checking task considering that not enough traces have been provided.*

- void `updateTypeIErrorUpperBound ()`

*Update the value of the type I error upper bound.*

- bool `indicatorFunction (unsigned int nrOfSuccesses)`

*Compute the value of the indicator function  $I_{\mathcal{B}(n,x) < 1/T}(x)$ .*

- double `computeMaximumBinomialPDF` (unsigned int nrOfSuccesses)  
*Compute the maximum value of the probability distribution function for the Binomial distribution.*
- double `computeBinomialPDF` (unsigned int nrOfSuccesses, double `probability`)  
*Compute the value of the probability distribution function for the Binomial distribution.*
- double `computeBayesFactorValue` (unsigned int nrOfObservations, unsigned int nrOfSuccesses)  
*Compute the value of the Bayes factor.*
- bool `doesPropertyHoldConsideringResult` ()  
*Check if the given property holds considering the obtained model checking result.*
- bool `doesPropertyHoldConsideringProbabilityComparator` (bool isNullHypothesis-True)  
*Check if the given property holds considering the obtained answer and probability comparator (i.e. <=, >=)*
- std::string `getDetailedUpdatedResults` ()  
*Get the detailed description of the updated results.*

### Private Attributes

- double `probability`
- double `alpha`
- double `beta`
- double `bayesFactorThreshold`
- double `bayesFactorThresholdInverse`
- double `typeIErrorUpperBound`
- BayesianModelCheckingResult `modelCheckingResult`

### Static Private Attributes

- static const std::string `ERR_UNEXPECTED_MODEL_CHECKING_RESULT` = "-An invalid Bayesian model checking result was obtained. Please check source code."
- static const std::string `ERR_SHAPE_PARAMETERS_BEGIN` = "The provided - Beta distribution shape parameters `alpha` and `beta` ("
- static const std::string `ERR_SHAPE_PARAMETERS_MIDDLE` = ", "
- static const std::string `ERR_SHAPE_PARAMETERS_END` = ") should be greater than zero. Please change."
- static const std::string `ERR_BAYES_FACTOR_THRESHOLD_BEGIN` = "The provided Bayes factor threshold ("
- static const std::string `ERR_BAYES_FACTOR_THRESHOLD_END` = ") should be greater than one. Please change."
- static const std::string `MSG_OUTPUT_MORE_TRACES_REQUIRED` = "More traces are required to provide a true/false answer assuming the given Beta distribution shape parameters and Bayes factor threshold value. Probabilistic black-box model checking was used instead to provide an answer."

- static const std::string `MSG_OUTPUT_RESULT_BEGIN` = "The provided answer is given for the Beta distribution shape parameters `alpha` = "
- static const std::string `MSG_OUTPUT_RESULT_MIDDLE1` = " and `beta` = "
- static const std::string `MSG_OUTPUT_RESULT_MIDDLE2` = ", and Bayes factor threshold value = "
- static const std::string `MSG_OUTPUT_RESULT_MIDDLE3` = ". The type I error upper bound for the provided answer is = "
- static const std::string `MSG_OUTPUT_RESULT_END` = ""
- static const std::string `MSG_OUTPUT_SEPARATOR` = " "

### 6.13.1 Detailed Description

Class used to run Bayesian model checking tasks.

The implementation of this class is (partially) based on the algorithms described in the following paper:

S. K. Jha, E. M. Clarke, C. J. Langmead, A. Legay, A. Platzer, and P. Zuliani, 'A - Bayesian Approach to Model Checking Biological Systems', in Computational Methods in Systems Biology, P. Degano and R. Gorrieri, Eds. Springer Berlin Heidelberg, 2009, pp. 218–234.

In our implementation the variables in the original paper (right hand side of the assignments) have been given the following new names (left hand side of assignments):

`probability` =  $\theta$

`alpha` =  $\alpha$

`beta` =  $\beta$

`bayesFactor` =  $\mathcal{B}_n$

`bayesFactorThreshold` =  $T$

`totalNumberOfEvaluations` =  $n$

`totalNumberOfTrueEvaluations` =  $x$

Definition at line 50 of file `BayesianModelChecker.hpp`.

### 6.13.2 Constructor & Destructor Documentation

6.13.2.1 `BayesianModelChecker::BayesianModelChecker ( const AbstractSyntaxTree & abstractSyntaxTree, const MultiscaleArchitectureGraph & multiscaleArchitectureGraph, double alpha, double beta, double bayesFactorThreshold )`

Definition at line 15 of file `BayesianModelChecker.cpp`.

References `alpha`, `bayesFactorThreshold`, `beta`, `initialise()`, and `validateInput()`.

### 6.13.2.2 BayesianModelChecker::~BayesianModelChecker( )

Definition at line 31 of file BayesianModelChecker.cpp.

### 6.13.3 Member Function Documentation

#### 6.13.3.1 bool BayesianModelChecker::acceptsMoreTraces( ) [override, virtual]

Check if more traces are accepted for evaluating the logic property.

Implements [multiscale::verification::ModelChecker](#).

Definition at line 33 of file BayesianModelChecker.cpp.

References [modelCheckingResult](#), and [updateModelCheckingResult\(\)](#).

Referenced by [requiresMoreTraces\(\)](#).

#### 6.13.3.2 double BayesianModelChecker::computeBayesFactorValue( unsigned int nrOfObservations, unsigned int nrOfSuccesses ) [private]

Compute the value of the Bayes factor.

According to the original paper the Bayes factor can be computed as follows:  $\mathcal{B} = \frac{1}{(F_{x+\alpha, n-x+\beta})(\theta)} - 1$

#### Parameters

<i>nrOfObservations</i>	The total number of observations
<i>nrOfSuccesses</i>	The total number of successes

Definition at line 161 of file BayesianModelChecker.cpp.

References [multiscale::Numeric::almostEqual\(\)](#), [alpha](#), [beta](#), [multiscale::Beta-Distribution::cdf\(\)](#), and [probability](#).

Referenced by [indicatorFunction\(\)](#), and [updateModelCheckingResult\(\)](#).

#### 6.13.3.3 double BayesianModelChecker::computeBinomialPDF( unsigned int nrOfSuccesses, double probability ) [private]

Compute the value of the probability distribution function for the Binomial distribution.

#### Parameters

<i>nrOfSuccesses</i>	The number of successful observations/trials
<i>probability</i>	The probability of success

Definition at line 155 of file BayesianModelChecker.cpp.

References multiscale::BinomialDistribution::pdf(), and multiscale::verification::ModelChecker::totalNumberOfEvaluations.

Referenced by computeMaximumBinomialPDF().

#### 6.13.3.4 double BayesianModelChecker::computeMaximumBinomialPDF ( unsigned int nrOfSuccesses ) [private]

Compute the maximum value of the probability distribution function for the Binomial distribution.

The maximum value is reached when  $p = \theta$  or  $p = \frac{2k}{n}$

##### Parameters

<i>nrOf- Successes</i>	The number of successful observations/trials
----------------------------	--

Definition at line 145 of file BayesianModelChecker.cpp.

References computeBinomialPDF(), probability, and multiscale::verification::ModelChecker::totalNumberOfEvaluations.

Referenced by updateTypeIErrorUpperBound().

#### 6.13.3.5 bool BayesianModelChecker::doesPropertyHold ( ) [override, virtual]

Check if the given property holds.

Implements [multiscale::verification::ModelChecker](#).

Definition at line 43 of file BayesianModelChecker.cpp.

References doesPropertyHoldConsideringResult(), and updateModelCheckingResult().

#### 6.13.3.6 bool BayesianModelChecker::doesPropertyHoldConsidering- ProbabilityComparator ( bool isNullHypothesisTrue ) [private]

Check if the given property holds considering the obtained answer and probability comparator (i.e.  $\leq$ ,  $\geq$ )

For queries of type : a)  $P \geq \theta[\phi]$  the isNullHypothesisTrue flag value is returned b)  $P \leq \theta[\phi]$  the !isNullHypothesisTrue flag value is returned

##### Parameters

<i>isNull- Hypothesis- True</i>	Flag indicating if the null hypothesis is true considering a $P \geq [\phi]$ query
---	--

Definition at line 192 of file BayesianModelChecker.cpp.

References multiscale::verification::ModelChecker::isGreaterThanOrEqualToComparator().

Referenced by doesPropertyHoldConsideringResult().

#### 6.13.3.7 bool BayesianModelChecker::doesPropertyHoldConsideringResult( ) [private]

Check if the given property holds considering the obtained model checking result.

Definition at line 173 of file BayesianModelChecker.cpp.

References doesPropertyHoldConsideringProbabilityComparator(), multiscale::verification::ModelChecker::doesPropertyHoldUsingPValues(), ERR\_UNEXPECTED\_MODEL\_CHECKING\_RESULT, and modelCheckingResult.

Referenced by doesPropertyHold().

#### 6.13.3.8 std::string BayesianModelChecker::getDetailedResults( ) [override, virtual]

Get the detailed description of the results.

Implements [multiscale::verification::ModelChecker](#).

Definition at line 49 of file BayesianModelChecker.cpp.

References getDetailedUpdatedResults(), and updateModelCheckingResult().

#### 6.13.3.9 std::string BayesianModelChecker::getDetailedUpdatedResults( ) [private]

Get the detailed description of the updated results.

Definition at line 200 of file BayesianModelChecker.cpp.

References alpha, bayesFactorThreshold, beta, multiscale::verification::ModelChecker::getDetailedResultsUsingPValues(), modelCheckingResult, MSG\_OUTPUT\_MORE\_TRACES\_REQUIRED, MSG\_OUTPUT\_RESULT\_BEGIN, MSG\_OUTPUT\_RESULT\_END, MSG\_OUTPUT\_RESULT\_MIDDLE1, MSG\_OUTPUT\_RESULT\_MIDDLE2, MSG\_OUTPUT\_RESULT\_MIDDLE3, MSG\_OUTPUT\_SEPARATOR, multiscale::StringManipulator::toString(), and typeErrorUpperBound.

Referenced by getDetailedResults().

#### 6.13.3.10 bool BayesianModelChecker::indicatorFunction( unsigned int nrOfSuccesses ) [private]

Compute the value of the indicator function  $I_{\mathcal{B}(n,x) < 1/T}(x)$ .

**Parameters**

<i>nrOf- Successes</i>	The number of successful observations/trials
----------------------------	--

Definition at line 139 of file BayesianModelChecker.cpp.

References bayesFactorThresholdInverse, computeBayesFactorValue(), and multiscale-::verification::ModelChecker::totalNumberOfEvaluations.

Referenced by updateTypeIErrorUpperBound().

**6.13.3.11 void BayesianModelChecker::initialise( ) [private]**

Initialisation of some of the class members.

Definition at line 90 of file BayesianModelChecker.cpp.

References multiscale::verification::ModelChecker::abstractSyntaxTree, multiscale-::Numeric::almostEqual(), bayesFactorThreshold, bayesFactorThresholdInverse, multiscale::verification::AbstractSyntaxTree::getProbability(), probability, and typeI-ErrorUpperBound.

Referenced by BayesianModelChecker().

**6.13.3.12 bool BayesianModelChecker::isValidShapeParameter( double  
shapeParameter ) [private]**

Check if the given shape parameter value is valid.

The shape parameter values should be greater than zero

**Parameters**

<i>shape- Parameter</i>	The given shape parameter
-----------------------------	---------------------------

Definition at line 75 of file BayesianModelChecker.cpp.

Referenced by validateShapeParameters().

**6.13.3.13 bool BayesianModelChecker::requiresMoreTraces( ) [override,  
virtual]**

Check if more traces are required for evaluating the logic property.

Implements [multiscale::verification::ModelChecker](#).

Definition at line 39 of file BayesianModelChecker.cpp.

References acceptsMoreTraces().

6.13.3.14 void BayesianModelChecker::updateDerivedModelChecker-  
ForFalseEvaluation ( ) [override, protected,  
virtual]

Update the results of the derived model checker type considering that the logic property was evaluated to false for the last trace.

Do not do anything

Implements [multiscale::verification::ModelChecker](#).

Definition at line 57 of file BayesianModelChecker.cpp.

6.13.3.15 void BayesianModelChecker::updateDerivedModelChecker-  
ForTrueEvaluation ( ) [override, protected,  
virtual]

Update the results of the derived model checker type considering that the logic property was evaluated to true for the last trace.

Do not do anything

Implements [multiscale::verification::ModelChecker](#).

Definition at line 55 of file BayesianModelChecker.cpp.

6.13.3.16 void BayesianModelChecker::updateModelCheckingResult ( )  
[private]

Update the result of the model checking task.

Definition at line 100 of file BayesianModelChecker.cpp.

References `computeBayesFactorValue()`, `multiscale::verification::ModelChecker::total-NumberOfWorkers`, `multiscale::verification::ModelChecker::totalNumberOfTrue-Evaluations`, and `updateTypeIErrorUpperBound()`.

Referenced by `acceptsMoreTraces()`, `doesPropertyHold()`, and `getDetailedResults()`.

6.13.3.17 void BayesianModelChecker::updateModelCheckingResult ( double  
*bayesFactor* ) [private]

Update the result of the model checking task considering the given Bayes factor value.

Parameters

<i>bayesFactor</i>	The given Bayes factor value
--------------------	------------------------------

Definition at line 108 of file BayesianModelChecker.cpp.

References `bayesFactorThreshold`, `bayesFactorThresholdInverse`, `updateModelCheckingResultEnoughTraces()`, and `updateModelCheckingResultNotEnoughTraces()`.

---

**6.13.3.18 void BayesianModelChecker::updateModelCheckingResultEnough-Traces ( double *bayesFactor* ) [private]**

Update the result of the model checking task considering that enough traces have been provided.

**Parameters**

<i>bayesFactor</i>	The given Bayes factor value
--------------------	------------------------------

Definition at line 117 of file BayesianModelChecker.cpp.

References *bayesFactorThreshold*, *bayesFactorThresholdInverse*, *multiscale::verification::FALSE*, *modelCheckingResult*, and *multiscale::verification::TRUE*.

Referenced by *updateModelCheckingResult()*.

**6.13.3.19 void BayesianModelChecker::updateModelCheckingResultNotEnough-Traces ( ) [private]**

Update the result of the model checking task considering that not enough traces have been provided.

Definition at line 125 of file BayesianModelChecker.cpp.

References *modelCheckingResult*, and *multiscale::verification::MORE\_TRACES\_REQUIRED*.

Referenced by *updateModelCheckingResult()*.

**6.13.3.20 void BayesianModelChecker::updateTypeIErrorUpperBound ( ) [private]**

Update the value of the type I error upper bound.

Definition at line 129 of file BayesianModelChecker.cpp.

References *computeMaximumBinomialPDF()*, *indicatorFunction()*, *multiscale::verification::ModelChecker::totalNumberOfEvaluations*, and *typeIErrorUpperBound*.

Referenced by *updateModelCheckingResult()*.

**6.13.3.21 void BayesianModelChecker::validateBayesFactorThreshold ( double *bayesFactorThreshold* ) [private]**

Validate the Bayes factor threshold.

The Bayes factor threshold should be greater than 1

**Parameters**

<i>bayesFactor-Threshold</i>	The Bayes factor threshold
------------------------------	----------------------------

Definition at line 79 of file BayesianModelChecker.cpp.

References `ERR_BAYES_FACTOR_THRESHOLD_BEGIN`, `ERR_BAYES_FACTOR_THRESHOLD_END`, `multiscale::Numeric::lessOrEqual()`, and `multiscale::StringManipulator::toString()`.

Referenced by `validateInput()`.

**6.13.3.22 void BayesianModelChecker::validateInput ( double *alpha*, double *beta*, double *bayesFactorThreshold* ) [private]**

Validate the input parameters  $\alpha$ ,  $\beta$  and the Bayes factor threshold.

$\alpha$  and  $\beta$  should be greater than zero, and Bayes factor threshold should be greater than 1

#### Parameters

<i>alpha</i>	The shape parameter $\alpha$ for the Beta distribution
<i>beta</i>	The shape parameter $\beta$ for the Beta distribution
<i>bayesFactorThreshold</i>	The Bayes factor threshold

Definition at line 59 of file BayesianModelChecker.cpp.

References `validateBayesFactorThreshold()`, and `validateShapeParameters()`.

Referenced by `BayesianModelChecker()`.

**6.13.3.23 void BayesianModelChecker::validateShapeParameters ( double *alpha*, double *beta* ) [private]**

Validate the shape parameters  $\alpha$  and  $\beta$ .

$\alpha$  and  $\beta$  should be greater than zero

#### Parameters

<i>alpha</i>	The shape parameter $\alpha$ for the Beta distribution
<i>beta</i>	The shape parameter $\beta$ for the Beta distribution

Definition at line 64 of file BayesianModelChecker.cpp.

References `ERR_SHAPE_PARAMETERS_BEGIN`, `ERR_SHAPE_PARAMETERS_END`, `ERR_SHAPE_PARAMETERS_MIDDLE`, `isValidShapeParameter()`, and `multiscale::StringManipulator::toString()`.

Referenced by `validateInput()`.

## 6.13.4 Member Data Documentation

**6.13.4.1 double multiscale::verification::BayesianModelChecker::alpha  
[private]**

The shape parameter  $\alpha$  for the Beta distribution prior

Definition at line 57 of file BayesianModelChecker.hpp.

Referenced by BayesianModelChecker(), computeBayesFactorValue(), and getDetailedUpdatedResults().

**6.13.4.2 double multiscale::verification::BayesianModelChecker::bayesFactor-  
Threshold [private]**

The Bayes factor threshold

Definition at line 60 of file BayesianModelChecker.hpp.

Referenced by BayesianModelChecker(), getDetailedUpdatedResults(), initialise(), updateModelCheckingResult(), and updateModelCheckingResultEnoughTraces().

**6.13.4.3 double multiscale::verification::BayesianModelChecker::bayesFactor-  
ThresholdInverse [private]**

The Bayes factor threshold to the power "-1"

Definition at line 61 of file BayesianModelChecker.hpp.

Referenced by indicatorFunction(), initialise(), updateModelCheckingResult(), and updateModelCheckingResultEnoughTraces().

**6.13.4.4 double multiscale::verification::BayesianModelChecker::beta  
[private]**

The shape parameter  $\beta$  for the Beta distribution prior

Definition at line 58 of file BayesianModelChecker.hpp.

Referenced by BayesianModelChecker(), computeBayesFactorValue(), and getDetailedUpdatedResults().

**6.13.4.5 const std::string BayesianModelChecker::ERR\_BAYES\_FACTOR\_THR-  
ESHOLD\_BEGIN = "The provided Bayes factor threshold (" [static,  
private]**

Definition at line 211 of file BayesianModelChecker.hpp.

Referenced by validateBayesFactorThreshold().

6.13.4.6 `const std::string BayesianModelChecker::ERR_BAYES_FACTOR_THRESHOLD_END = ") should be greater than one. Please change." [static, private]`

Definition at line 212 of file BayesianModelChecker.hpp.

Referenced by validateBayesFactorThreshold().

6.13.4.7 `const std::string BayesianModelChecker::ERR_SHAPE_PARAMETERS_BEGIN = "The provided Beta distribution shape parameters alpha and beta (" [static, private]`

Definition at line 207 of file BayesianModelChecker.hpp.

Referenced by validateShapeParameters().

6.13.4.8 `const std::string BayesianModelChecker::ERR_SHAPE_PARAMETERS_END = ") should be greater than zero. Please change." [static, private]`

Definition at line 209 of file BayesianModelChecker.hpp.

Referenced by validateShapeParameters().

6.13.4.9 `const std::string BayesianModelChecker::ERR_SHAPE_PARAMETERS_MIDDLE = ", " [static, private]`

Definition at line 208 of file BayesianModelChecker.hpp.

Referenced by validateShapeParameters().

6.13.4.10 `const std::string BayesianModelChecker::ERR_UNEXPECTED_MODEL_CHECKING_RESULT = "An invalid Bayesian model checking result was obtained. Please check source code." [static, private]`

Definition at line 205 of file BayesianModelChecker.hpp.

Referenced by doesPropertyHoldConsideringResult().

6.13.4.11 `BayesianModelCheckingResult multiscale::verification::BayesianModelChecker::modelCheckingResult [private]`

The result of the model checking task

Definition at line 66 of file BayesianModelChecker.hpp.

Referenced by acceptsMoreTraces(), doesPropertyHoldConsideringResult(), getDetailedUpdatedResults(), updateModelCheckingResultEnoughTraces(), and updateModelCheckingResultNotEnoughTraces().

6.13.4.12 `const std::string BayesianModelChecker::MSG_OUTPUT_MORE_TRACES_REQUIRED = "More traces are required to provide a true/false answer assuming the given Beta distribution shape parameters and Bayes factor threshold value. Probabilistic black-box model checking was used instead to provide an answer."`  
[static, private]

Definition at line 214 of file BayesianModelChecker.hpp.

Referenced by getDetailedUpdatedResults().

6.13.4.13 `const std::string BayesianModelChecker::MSG_OUTPUT_RESULT_BEGIN = "The provided answer is given for the Beta distribution shape parameters alpha = "`  
[static, private]

Definition at line 216 of file BayesianModelChecker.hpp.

Referenced by getDetailedUpdatedResults().

6.13.4.14 `const std::string BayesianModelChecker::MSG_OUTPUT_RESULT_END = ""`  
[static, private]

Definition at line 220 of file BayesianModelChecker.hpp.

Referenced by getDetailedUpdatedResults().

6.13.4.15 `const std::string BayesianModelChecker::MSG_OUTPUT_RESULT_MIDDLE1 = " and beta = "`  
[static, private]

Definition at line 217 of file BayesianModelChecker.hpp.

Referenced by getDetailedUpdatedResults().

6.13.4.16 `const std::string BayesianModelChecker::MSG_OUTPUT_RESULT_MIDDLE2 = ", and Bayes factor threshold value = "`  
[static, private]

Definition at line 218 of file BayesianModelChecker.hpp.

Referenced by getDetailedUpdatedResults().

## **6.14 multiscale::verification::BayesianModelCheckerFactory Class Reference**

---

**6.13.4.17 const std::string BayesianModelChecker::MSG\_OUTPUT\_RESULT\_MIDDLE3 = ”. The type I error upper bound for the provided answer is =” [static, private]**

Definition at line 219 of file BayesianModelChecker.hpp.

Referenced by getDetailedUpdatedResults().

**6.13.4.18 const std::string BayesianModelChecker::MSG\_OUTPUT\_SEPARATOR = ” [static, private]**

Definition at line 222 of file BayesianModelChecker.hpp.

Referenced by getDetailedUpdatedResults().

**6.13.4.19 double multiscale::verification::BayesianModelChecker::probability [private]**

The probability specified by the user for the logic property to be evaluated

Definition at line 54 of file BayesianModelChecker.hpp.

Referenced by computeBayesFactorValue(), computeMaximumBinomialPDF(), and initialise().

**6.13.4.20 double multiscale::verification::BayesianModelChecker::typeIErrorUpperBound [private]**

The type I error upper bound

Definition at line 63 of file BayesianModelChecker.hpp.

Referenced by getDetailedUpdatedResults(), initialise(), and updateTypeIErrorUpperBound().

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

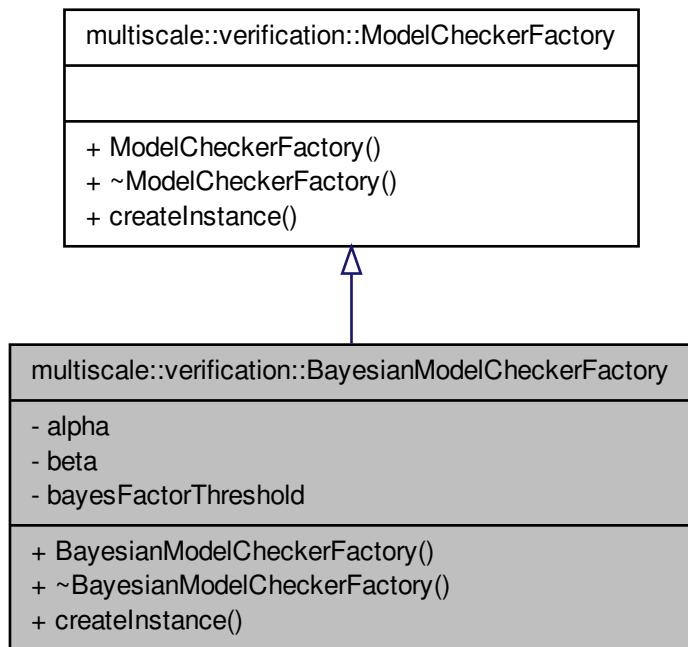
- BayesianModelChecker.hpp
- BayesianModelChecker.cpp

## **6.14 multiscale::verification::BayesianModelCheckerFactory Class Reference**

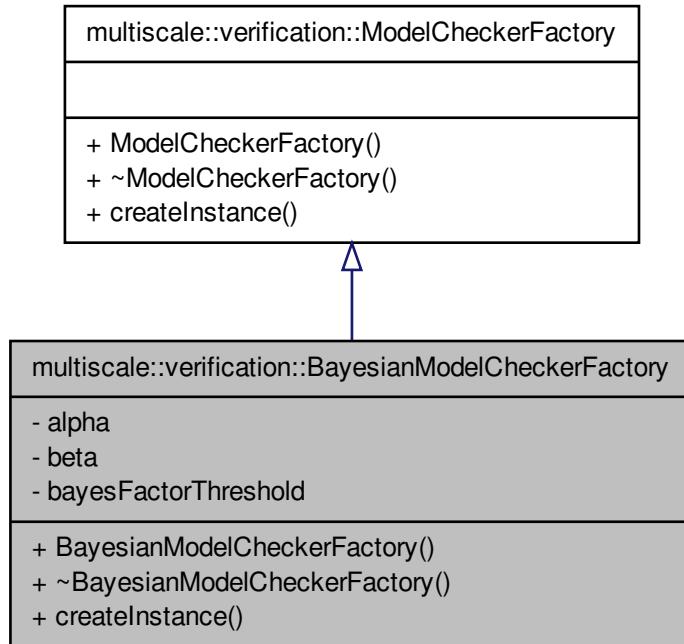
Class for creating [BayesianModelChecker](#) instances.

```
#include <BayesianModelCheckerFactory.hpp>
```

Inheritance diagram for multiscale::verification::BayesianModelCheckerFactory:



Collaboration diagram for multiscale::verification::BayesianModelCheckerFactory:



## Public Member Functions

- `BayesianModelCheckerFactory (double alpha, double beta, double bayesFactorThreshold)`
- `~BayesianModelCheckerFactory ()`
- `std::shared_ptr< ModelChecker > createInstance (const AbstractSyntaxTree &abstractSyntaxTree, const MultiscaleArchitectureGraph &multiscaleArchitectureGraph) override`

*Create an instance of `BayesianModelChecker`.*

## Private Attributes

- `double alpha`
- `double beta`
- `double bayesFactorThreshold`

### 6.14.1 Detailed Description

Class for creating [BayesianModelChecker](#) instances.

Definition at line 12 of file BayesianModelCheckerFactory.hpp.

### 6.14.2 Constructor & Destructor Documentation

#### 6.14.2.1 BayesianModelCheckerFactory::BayesianModelCheckerFactory ( double *alpha*, double *beta*, double *bayesFactorThreshold* )

Definition at line 7 of file BayesianModelCheckerFactory.cpp.

#### 6.14.2.2 BayesianModelCheckerFactory::~BayesianModelCheckerFactory ( )

Definition at line 12 of file BayesianModelCheckerFactory.cpp.

### 6.14.3 Member Function Documentation

#### 6.14.3.1 std::shared\_ptr< ModelChecker > BayesianModelCheckerFactory- ::createInstance ( const AbstractSyntaxTree & *abstractSyntaxTree*, const MultiscaleArchitectureGraph & *multiscaleArchitectureGraph* ) [override, virtual]

Create an instance of [BayesianModelChecker](#).

#### Parameters

<i>abstract-SyntaxTree</i>	The abstract syntax tree representing the logic property to be checked
<i>multiscale-Architecture-Graph</i>	The multiscale architecture graph encoding the hierarchical organization of scales and subsystems

Implements [multiscale::verification::ModelCheckerFactory](#).

Definition at line 15 of file BayesianModelCheckerFactory.cpp.

References alpha, bayesFactorThreshold, and beta.

### 6.14.4 Member Data Documentation

#### 6.14.4.1 double multiscale::verification::BayesianModelCheckerFactory::alpha [private]

The shape parameter  $\alpha$  for the Beta distribution prior

Definition at line 16 of file BayesianModelCheckerFactory.hpp.

Referenced by `createInstance()`.

#### 6.14.4.2 double multiscale::verification::BayesianModelCheckerFactory::bayesFactorThreshold [private]

The Bayes factor threshold

Definition at line 19 of file `BayesianModelCheckerFactory.hpp`.

Referenced by `createInstance()`.

#### 6.14.4.3 double multiscale::verification::BayesianModelCheckerFactory::beta [private]

The shape parameter  $\beta$  for the Beta distribution prior

Definition at line 17 of file `BayesianModelCheckerFactory.hpp`.

Referenced by `createInstance()`.

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

- `BayesianModelCheckerFactory.hpp`
- `BayesianModelCheckerFactory.cpp`

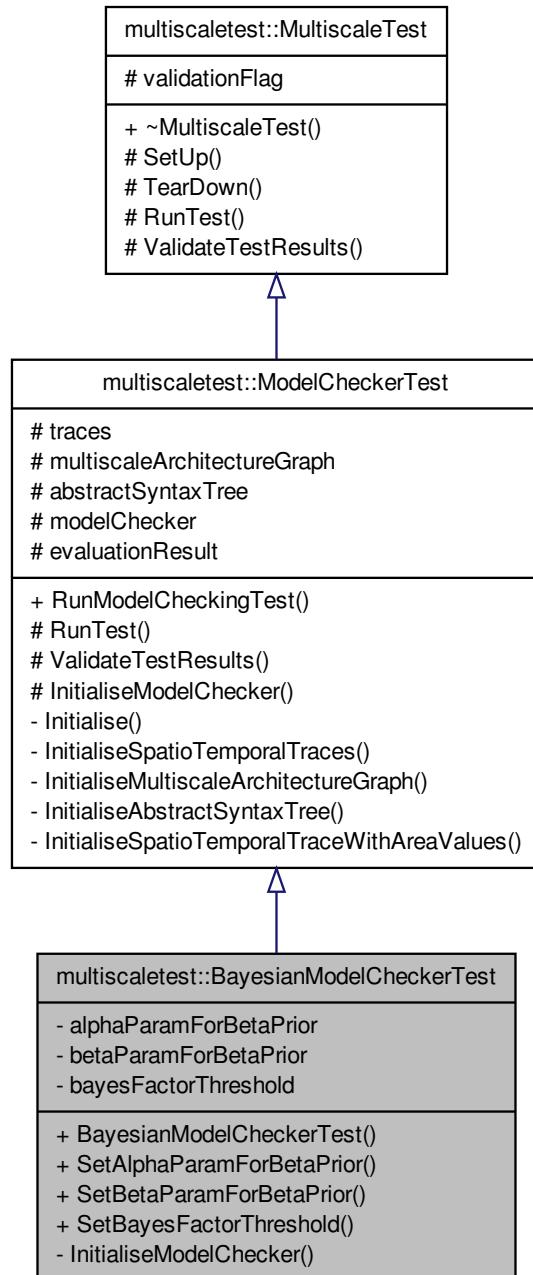
## 6.15 multiscaletest::BayesianModelCheckerTest Class Reference

Class for testing the Bayesian model checker.

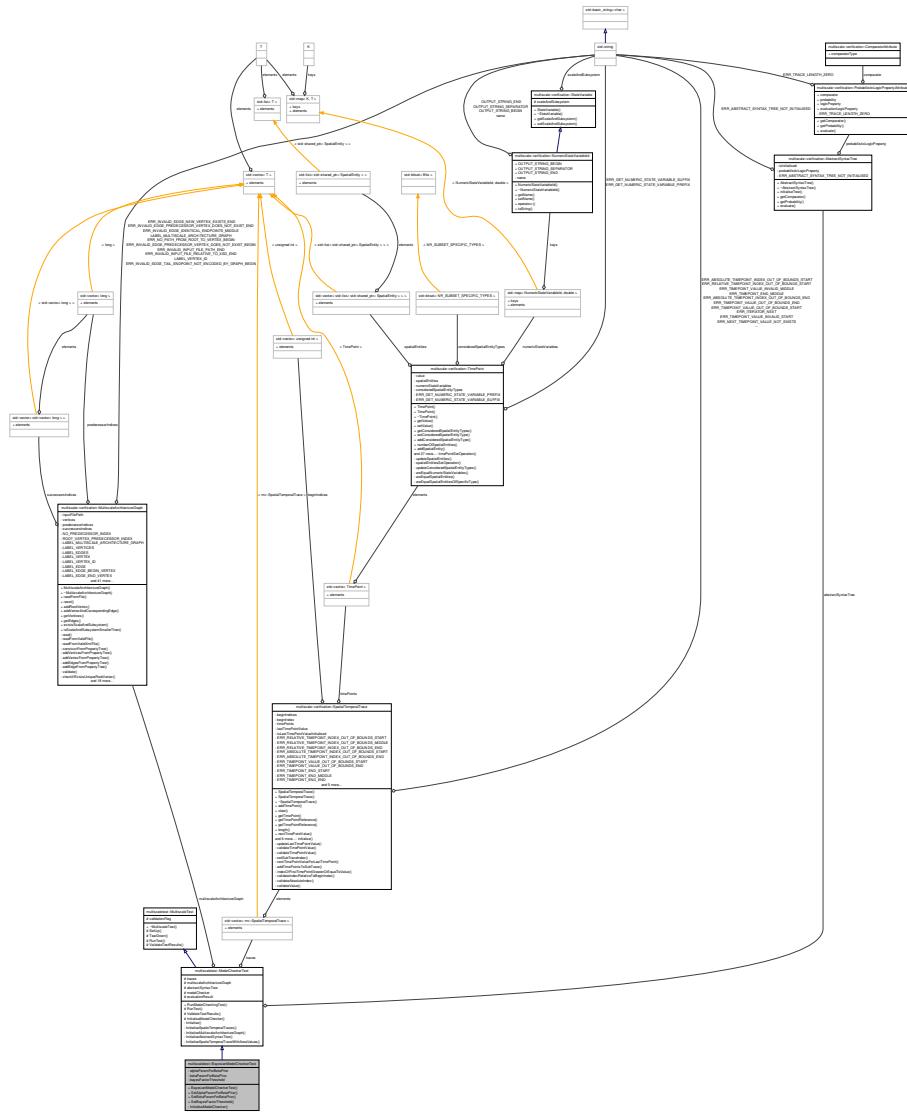
```
#include <BayesianModelCheckerTest.hpp>
```

---

Inheritance diagram for multiscaletest::BayesianModelCheckerTest:



## Collaboration diagram for multiscaletest::BayesianModelCheckerTest:



## Public Member Functions

- `BayesianModelCheckerTest()`
  - void `SetAlphaParamForBetaPrior` (double `alphaParamForBetaPrior`)  
*Set the value of the alpha parameter for the beta prior.*
  - void `SetBetaParamForBetaPrior` (double `betaParamForBetaPrior`)  
*Set the value of the beta parameter for the beta prior.*
  - void `SetBayesFactorThreshold` (double `bayesFactorThreshold`)  
*Set the value of the Bayes factor threshold.*

## Private Member Functions

- void [InitialiseModelChecker \(\) override](#)

*Initialise the model checker.*

## Private Attributes

- double [alphaParamForBetaPrior](#)
- double [betaParamForBetaPrior](#)
- double [bayesFactorThreshold](#)

### 6.15.1 Detailed Description

Class for testing the Bayesian model checker.

Definition at line 15 of file BayesianModelCheckerTest.hpp.

### 6.15.2 Constructor & Destructor Documentation

#### 6.15.2.1 [multiscaletest::BayesianModelCheckerTest::BayesianModelCheckerTest \( \) \[inline\]](#)

Definition at line 26 of file BayesianModelCheckerTest.hpp.

### 6.15.3 Member Function Documentation

#### 6.15.3.1 [void multiscaletest::BayesianModelCheckerTest::InitialiseModelChecker \( \) \[override, private, virtual\]](#)

Initialise the model checker.

Implements [multiscaletest::ModelCheckerTest](#).

Definition at line 68 of file BayesianModelCheckerTest.hpp.

#### 6.15.3.2 [void multiscaletest::BayesianModelCheckerTest::SetAlphaParamForBetaPrior \( double alphaParamForBetaPrior \)](#)

Set the value of the alpha parameter for the beta prior.

##### Parameters

<i>alphaParam- ForBetaPrior</i>	The alpha parameter for the beta prior
-------------------------------------	--

Definition at line 56 of file BayesianModelCheckerTest.hpp.

```
6.15.3.3 void multiscaletest::BayesianModelCheckerTest::Set-
    BayesFactorThreshold ( double bayesFactorThreshold
    )
```

Set the value of the Bayes factor threshold.

Parameters

<i>bayesFactor-</i> <i>Threshold</i>	The value of the Bayes factor threshold
---	---

Definition at line 64 of file BayesianModelCheckerTest.hpp.

```
6.15.3.4 void multiscaletest::BayesianModelCheckerTest::Set-
    BetaParamForBetaPrior ( double betaParamForBetaPrior
    )
```

Set the value of the beta parameter for the beta prior.

Parameters

<i>betaParam-</i> <i>ForBetaPrior</i>	The beta parameter for the beta prior
--	---------------------------------------

Definition at line 60 of file BayesianModelCheckerTest.hpp.

## 6.15.4 Member Data Documentation

```
6.15.4.1 double multiscaletest::BayesianModelCheckerTest::alphaParamForBeta-
    Prior [private]
```

The alpha parameter for the beta prior

Definition at line 19 of file BayesianModelCheckerTest.hpp.

```
6.15.4.2 double multiscaletest::BayesianModelCheckerTest::bayesFactor-
    Threshold [private]
```

The considered bayes factor threshold

Definition at line 22 of file BayesianModelCheckerTest.hpp.

```
6.15.4.3 double multiscaletest::BayesianModelCheckerTest::betaParamForBeta-
    Prior [private]
```

The beta parameter for the beta prior

Definition at line 20 of file BayesianModelCheckerTest.hpp.

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

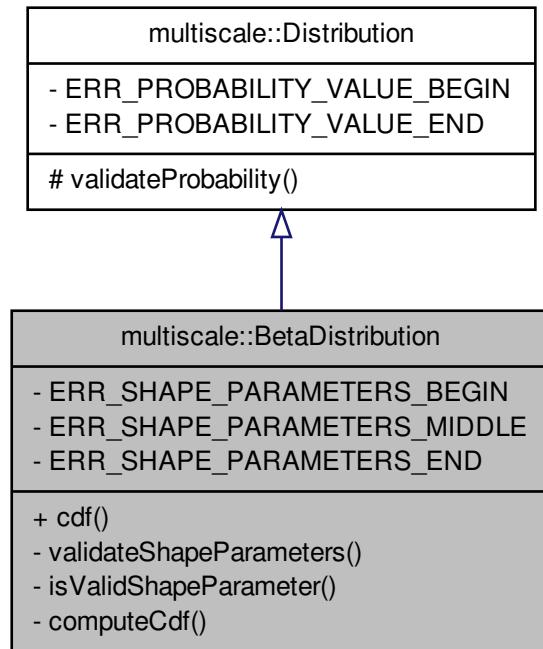
- BayesianModelCheckerTest.hpp

## 6.16 multiscale::BetaDistribution Class Reference

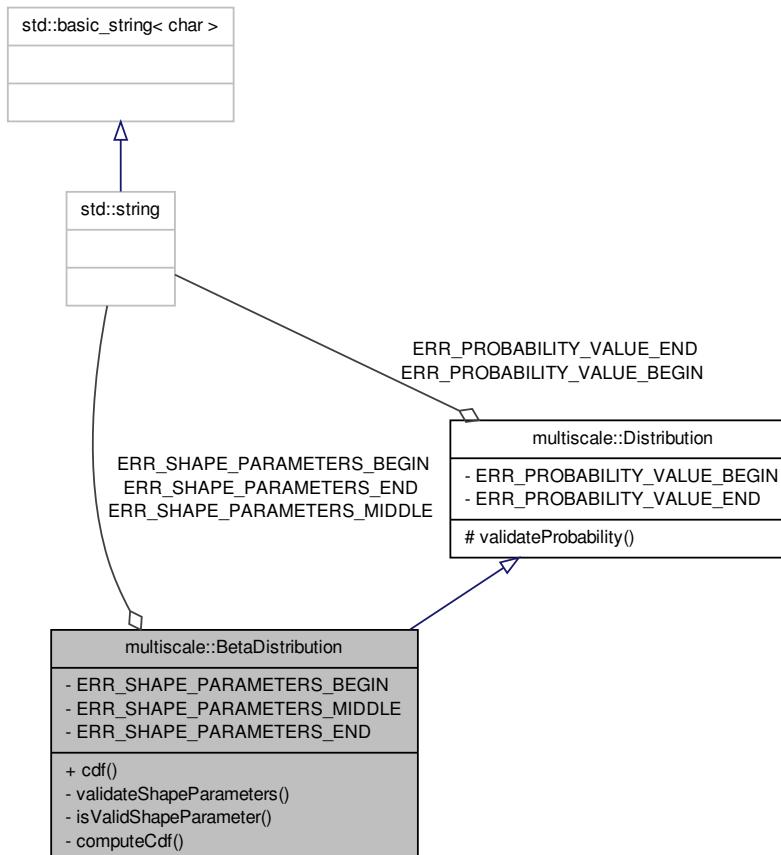
Class for analysing Beta distributed data.

```
#include <BetaDistribution.hpp>
```

Inheritance diagram for multiscale::BetaDistribution:



Collaboration diagram for multiscale::BetaDistribution:



### Static Public Member Functions

- static double [cdf](#) (double alpha, double beta, double probability)  
*Compute the value of the cumulative distribution function (cdf) for a Beta distribution.*

### Static Private Member Functions

- static void [validateShapeParameters](#) (double alpha, double beta)  
*Validate the shape parameters  $\alpha$  and  $\beta$ .*
- static bool [isValidShapeParameter](#) (double shapeParameter)  
*Check if the given shape parameter value is valid.*
- static double [computeCdf](#) (double alpha, double beta, double probability)

*Compute the value of the cumulative distribution function (cdf) for a Beta distribution considering that the parameters are valid.*

### Static Private Attributes

- static const std::string `ERR_SHAPE_PARAMETERS_BEGIN` = "The provided - Beta distribution shape parameters alpha and beta ("
- static const std::string `ERR_SHAPE_PARAMETERS_MIDDLE` = ", "
- static const std::string `ERR_SHAPE_PARAMETERS_END` = ") should be greater than zero. Please change."

#### 6.16.1 Detailed Description

Class for analysing Beta distributed data.

Definition at line 10 of file `BetaDistribution.hpp`.

#### 6.16.2 Member Function Documentation

##### 6.16.2.1 double `BetaDistribution::cdf` ( double *alpha*, double *beta*, double *probability* ) [static]

Compute the value of the cumulative distribution function (cdf) for a Beta distribution.

The value of the cumulative distribution function (cdf) is computed considering the given probability and shape parameters.

#### Parameters

<i>alpha</i>	Shape parameter <i>alpha</i>
<i>beta</i>	Shape parameter <i>beta</i>
<i>probability</i>	The considered probability when computing the value of the cdf

Definition at line 10 of file `BetaDistribution.cpp`.

References `computeCdf()`, `multiscale::Distribution::validateProbability()`, and `validateShapeParameters()`.

Referenced by `multiscale::verification::BayesianModelChecker::computeBayesFactorValue()`, and `computeCdf()`.

##### 6.16.2.2 double `BetaDistribution::computeCdf` ( double *alpha*, double *beta*, double *probability* ) [static, private]

Compute the value of the cumulative distribution function (cdf) for a Beta distribution considering that the parameters are valid.

**Parameters**

<i>alpha</i>	Shape parameter <i>alpha</i>
<i>beta</i>	Shape parameter <i>beta</i>
<i>probability</i>	The considered probability when computing the value of the cdf

Definition at line 32 of file BetaDistribution.cpp.

References [cdf\(\)](#).

Referenced by [cdf\(\)](#).

**6.16.2.3 bool BetaDistribution::isValidShapeParameter ( double *shapeParameter* ) [static, private]**

Check if the given shape parameter value is valid.

The shape parameter values should be greater than zero

**Parameters**

<i>shape-Parameter</i>	The given shape parameter
------------------------	---------------------------

Definition at line 28 of file BetaDistribution.cpp.

Referenced by [validateShapeParameters\(\)](#).

**6.16.2.4 void BetaDistribution::validateShapeParameters ( double *alpha*, double *beta* ) [static, private]**

Validate the shape parameters  $\alpha$  and  $\beta$ .

$\alpha$  and  $\beta$  should be greater than zero

**Parameters**

<i>alpha</i>	The shape parameter $\alpha$ for the Beta distribution
<i>beta</i>	The shape parameter $\beta$ for the Beta distribution

Definition at line 17 of file BetaDistribution.cpp.

References [ERR\\_SHAPE\\_PARAMETERS\\_BEGIN](#), [ERR\\_SHAPE\\_PARAMETER\\_S\\_END](#), [ERR\\_SHAPE\\_PARAMETERS\\_MIDDLE](#), [isValidShapeParameter\(\)](#), and [multiscale::StringManipulator::toString\(\)](#).

Referenced by [cdf\(\)](#).

### 6.16.3 Member Data Documentation

```
6.16.3.1 const std::string BetaDistribution::ERR_SHAPE_PARAMETERS_BEGIN =
    "The provided Beta distribution shape parameters alpha and beta (" [static,
    private]
```

Definition at line 51 of file BetaDistribution.hpp.

Referenced by validateShapeParameters().

```
6.16.3.2 const std::string BetaDistribution::ERR_SHAPE_PARAMETERS_END = ")
    should be greater than zero. Please change." [static, private]
```

Definition at line 53 of file BetaDistribution.hpp.

Referenced by validateShapeParameters().

```
6.16.3.3 const std::string BetaDistribution::ERR_SHAPE_PARAMETERS_MIDDLE =
    "," [static, private]
```

Definition at line 52 of file BetaDistribution.hpp.

Referenced by validateShapeParameters().

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

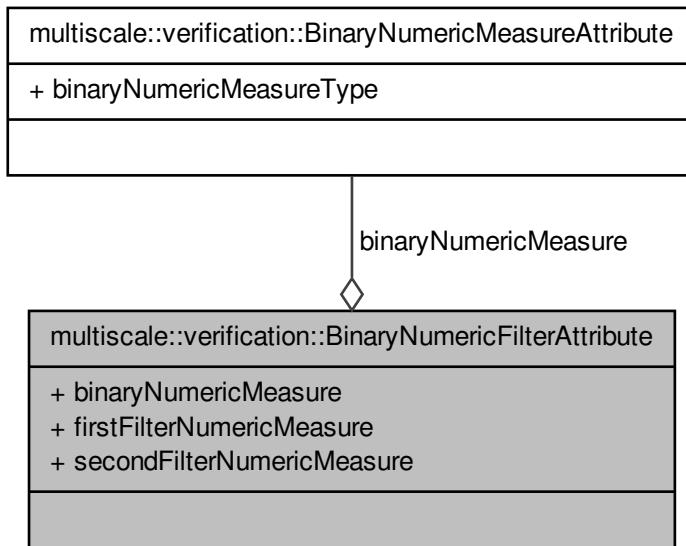
- BetaDistribution.hpp
- BetaDistribution.cpp

## 6.17 multiscale::verification::BinaryNumericFilterAttribute Class - Reference

Class for representing a binary numeric filter attribute.

```
#include <BinaryNumericFilterAttribute.hpp>
```

Collaboration diagram for multiscale::verification::BinaryNumericFilterAttribute:



## Public Attributes

- [BinaryNumericMeasureAttribute](#) `binaryNumericMeasure`
- [FilterNumericMeasureAttributeType](#) `firstFilterNumericMeasure`
- [FilterNumericMeasureAttributeType](#) `secondFilterNumericMeasure`

### 6.17.1 Detailed Description

Class for representing a binary numeric filter attribute.

Definition at line 15 of file `BinaryNumericFilterAttribute.hpp`.

### 6.17.2 Member Data Documentation

#### 6.17.2.1 [BinaryNumericMeasureAttribute](#) `multiscale::verification::Binary-NumericFilterAttribute::binaryNumericMeasure`

The binary numeric measure

Definition at line 20 of file `BinaryNumericFilterAttribute.hpp`.

Referenced by multiscale::verification::FilterNumericVisitor::operator()().

#### 6.17.2.2 FilterNumericMeasureAttributeType multiscale::verification::Binary-NumericFilterAttribute::firstFilterNumericMeasure

The first filter numeric measure

Definition at line 22 of file BinaryNumericFilterAttribute.hpp.

Referenced by multiscale::verification::FilterNumericVisitor::operator()().

#### 6.17.2.3 FilterNumericMeasureAttributeType multiscale::verification::Binary-NumericFilterAttribute::secondFilterNumericMeasure

The second filter numeric measure

Definition at line 24 of file BinaryNumericFilterAttribute.hpp.

Referenced by multiscale::verification::FilterNumericVisitor::operator()().

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

- BinaryNumericFilterAttribute.hpp

## 6.18 multiscale::verification::BinaryNumericMeasureAttribute - Class Reference

Class for representing a binary numeric measure attribute.

```
#include <BinaryNumericMeasureAttribute.hpp>
```

### Public Attributes

- [BinaryNumericMeasureType binaryNumericMeasureType](#)

#### 6.18.1 Detailed Description

Class for representing a binary numeric measure attribute.

Definition at line 33 of file BinaryNumericMeasureAttribute.hpp.

#### 6.18.2 Member Data Documentation

##### 6.18.2.1 BinaryNumericMeasureType multiscale::verification::BinaryNumeric-MeasureAttribute::binaryNumericMeasureType

The binary numeric measure type

Definition at line 37 of file BinaryNumericMeasureAttribute.hpp.

Referenced by multiscale::verification::SpatialMeasureCollectionVisitor::evaluateBinaryNumericSpatialMeasureCollection(), multiscale::verification::FilterNumericVisitor::operator()(), multiscale::verification::TemporalNumericVisitor::operator()(), and multiscale::verification::NumericVisitor::operator()().

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

- BinaryNumericMeasureAttribute.hpp

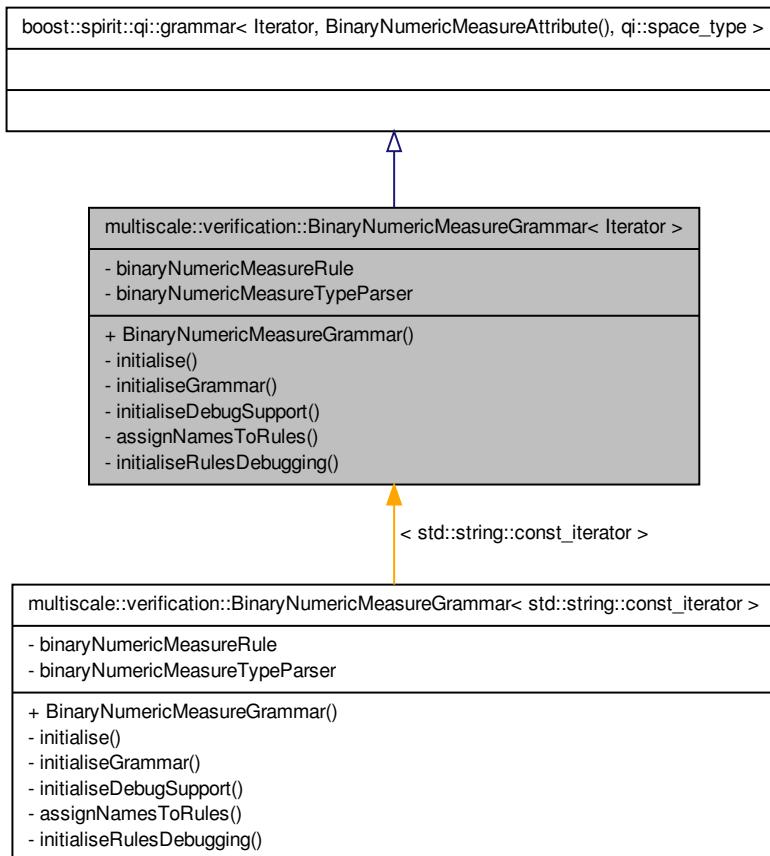
**6.19 multiscale::verification::BinaryNumericMeasureGrammar<  
Iterator > Class Template Reference**

The grammar for parsing binary numeric measure statements.

```
#include <BinaryNumericMeasureGrammar.hpp>
```

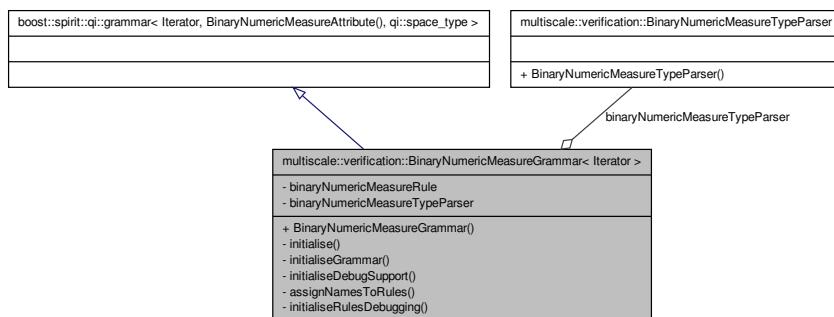
```
Inheritance diagram for multiscale::verification::BinaryNumericMeasureGrammar<
```

Iterator >:



Collaboration diagram for `multiscale::verification::BinaryNumericMeasureGrammar< ->`

Iterator >:



## Public Member Functions

- [BinaryNumericMeasureGrammar \(\)](#)

## Private Member Functions

- void [initialise \(\)](#)  
*Initialisation function.*
- void [initialiseGrammar \(\)](#)  
*Initialise the grammar.*
- void [initialiseDebugSupport \(\)](#)  
*Initialise debug support.*
- void [assignNamesToRules \(\)](#)  
*Assign names to the rules.*
- void [initialiseRulesDebugging \(\)](#)  
*Initialise the debugging of rules.*

## Private Attributes

- [qi::rule< Iterator, BinaryNumericMeasureAttribute\(\), qi::space\\_type > binaryNumericMeasureRule](#)
- [BinaryNumericMeasureTypeParser binaryNumericMeasureTypeParser](#)

### 6.19.1 Detailed Description

`template<typename Iterator>class multiscale::verification::BinaryNumericMeasureGrammar< Iterator >`

The grammar for parsing binary numeric measure statements.

Definition at line 24 of file BinaryNumericMeasureGrammar.hpp.

### 6.19.2 Constructor & Destructor Documentation

6.19.2.1 **template<typename Iterator > multiscale::verification::BinaryNumericMeasureGrammar< Iterator >::BinaryNumericMeasureGrammar ( )**

Definition at line 17 of file BinaryNumericMeasureGrammarDefinition.hpp.

References multiscale::verification::BinaryNumericMeasureGrammar< Iterator >::initialise().

### 6.19.3 Member Function Documentation

6.19.3.1 **template<typename Iterator > void multiscale::verification::BinaryNumericMeasureGrammar< Iterator >::assignNamesToRules ( ) [private]**

Assign names to the rules.

Definition at line 50 of file BinaryNumericMeasureGrammarDefinition.hpp.

6.19.3.2 **template<typename Iterator > void multiscale::verification::BinaryNumericMeasureGrammar< Iterator >::initialise ( ) [private]**

Initialisation function.

Definition at line 27 of file BinaryNumericMeasureGrammarDefinition.hpp.

Referenced by multiscale::verification::BinaryNumericMeasureGrammar< Iterator >::BinaryNumericMeasureGrammar().

6.19.3.3 **template<typename Iterator > void multiscale::verification::BinaryNumericMeasureGrammar< Iterator >::initialiseDebugSupport ( ) [private]**

Initialise debug support.

Definition at line 41 of file BinaryNumericMeasureGrammarDefinition.hpp.

6.19.3.4 **template<typename Iterator > void multiscale::verification::BinaryNumericMeasureGrammar< Iterator >::initialiseGrammar ( ) [private]**

Initialise the grammar.

Definition at line 34 of file BinaryNumericMeasureGrammarDefinition.hpp.

```
6.19.3.5 template<typename Iterator> void multiscale::verification::Binary-
    NumericMeasureGrammar< Iterator >::initialiseRulesDebugging( )
        [private]
```

Initialise the debugging of rules.

Definition at line 56 of file BinaryNumericMeasureGrammarDefinition.hpp.

#### 6.19.4 Member Data Documentation

```
6.19.4.1 template<typename Iterator> qi::rule<Iterator,
    BinaryNumericMeasureAttribute(), qi::space_type>
    multiscale::verification::BinaryNumericMeasureGrammar< Iterator
    >::binaryNumericMeasureRule [private]
```

The rule for parsing a binary numeric measure

Definition at line 30 of file BinaryNumericMeasureGrammar.hpp.

```
6.19.4.2 template<typename Iterator> BinaryNumericMeasureTypeParser
    multiscale::verification::BinaryNumericMeasureGrammar< Iterator
    >::binaryNumericMeasureTypeParser [private]
```

The binary numeric measure type parser

Definition at line 35 of file BinaryNumericMeasureGrammar.hpp.

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

- BinaryNumericMeasureGrammar.hpp
- BinaryNumericMeasureGrammarDefinition.hpp

## 6.20 multiscale::verification::BinaryNumericMeasureTypeParser Struct Reference

Symbol table and parser for the binary numeric measure type.

```
#include <SymbolTables.hpp>
```

### Public Member Functions

- [BinaryNumericMeasureTypeParser \(\)](#)

#### 6.20.1 Detailed Description

Symbol table and parser for the binary numeric measure type.

Definition at line 27 of file SymbolTables.hpp.

## 6.20.2 Constructor & Destructor Documentation

### 6.20.2.1 multiscale::verification::BinaryNumericMeasureTypeParser::BinaryNumericMeasureTypeParser ( ) [inline]

Definition at line 30 of file SymbolTables.hpp.

References multiscale::verification::Div, multiscale::verification::Mod, and multiscale::verification::Power.

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

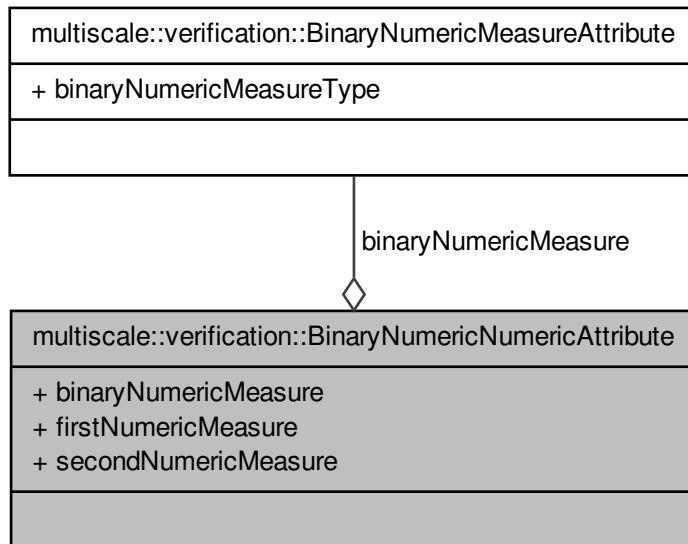
- SymbolTables.hpp

## 6.21 multiscale::verification::BinaryNumericNumericAttribute - Class Reference

Class for representing a binary numeric numeric measure attribute.

```
#include <BinaryNumericNumericAttribute.hpp>
```

Collaboration diagram for multiscale::verification::BinaryNumericNumericAttribute:



## Public Attributes

- [BinaryNumericMeasureAttribute binaryNumericMeasure](#)
- [NumericMeasureType firstNumericMeasure](#)
- [NumericMeasureType secondNumericMeasure](#)

### 6.21.1 Detailed Description

Class for representing a binary numeric numeric measure attribute.

Definition at line 15 of file [BinaryNumericNumericAttribute.hpp](#).

### 6.21.2 Member Data Documentation

#### 6.21.2.1 BinaryNumericMeasureAttribute multiscale::verification::Binary-NumericNumericAttribute::binaryNumericMeasure

The binary numeric measure

Definition at line 19 of file [BinaryNumericNumericAttribute.hpp](#).

Referenced by multiscale::verification::NumericVisitor::operator()().

#### 6.21.2.2 NumericMeasureType multiscale::verification::BinaryNumericNumericAttribute::firstNumericMeasure

The first numeric measure

Definition at line 20 of file BinaryNumericNumericAttribute.hpp.

Referenced by multiscale::verification::NumericVisitor::operator()().

#### 6.21.2.3 NumericMeasureType multiscale::verification::BinaryNumericNumericAttribute::secondNumericMeasure

The second numeric measure

Definition at line 21 of file BinaryNumericNumericAttribute.hpp.

Referenced by multiscale::verification::NumericVisitor::operator()().

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

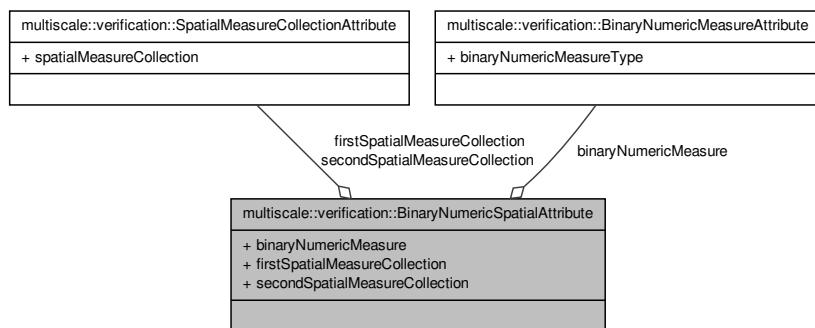
- BinaryNumericNumericAttribute.hpp

## 6.22 multiscale::verification::BinaryNumericSpatialAttribute Class Reference

Class for representing a binary numeric spatial measure collection attribute.

```
#include <BinaryNumericSpatialAttribute.hpp>
```

Collaboration diagram for multiscale::verification::BinaryNumericSpatialAttribute:



## Public Attributes

- `BinaryNumericMeasureAttribute binaryNumericMeasure`
- `SpatialMeasureCollectionAttribute firstSpatialMeasureCollection`
- `SpatialMeasureCollectionAttribute secondSpatialMeasureCollection`

### 6.22.1 Detailed Description

Class for representing a binary numeric spatial measure collection attribute.

Definition at line 15 of file `BinaryNumericSpatialAttribute.hpp`.

### 6.22.2 Member Data Documentation

#### 6.22.2.1 `BinaryNumericMeasureAttribute multiscale::verification::Binary-NumericSpatialAttribute::binaryNumericMeasure`

The binary numeric measure

Definition at line 19 of file `BinaryNumericSpatialAttribute.hpp`.

Referenced by `multiscale::verification::SpatialMeasureCollectionVisitor::operator()()`.

#### 6.22.2.2 `SpatialMeasureCollectionAttribute multiscale::verification::Binary-NumericSpatialAttribute::firstSpatialMeasureCollection`

The first considered spatial measure collection

Definition at line 20 of file `BinaryNumericSpatialAttribute.hpp`.

Referenced by `multiscale::verification::SpatialMeasureCollectionVisitor::operator()()`.

#### 6.22.2.3 `SpatialMeasureCollectionAttribute multiscale::verification::Binary-NumericSpatialAttribute::secondSpatialMeasureCollection`

The second considered spatial measure collection

Definition at line 22 of file `BinaryNumericSpatialAttribute.hpp`.

Referenced by `multiscale::verification::SpatialMeasureCollectionVisitor::operator()()`.

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

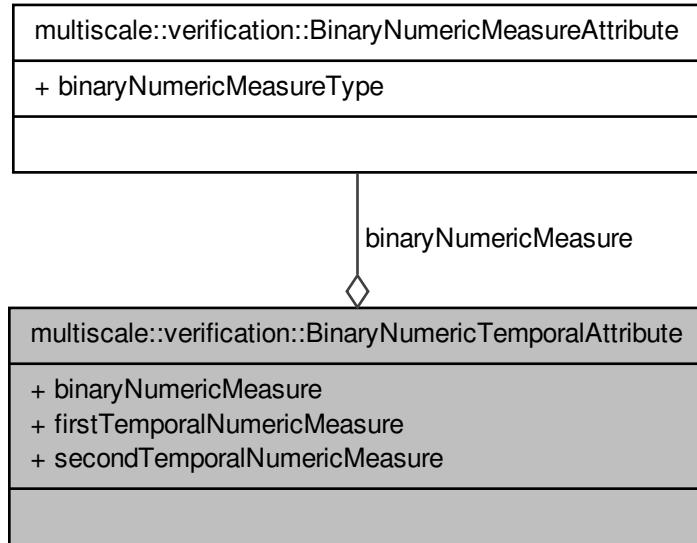
- `BinaryNumericSpatialAttribute.hpp`

## 6.23 `multiscale::verification::BinaryNumericTemporalAttribute` - Class Reference

Class for representing a binary numeric temporal measure attribute.

```
#include <BinaryNumericTemporalAttribute.hpp>
```

Collaboration diagram for multiscale::verification::BinaryNumericTemporalAttribute:



## Public Attributes

- `BinaryNumericMeasureAttribute binaryNumericMeasure`
- `TemporalNumericMeasureType firstTemporalNumericMeasure`
- `TemporalNumericMeasureType secondTemporalNumericMeasure`

### 6.23.1 Detailed Description

Class for representing a binary numeric temporal measure attribute.

Definition at line 15 of file `BinaryNumericTemporalAttribute.hpp`.

### 6.23.2 Member Data Documentation

#### 6.23.2.1 `BinaryNumericMeasureAttribute multiscale::verification::BinaryNumericTemporalAttribute::binaryNumericMeasure`

The binary numeric measure

## **6.24 multiscale::verification::BinaryStatisticalMeasureAttribute Class Reference**

Definition at line 20 of file BinaryNumericTemporalAttribute.hpp.

Referenced by multiscale::verification::TemporalNumericVisitor::operator()().

### **6.23.2.2 TemporalNumericMeasureType multiscale::verification::BinaryNumericTemporalAttribute::firstTemporalNumericMeasure**

The first temporal numeric measure

Definition at line 22 of file BinaryNumericTemporalAttribute.hpp.

Referenced by multiscale::verification::TemporalNumericVisitor::operator()().

### **6.23.2.3 TemporalNumericMeasureType multiscale::verification::BinaryNumericTemporalAttribute::secondTemporalNumericMeasure**

The second temporal numeric measure

Definition at line 24 of file BinaryNumericTemporalAttribute.hpp.

Referenced by multiscale::verification::TemporalNumericVisitor::operator()().

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

- BinaryNumericTemporalAttribute.hpp

## **6.24 multiscale::verification::BinaryStatisticalMeasureAttribute - Class Reference**

Class for representing a binary statistical measure attribute.

```
#include <BinaryStatisticalMeasureAttribute.hpp>
```

### **Public Attributes**

- [BinaryStatisticalMeasureType binaryStatisticalMeasureType](#)

#### **6.24.1 Detailed Description**

Class for representing a binary statistical measure attribute.

Definition at line 28 of file BinaryStatisticalMeasureAttribute.hpp.

#### **6.24.2 Member Data Documentation**

### 6.24.2.1 **BinaryStatisticalMeasureType multiscale::verification::BinaryStatisticalMeasureAttribute::binaryStatisticalMeasureType**

The binary statistical measure type

Definition at line 32 of file `BinaryStatisticalMeasureAttribute.hpp`.

Referenced by `multiscale::verification::NumericVisitor::operator()()`, and `multiscale::verification::TemporalNumericVisitor::operator()()`.

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

- `BinaryStatisticalMeasureAttribute.hpp`

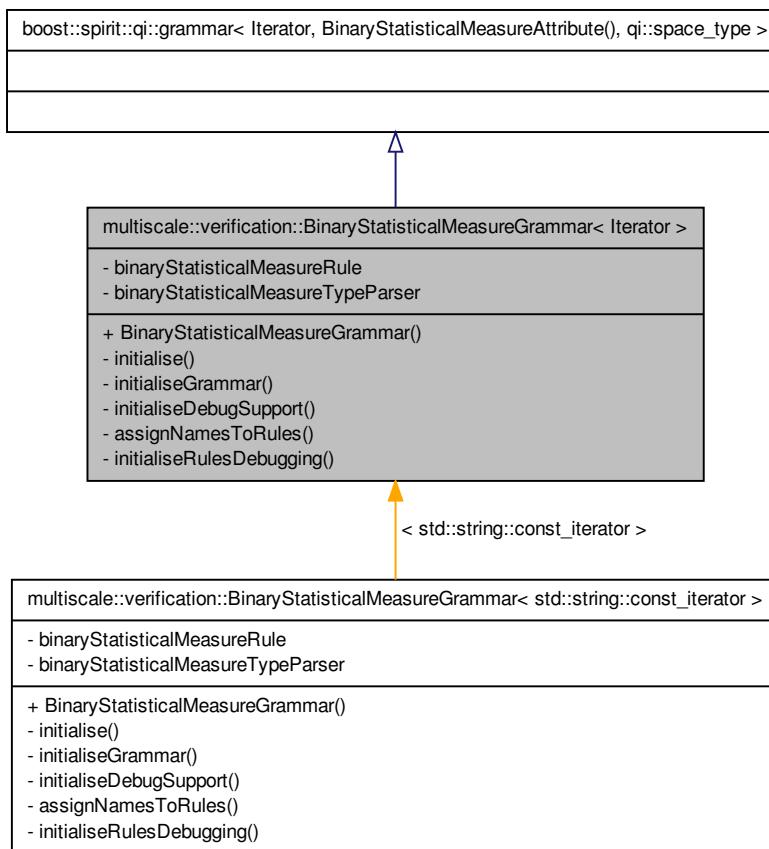
## 6.25 **multiscale::verification::BinaryStatisticalMeasureGrammar< Iterator >** Class Template Reference

The grammar for parsing binary statistical measure statements.

```
#include <BinaryStatisticalMeasureGrammar.hpp>
```

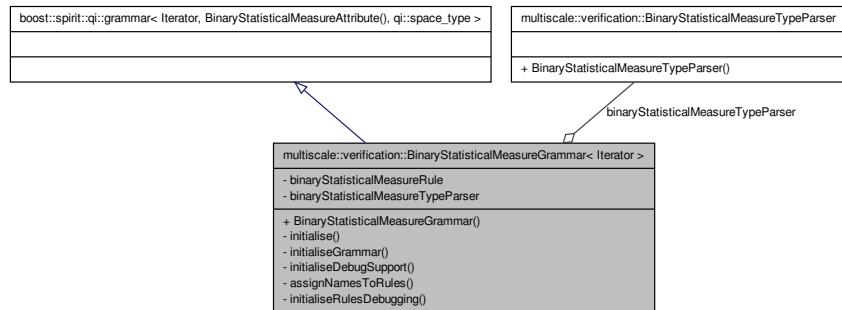
Inheritance diagram for `multiscale::verification::BinaryStatisticalMeasureGrammar< -`

Iterator >:



Collaboration diagram for `multiscale::verification::BinaryStatisticalMeasureGrammar< ->`

Iterator >:



## Public Member Functions

- [BinaryStatisticalMeasureGrammar \(\)](#)

## Private Member Functions

- void [initialise \(\)](#)  
*Initialisation function.*
- void [initialiseGrammar \(\)](#)  
*Initialise the grammar.*
- void [initialiseDebugSupport \(\)](#)  
*Initialise debug support.*
- void [assignNamesToRules \(\)](#)  
*Assign names to the rules.*
- void [initialiseRulesDebugging \(\)](#)  
*Initialise the debugging of rules.*

## Private Attributes

- `qi::rule< Iterator, BinaryStatisticalMeasureAttribute(), qi::space_type > binaryStatisticalMeasureRule`
- `BinaryStatisticalMeasureTypeParser binaryStatisticalMeasureTypeParser`

### 6.25.1 Detailed Description

```
template<typename Iterator>class multiscale::verification::BinaryStatisticalMeasureGrammar< Iterator >
```

The grammar for parsing binary statistical measure statements.

Definition at line 24 of file BinaryStatisticalMeasureGrammar.hpp.

## 6.25.2 Constructor & Destructor Documentation

6.25.2.1 template<typename Iterator> multiscale::verification::BinaryStatisticalMeasureGrammar< Iterator >::BinaryStatisticalMeasureGrammar( )

Definition at line 17 of file BinaryStatisticalMeasureGrammarDefinition.hpp.

References multiscale::verification::BinaryStatisticalMeasureGrammar< Iterator >::initialise().

## 6.25.3 Member Function Documentation

6.25.3.1 template<typename Iterator> void multiscale::verification::BinaryStatisticalMeasureGrammar< Iterator >::assignNamesToRules( )  
[private]

Assign names to the rules.

Definition at line 50 of file BinaryStatisticalMeasureGrammarDefinition.hpp.

6.25.3.2 template<typename Iterator> void multiscale::verification::BinaryStatisticalMeasureGrammar< Iterator >::initialise( )  
[private]

Initialisation function.

Definition at line 27 of file BinaryStatisticalMeasureGrammarDefinition.hpp.

Referenced by multiscale::verification::BinaryStatisticalMeasureGrammar< Iterator >::BinaryStatisticalMeasureGrammar().

6.25.3.3 template<typename Iterator> void multiscale::verification::BinaryStatisticalMeasureGrammar< Iterator >::initialiseDebugSupport( )  
[private]

Initialise debug support.

Definition at line 41 of file BinaryStatisticalMeasureGrammarDefinition.hpp.

6.25.3.4 template<typename Iterator> void multiscale::verification::BinaryStatisticalMeasureGrammar< Iterator >::initialiseGrammar( )  
[private]

Initialise the grammar.

Definition at line 34 of file BinaryStatisticalMeasureGrammarDefinition.hpp.

---

---

**6.25.3.5 template<typename Iterator > void multiscale::verification::BinaryStatisticalMeasureGrammar< Iterator >::initialiseRulesDebugging ( ) [private]**

Initialise the debugging of rules.

Definition at line 56 of file BinaryStatisticalMeasureGrammarDefinition.hpp.

#### 6.25.4 Member Data Documentation

**6.25.4.1 template<typename Iterator> qi::rule<Iterator, BinaryStatisticalMeasureAttribute(), qi::space\_type> multiscale::verification::BinaryStatisticalMeasureGrammar< Iterator >::binaryStatisticalMeasureRule [private]**

The rule for parsing a binary statistical measure

Definition at line 30 of file BinaryStatisticalMeasureGrammar.hpp.

**6.25.4.2 template<typename Iterator> BinaryStatisticalMeasureTypeParser multiscale::verification::BinaryStatisticalMeasureGrammar< Iterator >::binaryStatisticalMeasureTypeParser [private]**

The binary statistical measure type parser

Definition at line 36 of file BinaryStatisticalMeasureGrammar.hpp.

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

- BinaryStatisticalMeasureGrammar.hpp
- BinaryStatisticalMeasureGrammarDefinition.hpp

### 6.26 multiscale::verification::BinaryStatisticalMeasureTypeParser Struct Reference

Symbol table and parser for the binary statistical measure type.

```
#include <SymbolTables.hpp>
```

#### Public Member Functions

- [BinaryStatisticalMeasureTypeParser \(\)](#)

#### 6.26.1 Detailed Description

Symbol table and parser for the binary statistical measure type.

## **6.27 multiscale::verification::BinaryStatisticalNumericAttribute Class Reference**

Definition at line 45 of file SymbolTables.hpp.

### **6.26.2 Constructor & Destructor Documentation**

**6.26.2.1 multiscale::verification::BinaryStatisticalMeasureTypeParser::BinaryStatisticalMeasureTypeParser( )**  
[inline]

Definition at line 48 of file SymbolTables.hpp.

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

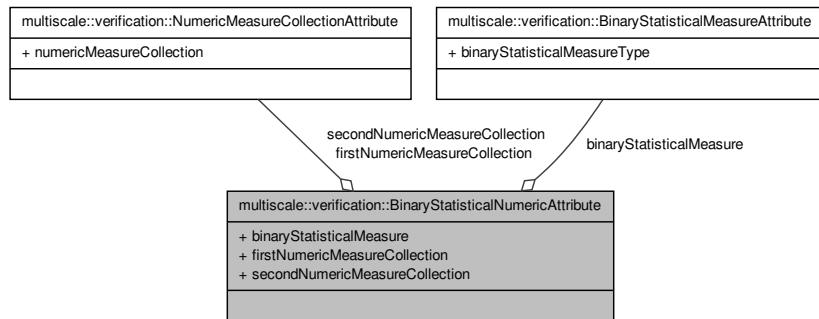
- SymbolTables.hpp

## **6.27 multiscale::verification::BinaryStatisticalNumericAttribute - Class Reference**

Class for representing a binary statistical numeric attribute.

```
#include <BinaryStatisticalNumericAttribute.hpp>
```

Collaboration diagram for multiscale::verification::BinaryStatisticalNumericAttribute:



### **Public Attributes**

- `BinaryStatisticalMeasureAttribute binaryStatisticalMeasure`
- `NumericMeasureCollectionAttribute firstNumericMeasureCollection`
- `NumericMeasureCollectionAttribute secondNumericMeasureCollection`

### 6.27.1 Detailed Description

Class for representing a binary statistical numeric attribute.

Definition at line 15 of file BinaryStatisticalNumericAttribute.hpp.

### 6.27.2 Member Data Documentation

#### 6.27.2.1 **BinaryStatisticalMeasureAttribute multiscale::verification::BinaryStatisticalNumericAttribute::binaryStatisticalMeasure**

The binary statistical subset measure

Definition at line 20 of file BinaryStatisticalNumericAttribute.hpp.

Referenced by multiscale::verification::TemporalNumericVisitor::operator()().

#### 6.27.2.2 **NumericMeasureCollectionAttribute multiscale::verification::BinaryStatisticalNumericAttribute::firstNumericMeasureCollection**

The first considered numeric measure collection

Definition at line 22 of file BinaryStatisticalNumericAttribute.hpp.

Referenced by multiscale::verification::TemporalNumericVisitor::operator()().

#### 6.27.2.3 **NumericMeasureCollectionAttribute multiscale::verification::BinaryStatisticalNumericAttribute::secondNumericMeasureCollection**

The second considered numeric measure collection

Definition at line 24 of file BinaryStatisticalNumericAttribute.hpp.

Referenced by multiscale::verification::TemporalNumericVisitor::operator()().

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

- [BinaryStatisticalNumericAttribute.hpp](#)

## 6.28 **multiscale::verification::BinaryStatisticalQuantileMeasure-Attribute Class Reference**

Class for representing a binary statistical quantile measure attribute.

```
#include <BinaryStatisticalQuantileMeasureAttribute.hpp>
```

### Public Attributes

- [BinaryStatisticalQuantileMeasureType binaryStatisticalQuantileMeasureType](#)

### **6.28.1 Detailed Description**

Class for representing a binary statistical quantile measure attribute.

Definition at line 29 of file `BinaryStatisticalQuantileMeasureAttribute.hpp`.

### **6.28.2 Member Data Documentation**

#### **6.28.2.1 `BinaryStatisticalQuantileMeasureType` multiscale::verification::BinaryStatisticalQuantileMeasureAttribute::binaryStatisticalQuantileMeasureType**

The binary statistical quantile measure type

Definition at line 34 of file `BinaryStatisticalQuantileMeasureAttribute.hpp`.

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

- `BinaryStatisticalQuantileMeasureAttribute.hpp`

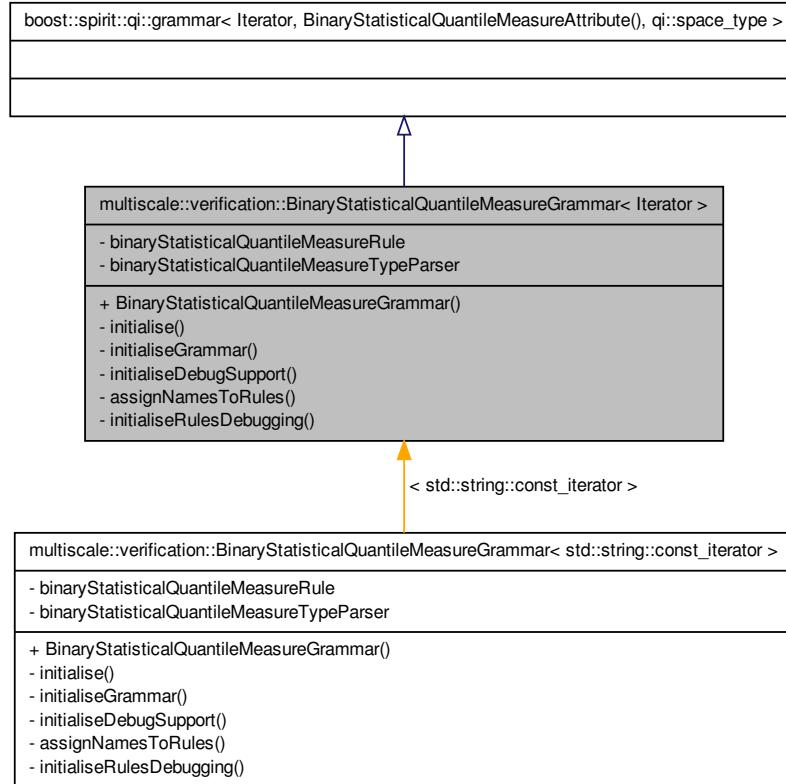
## **6.29 multiscale::verification::BinaryStatisticalQuantileMeasureGrammar< Iterator > Class Template Reference**

The grammar for parsing binary statistical quantile measure statements.

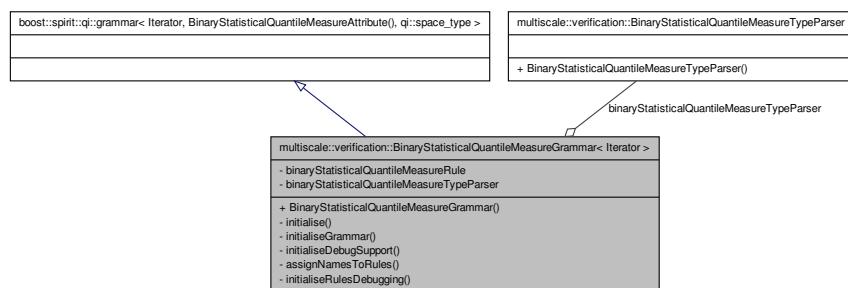
```
#include <BinaryStatisticalQuantileMeasureGrammar.hpp>
```

Inheritance diagram for multiscale::verification::BinaryStatisticalQuantileMeasure-

Grammar< Iterator >:



Collaboration diagram for `multiscale::verification::BinaryStatisticalQuantileMeasureGrammar< Iterator >`:



## Public Member Functions

- [BinaryStatisticalQuantileMeasureGrammar \(\)](#)

## Private Member Functions

- void [initialise \(\)](#)  
*Initialisation function.*
- void [initialiseGrammar \(\)](#)  
*Initialise the grammar.*
- void [initialiseDebugSupport \(\)](#)  
*Initialise debug support.*
- void [assignNamesToRules \(\)](#)  
*Assign names to the rules.*
- void [initialiseRulesDebugging \(\)](#)  
*Initialise the debugging of rules.*

## Private Attributes

- qi::rule< Iterator, [BinaryStatisticalQuantileMeasureAttribute\(\)](#), qi::space\_type > [binaryStatisticalQuantileMeasureRule](#)
- [BinaryStatisticalQuantileMeasureTypeParser](#) [binaryStatisticalQuantileMeasureTypeParser](#)

### 6.29.1 Detailed Description

```
template<typename Iterator>class multiscale::verification::BinaryStatisticalQuantileMeasure-
Grammar< Iterator >
```

The grammar for parsing binary statistical quantile measure statements.

Definition at line 24 of file `BinaryStatisticalQuantileMeasureGrammar.hpp`.

### 6.29.2 Constructor & Destructor Documentation

```
6.29.2.1 template<typename Iterator > multiscale::verification::-
BinaryStatisticalQuantileMeasureGrammar< Iterator
>::BinaryStatisticalQuantileMeasureGrammar( )
```

Definition at line 17 of file `BinaryStatisticalQuantileMeasureGrammarDefinition.hpp`.

References `multiscale::verification::BinaryStatisticalQuantileMeasureGrammar< Iterator >::initialise()`.

### 6.29.3 Member Function Documentation

6.29.3.1 `template<typename Iterator > void multiscale::verification::BinaryStatisticalQuantileMeasureGrammar< Iterator >::assignNamesToRules( ) [private]`

Assign names to the rules.

Definition at line 50 of file BinaryStatisticalQuantileMeasureGrammarDefinition.hpp.

6.29.3.2 `template<typename Iterator > void multiscale::verification::BinaryStatisticalQuantileMeasureGrammar< Iterator >::initialise( ) [private]`

Initialisation function.

Definition at line 27 of file BinaryStatisticalQuantileMeasureGrammarDefinition.hpp.

Referenced by `multiscale::verification::BinaryStatisticalQuantileMeasureGrammar< Iterator >::BinaryStatisticalQuantileMeasureGrammar()`.

6.29.3.3 `template<typename Iterator > void multiscale::verification::BinaryStatisticalQuantileMeasureGrammar< Iterator >::initialiseDebugSupport( ) [private]`

Initialise debug support.

Definition at line 41 of file BinaryStatisticalQuantileMeasureGrammarDefinition.hpp.

6.29.3.4 `template<typename Iterator > void multiscale::verification::BinaryStatisticalQuantileMeasureGrammar< Iterator >::initialiseGrammar( ) [private]`

Initialise the grammar.

Definition at line 34 of file BinaryStatisticalQuantileMeasureGrammarDefinition.hpp.

6.29.3.5 `template<typename Iterator > void multiscale::verification::BinaryStatisticalQuantileMeasureGrammar< Iterator >::initialiseRulesDebugging( ) [private]`

Initialise the debugging of rules.

Definition at line 56 of file BinaryStatisticalQuantileMeasureGrammarDefinition.hpp.

### 6.29.4 Member Data Documentation

---

**6.29.4.1 template<typename Iterator> qi::rule<Iterator, BinaryStatisticalQuantileMeasureAttribute(), qi::space\_type> multiscale::verification::BinaryStatisticalQuantileMeasureGrammar<Iterator>::binaryStatisticalQuantileMeasureRule [private]**

The rule for parsing a binary statistical quantile measure

Definition at line 30 of file BinaryStatisticalQuantileMeasureGrammar.hpp.

**6.29.4.2 template<typename Iterator> BinaryStatisticalQuantileMeasureTypeParser multiscale::verification::BinaryStatisticalQuantileMeasureGrammar<Iterator>::binaryStatisticalQuantileMeasureTypeParser [private]**

The binary statistical quantile measure type parser

Definition at line 36 of file BinaryStatisticalQuantileMeasureGrammar.hpp.

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

- BinaryStatisticalQuantileMeasureGrammar.hpp
- BinaryStatisticalQuantileMeasureGrammarDefinition.hpp

## **6.30 multiscale::verification::BinaryStatisticalQuantileMeasureTypeParser Struct Reference**

Symbol table and parser for the binary statistical quantile measure type.

```
#include <SymbolTables.hpp>
```

### **Public Member Functions**

- [BinaryStatisticalQuantileMeasureTypeParser \(\)](#)

#### **6.30.1 Detailed Description**

Symbol table and parser for the binary statistical quantile measure type.

Definition at line 57 of file SymbolTables.hpp.

#### **6.30.2 Constructor & Destructor Documentation**

**6.30.2.1 multiscale::verification::BinaryStatisticalQuantileMeasureTypeParser::BinaryStatisticalQuantileMeasureTypeParser( ) [inline]**

Definition at line 60 of file SymbolTables.hpp.

References multiscale::verification::Quartile.

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

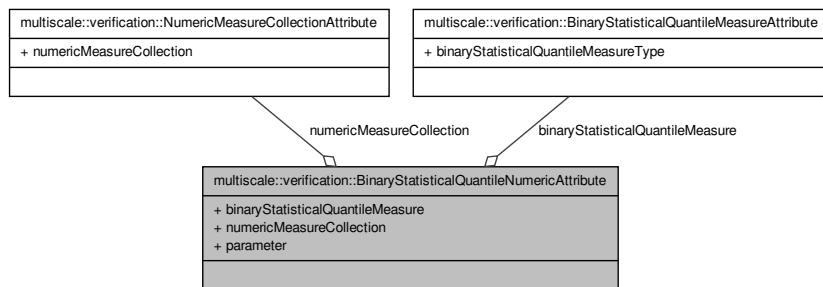
- SymbolTables.hpp

## 6.31 multiscale::verification::BinaryStatisticalQuantileNumeric-Attribute Class Reference

Class for representing a binary statistical quantile numeric attribute.

```
#include <BinaryStatisticalQuantileNumericAttribute.hpp>
```

Collaboration diagram for multiscale::verification::BinaryStatisticalQuantileNumeric-Attribute:



### Public Attributes

- [BinaryStatisticalQuantileMeasureAttribute binaryStatisticalQuantileMeasure](#)
- [NumericMeasureCollectionAttribute numericMeasureCollection](#)
- [double parameter](#)

#### 6.31.1 Detailed Description

Class for representing a binary statistical quantile numeric attribute.

Definition at line 15 of file `BinaryStatisticalQuantileNumericAttribute.hpp`.

#### 6.31.2 Member Data Documentation

**6.31.2.1 BinaryStatisticalQuantileMeasureAttribute  
multiscale::verification::BinaryStatisticalQuantileNumericAttribute-  
::binaryStatisticalQuantileMeasure**

The binary statistical quantile measure

Definition at line 20 of file BinaryStatisticalQuantileNumericAttribute.hpp.

Referenced by multiscale::verification::TemporalNumericVisitor::operator()().

**6.31.2.2 NumericMeasureCollectionAttribute multiscale::verification::Binary-  
StatisticalQuantileNumericAttribute::numericMeasureCollection**

The considered numeric measure collection

Definition at line 22 of file BinaryStatisticalQuantileNumericAttribute.hpp.

Referenced by multiscale::verification::TemporalNumericVisitor::operator()().

**6.31.2.3 double multiscale::verification::BinaryStatisticalQuantileNumeric-  
Attribute::parameter**

The considered parameter

Definition at line 24 of file BinaryStatisticalQuantileNumericAttribute.hpp.

Referenced by multiscale::verification::TemporalNumericVisitor::operator()().

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

- BinaryStatisticalQuantileNumericAttribute.hpp

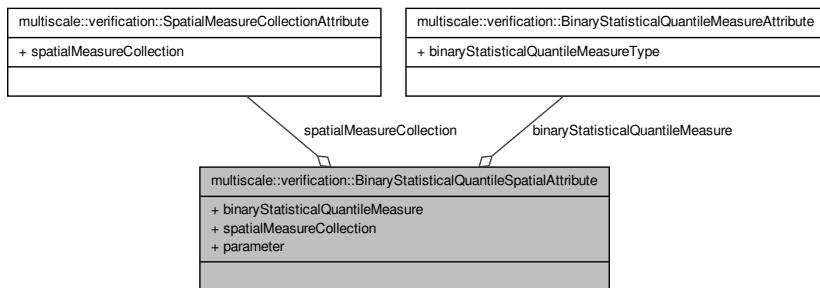
**6.32 multiscale::verification::BinaryStatisticalQuantileSpatial-  
Attribute Class Reference**

Class for representing a binary statistical quantile spatial attribute.

```
#include <BinaryStatisticalQuantileSpatialAttribute.hpp>
```

Collaboration diagram for multiscale::verification::BinaryStatisticalQuantileSpatial-

Attribute:



## Public Attributes

- **BinaryStatisticalQuantileMeasureAttribute** `binaryStatisticalQuantileMeasure`
- **SpatialMeasureCollectionAttribute** `spatialMeasureCollection`
- double `parameter`

### 6.32.1 Detailed Description

Class for representing a binary statistical quantile spatial attribute.

Definition at line 15 of file `BinaryStatisticalQuantileSpatialAttribute.hpp`.

### 6.32.2 Member Data Documentation

#### 6.32.2.1 **BinaryStatisticalQuantileMeasureAttribute**

`multiscale::verification::BinaryStatisticalQuantileSpatialAttribute-::binaryStatisticalQuantileMeasure`

The binary statistical quantile measure

Definition at line 20 of file `BinaryStatisticalQuantileSpatialAttribute.hpp`.

Referenced by `multiscale::verification::NumericVisitor::operator()()`.

#### 6.32.2.2 **double multiscale::verification::BinaryStatisticalQuantileSpatialAttribute-::parameter**

The considered parameter

Definition at line 24 of file `BinaryStatisticalQuantileSpatialAttribute.hpp`.

Referenced by `multiscale::verification::NumericVisitor::operator()()`.

**6.32.2.3 SpatialMeasureCollectionAttribute multiscale::verification::BinaryStatisticalQuantileSpatialAttribute::spatialMeasureCollection**

The considered spatial measure collection

Definition at line 22 of file BinaryStatisticalQuantileSpatialAttribute.hpp.

Referenced by multiscale::verification::NumericVisitor::operator()().

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

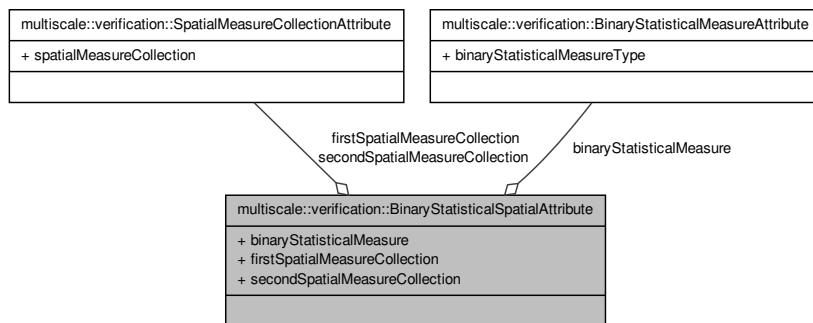
- [BinaryStatisticalQuantileSpatialAttribute.hpp](#)

## **6.33 multiscale::verification::BinaryStatisticalSpatialAttribute Class Reference**

Class for representing a binary statistical spatial attribute.

```
#include <BinaryStatisticalSpatialAttribute.hpp>
```

Collaboration diagram for multiscale::verification::BinaryStatisticalSpatialAttribute:



### **Public Attributes**

- [BinaryStatisticalMeasureAttribute binaryStatisticalMeasure](#)
- [SpatialMeasureCollectionAttribute firstSpatialMeasureCollection](#)
- [SpatialMeasureCollectionAttribute secondSpatialMeasureCollection](#)

### **6.33.1 Detailed Description**

Class for representing a binary statistical spatial attribute.

Definition at line 15 of file BinaryStatisticalSpatialAttribute.hpp.

### 6.33.2 Member Data Documentation

#### 6.33.2.1 **BinaryStatisticalMeasureAttribute multiscale::verification::BinaryStatisticalSpatialAttribute::binaryStatisticalMeasure**

The binary statistical subset measure

Definition at line 20 of file BinaryStatisticalSpatialAttribute.hpp.

Referenced by multiscale::verification::NumericVisitor::operator()().

#### 6.33.2.2 **SpatialMeasureCollectionAttribute multiscale::verification::BinaryStatisticalSpatialAttribute::firstSpatialMeasureCollection**

The first considered spatial measure collection

Definition at line 22 of file BinaryStatisticalSpatialAttribute.hpp.

Referenced by multiscale::verification::NumericVisitor::operator()().

#### 6.33.2.3 **SpatialMeasureCollectionAttribute multiscale::verification::BinaryStatisticalSpatialAttribute::secondSpatialMeasureCollection**

The second considered spatial measure collection

Definition at line 24 of file BinaryStatisticalSpatialAttribute.hpp.

Referenced by multiscale::verification::NumericVisitor::operator()().

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

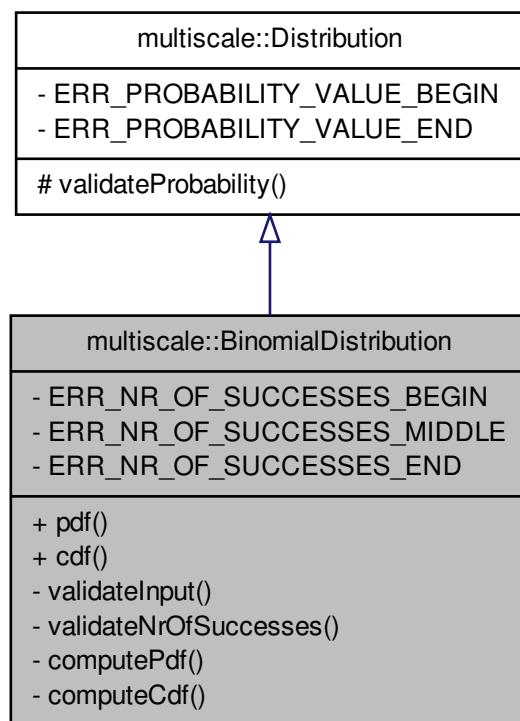
- BinaryStatisticalSpatialAttribute.hpp

## 6.34 multiscale::BinomialDistribution Class Reference

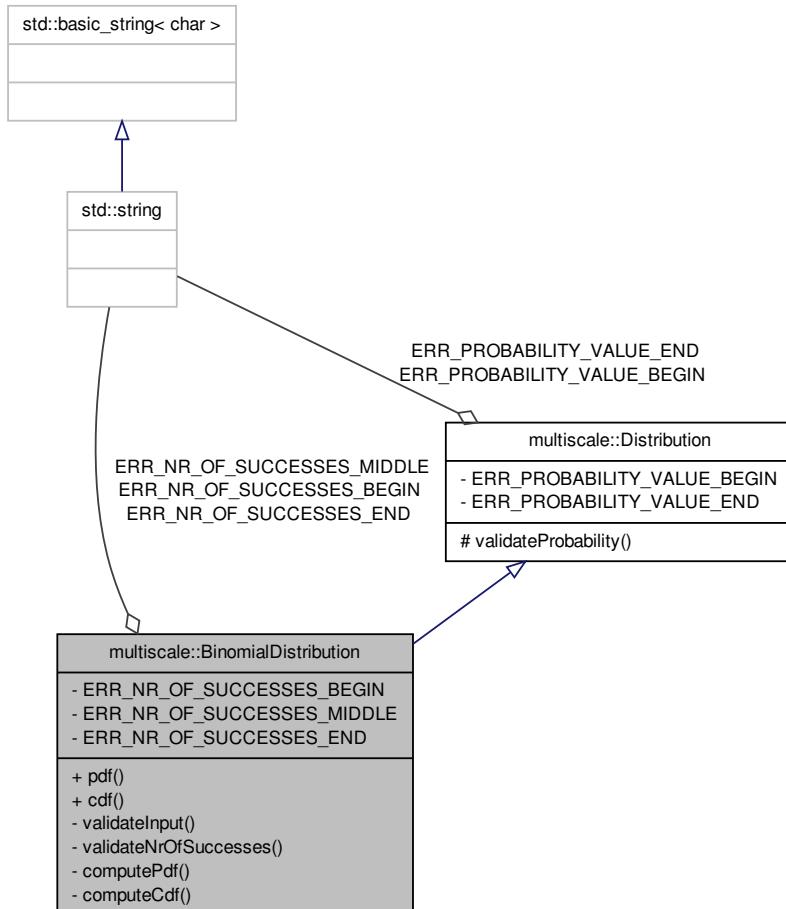
Class for analysing Binomial distributed data.

```
#include <BinomialDistribution.hpp>
```

Inheritance diagram for multiscale::BinomialDistribution:



Collaboration diagram for multiscale::BinomialDistribution:



## Static Public Member Functions

- static double **pdf** (unsigned int nrOfObservations, unsigned int nrOfSuccesses, double probability)

*Compute the value of the probability distribution/mass function (pdf) for a binomial distribution.*

- static double **cdf** (unsigned int nrOfObservations, unsigned int nrOfSuccesses, double probability)

*Compute the value of the cumulative distribution function (cdf) for a binomial distribution.*

### Static Private Member Functions

- static void `validateInput` (unsigned int nrOfObservations, unsigned int nrOfSuccesses, double probability)
 

*Validate the given input data.*
- static void `validateNrOfSuccesses` (unsigned int nrOfObservations, unsigned int nrOfSuccesses)
 

*Check if the number of true observations is less than or equal to the total number of observations.*
- static double `computePdf` (unsigned int nrOfObservations, unsigned int nrOfSuccesses, double probability)
 

*Compute the value of the probability distribution function for a binomial distribution.*
- static double `computeCdf` (unsigned int nrOfObservations, unsigned int nrOfSuccesses, double probability)
 

*Compute the value of the cumulative distribution function for a binomial distribution.*

### Static Private Attributes

- static const std::string `ERR_NR_OF_SUCCESSES_BEGIN` = "The given number of successes ("
- static const std::string `ERR_NR_OF_SUCCESSES_MIDDLE` = ") should be less than or equal to the total number of observations ("
- static const std::string `ERR_NR_OF_SUCCESSES_END` = ")."

#### 6.34.1 Detailed Description

Class for analysing Binomial distributed data.

Definition at line 12 of file BinomialDistribution.hpp.

#### 6.34.2 Member Function Documentation

##### 6.34.2.1 double `BinomialDistribution::cdf` ( `unsigned int nrOfObservations, unsigned int nrOfSuccesses, double probability` ) [static]

Compute the value of the cumulative distribution function (cdf) for a binomial distribution.

#### Parameters

<code>nrOfObservations</code>	The total number of observations
<code>nrOfSuccesses</code>	The number of successes
<code>probability</code>	The probability p used by the cumulative distribution function

Definition at line 17 of file BinomialDistribution.cpp.

References computeCdf(), and validateInput().

Referenced by computeCdf(), multiscale::verification::ModelChecker::updateAlternativeHypothesisPValue(), and multiscale::verification::ModelChecker::updateNullHypothesisPValue().

**6.34.2.2 double BinomialDistribution::computeCdf ( unsigned int *nrOfObservations*, unsigned int *nrOfSuccesses*, double *probability* ) [static, private]**

Compute the value of the cumulative distribution function for a binomial distribution.

#### Parameters

<i>nrOf- Observations</i>	The total number of observations
<i>nrOf- Successes</i>	The number of successes
<i>probability</i>	The probability p used by the cumulative distribution function

Definition at line 50 of file BinomialDistribution.cpp.

References cdf().

Referenced by cdf().

**6.34.2.3 double BinomialDistribution::computePdf ( unsigned int *nrOfObservations*, unsigned int *nrOfSuccesses*, double *probability* ) [static, private]**

Compute the value of the probability distribution function for a binomial distribution.

#### Parameters

<i>nrOf- Observations</i>	The total number of observations
<i>nrOf- Successes</i>	The number of successes
<i>probability</i>	The probability p used by the cumulative distribution function

Definition at line 43 of file BinomialDistribution.cpp.

References pdf().

Referenced by pdf().

**6.34.2.4 double BinomialDistribution::pdf ( unsigned int *nrOfObservations*, unsigned int *nrOfSuccesses*, double *probability* ) [static]**

Compute the value of the probability distribution/mass function (pdf) for a binomial distribution.

**Parameters**

<i>nrOfObservations</i>	The total number of observations
<i>nrOfSuccesses</i>	The number of successes
<i>probability</i>	The probability p used by the cumulative distribution function

Definition at line 10 of file BinomialDistribution.cpp.

References computePdf(), and validateInput().

Referenced by multiscale::verification::BayesianModelChecker::computeBinomialPDF(), and computePdf().

**6.34.2.5 void BinomialDistribution::validateInput ( unsigned int *nrOfObservations*,  
unsigned int *nrOfSuccesses*, double *probability* ) [static, private]**

Validate the given input data.

**Parameters**

<i>nrOfObservations</i>	The total number of observations
<i>nrOfSuccesses</i>	The number of successes
<i>probability</i>	The probability p used by the cumulative distribution function

Definition at line 24 of file BinomialDistribution.cpp.

References validateNrOfSuccesses(), and multiscale::Distribution::validateProbability().

Referenced by cdf(), and pdf().

**6.34.2.6 void BinomialDistribution::validateNrOfSuccesses ( unsigned int  
*nrOfObservations*, unsigned int *nrOfSuccesses* ) [static, private]**

Check if the number of true observations is less than or equal to the total number of observations.

**Parameters**

<i>nrOfObservations</i>	The total number of observations
<i>nrOfSuccesses</i>	The number of successes

Definition at line 30 of file BinomialDistribution.cpp.

References ERR\_NR\_OF\_SUCCESSES\_BEGIN, ERR\_NR\_OF\_SUCCESSES\_END, ERR\_NR\_OF\_SUCCESSES\_MIDDLE, and multiscale::StringManipulator::toString().

Referenced by validateInput().

### 6.34.3 Member Data Documentation

6.34.3.1 `const std::string BinomialDistribution::ERR_NR_OF_SUCCESSES_BEGIN = "The given number of successes (" [static, private]`

Definition at line 73 of file BinomialDistribution.hpp.

Referenced by validateNrOfSuccesses().

6.34.3.2 `const std::string BinomialDistribution::ERR_NR_OF_SUCCESSES_END = ")" [static, private]`

Definition at line 75 of file BinomialDistribution.hpp.

Referenced by validateNrOfSuccesses().

6.34.3.3 `const std::string BinomialDistribution::ERR_NR_OF_SUCCESSES_MIDDLE = ") should be less than or equal to the total number of observations (" [static, private]`

Definition at line 74 of file BinomialDistribution.hpp.

Referenced by validateNrOfSuccesses().

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

- BinomialDistribution.hpp
- BinomialDistribution.cpp

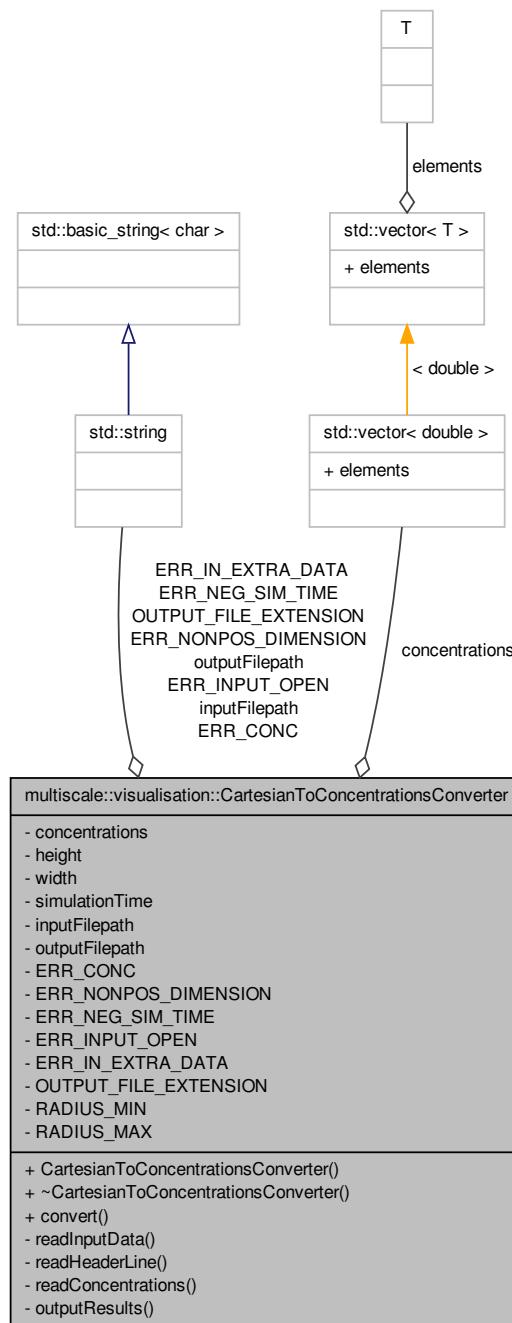
## 6.35 multiscale::visualisation::CartesianToConcentrationsConverter Class Reference

Scale the values of the rectangular geometry grid cells.

```
#include <CartesianToConcentrationsConverter.hpp>
```

Collaboration diagram for multiscale::visualisation::CartesianToConcentrations-

Converter:



## Public Member Functions

- `CartesianToConcentrationsConverter` (const std::string &`inputFilepath`, const std::string &`outputFilepath`)
- `~CartesianToConcentrationsConverter` ()
- void `convert` ()

*Start the conversion.*

## Private Member Functions

- void `readInputData` ()  
*Read the input data.*
- void `readHeaderLine` (std::ifstream &`fin`)  
*Read the header line.*
- void `readConcentrations` (std::ifstream &`fin`)  
*Read the concentrations.*
- void `outputResults` ()  
*Output the results.*

## Private Attributes

- std::vector< double > `concentrations`
- unsigned long `height`
- unsigned long `width`
- double `simulationTime`
- std::string `inputFilepath`
- std::string `outputFilepath`

## Static Private Attributes

- static const std::string `ERR_CONC` = "All `concentrations` have to be between 0 and 1."
- static const std::string `ERR_NONPOS_DIMENSION` = "The dimensions N and M must be positive."
- static const std::string `ERR_NEG_SIM_TIME` = "The simulation time must be non-negative."
- static const std::string `ERR_INPUT_OPEN` = "The input file could not be opened"
- static const std::string `ERR_IN_EXTRA_DATA` = "The input file contains more data than required."
- static const std::string `OUTPUT_FILE_EXTENSION` = ".out"
- static const double `RADIUS_MIN` = 0.001
- static const double `RADIUS_MAX` = 0.3

### **6.35.1 Detailed Description**

Scale the values of the rectangular geometry grid cells.

Definition at line 13 of file `CartesianToConcentrationsConverter.hpp`.

### **6.35.2 Constructor & Destructor Documentation**

**6.35.2.1 `CartesianToConcentrationsConverter::CartesianToConcentrationsConverter ( const std::string & inputFilepath, const std::string & outputFilepath )`**

Definition at line 16 of file `CartesianToConcentrationsConverter.cpp`.

References `height`, `simulationTime`, and `width`.

**6.35.2.2 `CartesianToConcentrationsConverter::~CartesianToConcentrationsConverter ( )`**

Definition at line 26 of file `CartesianToConcentrationsConverter.cpp`.

### **6.35.3 Member Function Documentation**

**6.35.3.1 `void CartesianToConcentrationsConverter::convert ( )`**

Start the conversion.

Definition at line 28 of file `CartesianToConcentrationsConverter.cpp`.

References `outputResults()`, and `readInputData()`.

**6.35.3.2 `void CartesianToConcentrationsConverter::outputResults ( [private] )`**

Output the results.

Definition at line 86 of file `CartesianToConcentrationsConverter.cpp`.

References `concentrations`, `multiscale::visualisation::RectangularGnuplotScriptGenerator::generateScript()`, `height`, `outputFilepath`, `simulationTime`, and `width`.

Referenced by `convert()`.

**6.35.3.3 `void CartesianToConcentrationsConverter::readConcentrations ( std::ifstream & fin ) [private]`**

Read the concentrations.

**Parameters**

<i>fin</i>	Input file stream
------------	-------------------

Definition at line 66 of file CartesianToConcentrationsConverter.cpp.

References concentrations, ERR\_CONC, height, and width.

Referenced by readInputData().

**6.35.3.4 void `CartesianToConcentrationsConverter::readHeaderLine ( std::ifstream & fin )` [private]**

Read the header line.

The header line contains values for number of concentric circles, number of sectors and simulation time

**Parameters**

<i>fin</i>	Input file stream
------------	-------------------

Definition at line 57 of file CartesianToConcentrationsConverter.cpp.

References ERR\_NEG\_SIM\_TIME, ERR\_NONPOS\_DIMENSION, height, simulationTime, and width.

Referenced by readInputData().

**6.35.3.5 void `CartesianToConcentrationsConverter::readInputData ( )` [private]**

Read the input data.

Definition at line 33 of file CartesianToConcentrationsConverter.cpp.

References ERR\_IN\_EXTRA\_DATA, ERR\_INPUT\_OPEN, inputFilepath, readConcentrations(), and readHeaderLine().

Referenced by convert().

## 6.35.4 Member Data Documentation

**6.35.4.1 `std::vector<double> multiscale::visualisation::CartesianToConcentrationsConverter::concentrations` [private]**

Concentrations received as input

Definition at line 17 of file CartesianToConcentrationsConverter.hpp.

Referenced by outputResults(), and readConcentrations().

**6.35.4.2 const std::string CartesianToConcentrationsConverter::ERR\_CONC = "All concentrations have to be between 0 and 1."** [static, private]

Definition at line 61 of file `CartesianToConcentrationsConverter.hpp`.

Referenced by `readConcentrations()`.

**6.35.4.3 const std::string CartesianToConcentrationsConverter::ERR\_IN\_EXTR-A\_DATA = "The input file contains more data than required."** [static, private]

Definition at line 65 of file `CartesianToConcentrationsConverter.hpp`.

Referenced by `readInputData()`.

**6.35.4.4 const std::string CartesianToConcentrationsConverter::ERR\_INPUT\_OPEN = "The input file could not be opened"** [static, private]

Definition at line 64 of file `CartesianToConcentrationsConverter.hpp`.

Referenced by `readInputData()`.

**6.35.4.5 const std::string CartesianToConcentrationsConverter::ERR\_NEG-SIM\_TIME = "The simulation time must be non-negative."** [static, private]

Definition at line 63 of file `CartesianToConcentrationsConverter.hpp`.

Referenced by `readHeaderLine()`.

**6.35.4.6 const std::string CartesianToConcentrationsConverter::ERR\_NONPOS-DIMENSION = "The dimensions N and M must be positive."** [static, private]

Definition at line 62 of file `CartesianToConcentrationsConverter.hpp`.

Referenced by `readHeaderLine()`.

**6.35.4.7 unsigned long multiscale::visualisation::CartesianToConcentrations-Converter::height** [private]

Height of the grid

Definition at line 19 of file `CartesianToConcentrationsConverter.hpp`.

Referenced by `CartesianToConcentrationsConverter()`, `outputResults()`, `readConcentrations()`, and `readHeaderLine()`.

6.35.4.8 **std::string multiscale::visualisation::CartesianToConcentrations-Converter::filepath** [private]

Path to the input file

Definition at line 23 of file `CartesianToConcentrationsConverter.hpp`.

Referenced by `readInputData()`.

6.35.4.9 **const std::string CartesianToConcentrationsConverter-::OUTPUT\_FILE\_EXTENSION = ".out"** [static, private]

Definition at line 67 of file `CartesianToConcentrationsConverter.hpp`.

6.35.4.10 **std::string multiscale::visualisation::CartesianToConcentrations-Converter::outputfilepath** [private]

Path to the output file

Definition at line 24 of file `CartesianToConcentrationsConverter.hpp`.

Referenced by `outputResults()`.

6.35.4.11 **const double CartesianToConcentrationsConverter::RADIUS\_MAX = 0.3** [static, private]

Definition at line 70 of file `CartesianToConcentrationsConverter.hpp`.

6.35.4.12 **const double CartesianToConcentrationsConverter::RADIUS\_MIN = 0.001** [static, private]

Definition at line 69 of file `CartesianToConcentrationsConverter.hpp`.

6.35.4.13 **double multiscale::visualisation::CartesianToConcentrationsConverter-::simulationTime** [private]

Simulation time

Definition at line 21 of file `CartesianToConcentrationsConverter.hpp`.

Referenced by `CartesianToConcentrationsConverter()`, `outputResults()`, and `readHeaderLine()`.

6.35.4.14 **unsigned long multiscale::visualisation::CartesianToConcentrations-Converter::width** [private]

Width of the grid

Definition at line 20 of file `CartesianToConcentrationsConverter.hpp`.

Referenced by `CartesianToConcentrationsConverter()`, `outputResults()`, `readConcentrations()`, and `readHeaderLine()`.

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

- `CartesianToConcentrationsConverter.hpp`

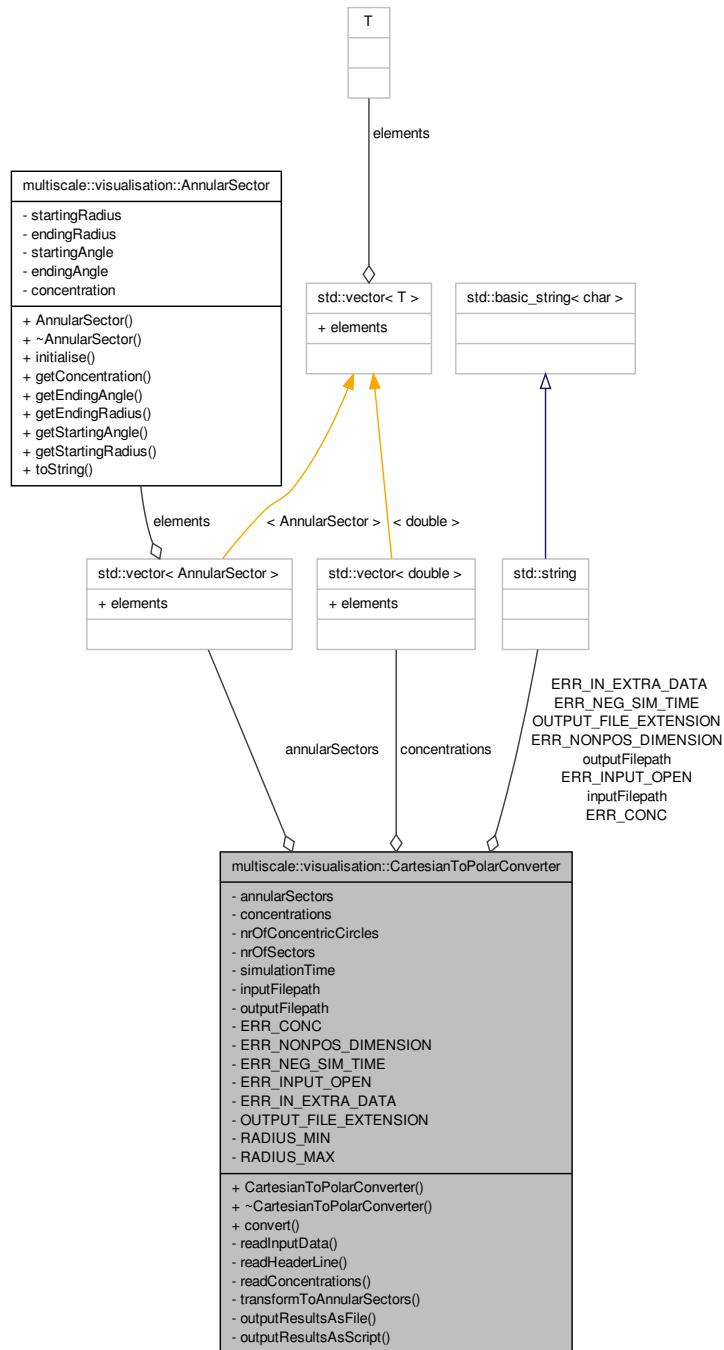
- `CartesianToConcentrationsConverter.cpp`

## **6.36 multiscale::visualisation::CartesianToPolarConverter Class - Reference**

Converter from the rectangular geometry grid cells to annular sectors.

```
#include <CartesianToPolarConverter.hpp>
```

Collaboration diagram for multiscale::visualisation::CartesianToPolarConverter:



## Public Member Functions

- `CartesianToPolarConverter` (const std::string &`inputFilepath`, const std::string &`outputFilepath`)
- `~CartesianToPolarConverter` ()
- void `convert` (bool `outputToScript`)

*Start the conversion.*

## Private Member Functions

- void `readInputData` ()  
*Read the input data.*
- void `readHeaderLine` (std::ifstream &`fin`)  
*Read the header line.*
- void `readConcentrations` (std::ifstream &`fin`)  
*Read the concentrations.*
- void `transformToAnnularSectors` ()  
*Convert the concentrations to annular sectors.*
- void `outputResultsAsFile` ()  
*Output the results as a plain file.*
- void `outputResultsAsScript` ()  
*Output the results as a gnuplot script.*

## Private Attributes

- std::vector< `AnnularSector` > `annularSectors`
- std::vector< double > `concentrations`
- unsigned long `nrOfConcentricCircles`
- unsigned long `nrOfSectors`
- double `simulationTime`
- std::string `inputFilepath`
- std::string `outputFilepath`

## Static Private Attributes

- static const std::string `ERR_CONC` = "All concentrations have to be between 0 and 1."
- static const std::string `ERR_NONPOS_DIMENSION` = "The dimensions N and M must be positive."
- static const std::string `ERR_NEG_SIM_TIME` = "The simulation time must be non-negative."
- static const std::string `ERR_INPUT_OPEN` = "The input file could not be opened"
- static const std::string `ERR_IN_EXTRA_DATA` = "The input file contains more data than required."

- static const std::string **OUTPUT\_FILE\_EXTENSION** = ".out"
- static const double **RADIUS\_MIN** = 0.001
- static const double **RADIUS\_MAX** = 0.3

### 6.36.1 Detailed Description

Converter from the rectangular geometry grid cells to annular sectors.

Definition at line 15 of file `CartesianToPolarConverter.hpp`.

### 6.36.2 Constructor & Destructor Documentation

#### 6.36.2.1 `CartesianToPolarConverter::CartesianToPolarConverter ( const std::string & inputFilepath, const std::string & outputFilepath )`

Definition at line 16 of file `CartesianToPolarConverter.cpp`.

References `nrOfConcentricCircles`, `nrOfSectors`, and `simulationTime`.

#### 6.36.2.2 `CartesianToPolarConverter::~CartesianToPolarConverter ( )`

Definition at line 26 of file `CartesianToPolarConverter.cpp`.

### 6.36.3 Member Function Documentation

#### 6.36.3.1 `void CartesianToPolarConverter::convert ( bool outputToScript )`

Start the conversion.

##### Parameters

<code>outputTo- Script</code>	Output to script or to plain file
-----------------------------------	-----------------------------------

Definition at line 28 of file `CartesianToPolarConverter.cpp`.

References `outputResultsAsFile()`, `outputResultsAsScript()`, `readInputData()`, and `transformToAnnularSectors()`.

#### 6.36.3.2 `void CartesianToPolarConverter::outputResultsAsFile ( ) [private]`

Output the results as a plain file.

Definition at line 123 of file `CartesianToPolarConverter.cpp`.

References `annularSectors`, `OUTPUT_FILE_EXTENSION`, and `outputFilepath`.

Referenced by `convert()`.

**6.36.3.3 void `CartesianToPolarConverter::outputResultsAsScript( )`**  
[private]

Output the results as a gnuplot script.

Definition at line 138 of file `CartesianToPolarConverter.cpp`.

References `annularSectors`, `multiscale::visualisation::PolarGnuplotScriptGenerator::generateScript()`, `outputFilepath`, and `simulationTime`.

Referenced by `convert()`.

**6.36.3.4 void `CartesianToPolarConverter::readConcentrations( std::ifstream & fin )`**  
[private]

Read the concentrations.

Parameters

<i>fin</i>	Input file stream
------------	-------------------

Definition at line 72 of file `CartesianToPolarConverter.cpp`.

References `concentrations`, `ERR_CONC`, `nrOfConcentricCircles`, and `nrOfSectors`.

Referenced by `readInputData()`.

**6.36.3.5 void `CartesianToPolarConverter::readHeaderLine( std::ifstream & fin )`**  
[private]

Read the header line.

The header line contains values for number of concentric circles, number of sectors and simulation time

Parameters

<i>fin</i>	Input file stream
------------	-------------------

Definition at line 63 of file `CartesianToPolarConverter.cpp`.

References `ERR_NEG_SIM_TIME`, `ERR_NONPOS_DIMENSION`, `nrOfConcentricCircles`, `nrOfSectors`, and `simulationTime`.

Referenced by `readInputData()`.

**6.36.3.6 void `CartesianToPolarConverter::readInputData( )`** [private]

Read the input data.

Definition at line 39 of file `CartesianToPolarConverter.cpp`.

References `ERR_IN_EXTRA_DATA`, `ERR_INPUT_OPEN`, `inputFilepath`, `read-`

Concentrations(), and readHeaderLine().

Referenced by convert().

**6.36.3.7 void CartesianToPolarConverter::transformToAnnularSectors ( )**  
[private]

Convert the concentrations to annular sectors.

Definition at line 92 of file CartesianToPolarConverter.cpp.

References annularSectors, concentrations, nrOfConcentricCircles, nrOfSectors, RADIUS\_MAX, and RADIUS\_MIN.

Referenced by convert().

#### 6.36.4 Member Data Documentation

**6.36.4.1 std::vector<AnnularSector> multiscale::visualisation::CartesianToPolarConverter::annularSectors**  
[private]

Resulting annular sectors

Definition at line 20 of file CartesianToPolarConverter.hpp.

Referenced by outputResultsAsFile(), outputResultsAsScript(), and transformToAnnularSectors().

**6.36.4.2 std::vector<double> multiscale::visualisation::CartesianToPolarConverter::concentrations** [private]

Concentrations received as input

Definition at line 22 of file CartesianToPolarConverter.hpp.

Referenced by readConcentrations(), and transformToAnnularSectors().

**6.36.4.3 const std::string CartesianToPolarConverter::ERR\_CONC = "All concentrations have to be between 0 and 1."** [static, private]

Definition at line 79 of file CartesianToPolarConverter.hpp.

Referenced by readConcentrations().

**6.36.4.4 const std::string CartesianToPolarConverter::ERR\_IN\_EXTRA\_DATA = "The input file contains more data than required."** [static, private]

Definition at line 83 of file CartesianToPolarConverter.hpp.

Referenced by readInputData().

**6.36.4.5 const std::string CartesianToPolarConverter::ERR\_INPUT\_OPEN = "The input file could not be opened" [static, private]**

Definition at line 82 of file `CartesianToPolarConverter.hpp`.

Referenced by `readInputData()`.

**6.36.4.6 const std::string CartesianToPolarConverter::ERR\_NEG\_SIM\_TIME = "The simulation time must be non-negative." [static, private]**

Definition at line 81 of file `CartesianToPolarConverter.hpp`.

Referenced by `readHeaderLine()`.

**6.36.4.7 const std::string CartesianToPolarConverter::ERR\_NONPOS\_DIMENSION = "The dimensions N and M must be positive." [static, private]**

Definition at line 80 of file `CartesianToPolarConverter.hpp`.

Referenced by `readHeaderLine()`.

**6.36.4.8 std::string multiscale::visualisation::CartesianToPolarConverter::inputFilepath [private]**

Path to the input file

Definition at line 32 of file `CartesianToPolarConverter.hpp`.

Referenced by `readInputData()`.

**6.36.4.9 unsigned long multiscale::visualisation::CartesianToPolarConverter::nrOfConcentricCircles [private]**

Number of concentric circles

Definition at line 25 of file `CartesianToPolarConverter.hpp`.

Referenced by `CartesianToPolarConverter()`, `readConcentrations()`, `readHeaderLine()`, and `transformToAnnularSectors()`.

**6.36.4.10 unsigned long multiscale::visualisation::CartesianToPolarConverter::nrOfSectors [private]**

Number of sectors

Definition at line 27 of file `CartesianToPolarConverter.hpp`.

Referenced by `CartesianToPolarConverter()`, `readConcentrations()`, `readHeaderLine()`, and `transformToAnnularSectors()`.

6.36.4.11 `const std::string CartesianToPolarConverter::OUTPUT_FILE_EXTENSION = ".out" [static, private]`

Definition at line 85 of file `CartesianToPolarConverter.hpp`.

Referenced by `outputResultsAsFile()`.

6.36.4.12 `std::string multiscale::visualisation::CartesianToPolarConverter::outputFilepath [private]`

Path to the output file

Definition at line 34 of file `CartesianToPolarConverter.hpp`.

Referenced by `outputResultsAsFile()`, and `outputResultsAsScript()`.

6.36.4.13 `const double CartesianToPolarConverter::RADIUS_MAX = 0.3 [static, private]`

Definition at line 88 of file `CartesianToPolarConverter.hpp`.

Referenced by `transformToAnnularSectors()`.

6.36.4.14 `const double CartesianToPolarConverter::RADIUS_MIN = 0.001 [static, private]`

Definition at line 87 of file `CartesianToPolarConverter.hpp`.

Referenced by `transformToAnnularSectors()`.

6.36.4.15 `double multiscale::visualisation::CartesianToPolarConverter::simulationTime [private]`

Simulation time corresponding to the input data

Definition at line 29 of file `CartesianToPolarConverter.hpp`.

Referenced by `CartesianToPolarConverter()`, `outputResultsAsScript()`, and `readHeaderLine()`.

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

- `CartesianToPolarConverter.hpp`
- `CartesianToPolarConverter.cpp`

## 6.37 multiscale::verification::ChangeMeasureAttribute Class - Reference

Class for representing a change measure attribute.

```
#include <ChangeMeasureAttribute.hpp>
```

#### Public Attributes

- ChangeMeasureType changeMeasureType

#### 6.37.1 Detailed Description

Class for representing a change measure attribute.

Definition at line 28 of file ChangeMeasureAttribute.hpp.

#### 6.37.2 Member Data Documentation

##### 6.37.2.1 ChangeMeasureType multiscale::verification::ChangeMeasureAttribute- ::changeMeasureType

The change measure type

Definition at line 32 of file ChangeMeasureAttribute.hpp.

Referenced by multiscale::verification::LogicPropertyVisitor::evaluateChangeLhsTemporalNumericMeasure(), and multiscale::verification::NumericMeasureCollectionVisitor::evaluateChangeTemporalNumericCollection().

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

- ChangeMeasureAttribute.hpp

## 6.38 multiscale::verification::ChangeMeasureEvaluator Class - Reference

Class for evaluating change measure expressions.

```
#include <ChangeMeasureEvaluator.hpp>
```

#### Static Public Member Functions

- static double **evaluate** (const ChangeMeasureType &changeMeasureType, double temporalNumericMeasureCollectionFirstValue, double temporalNumericMeasureCollectionSecondValue)  
*Compute the change measure value considering temporal numeric measure collection and time values.*
- static double **evaluate** (const ChangeMeasureType &changeMeasureType, double temporalNumericMeasureFirstTimepoint, double temporalNumericMeasureSecondTimepoint, unsigned long timeValueFirstTimepoint, unsigned long timeValueSecondTimepoint)

*Compute the change measure value considering the given temporal numeric measure and time values.*

### Static Private Member Functions

- static double `computeTimeValueDifference` (unsigned long `timeValueFirstTimepoint`, unsigned long `timeValueSecondTimepoint`)

*Compute the time value difference considering the given time values.*

- static double `computeNumericMeasureValueChange` (const `ChangeMeasureType` &`changeMeasureType`, double `temporalNumericMeasureFirstTimepoint`, double `temporalNumericMeasureSecondTimepoint`)

*Compute the numeric measure value change considering the given change measure and numeric values.*

#### 6.38.1 Detailed Description

Class for evaluating change measure expressions.

Definition at line 15 of file ChangeMeasureEvaluator.hpp.

#### 6.38.2 Member Function Documentation

```
6.38.2.1 static double multiscale::verification::ChangeMeasureEvaluator-
         ::computeNumericMeasureValueChange ( const ChangeMeasureType
         & changeMeasureType, double temporalNumericMeasureFirstTimepoint, double
         temporalNumericMeasureSecondTimepoint ) [inline, static,
         private]
```

Compute the numeric measure value change considering the given change measure and numeric values.

##### Parameters

<code>changeMeasureType</code>	The type of the change measure
<code>temporalNumericMeasureFirstTimepoint</code>	The temporal numeric measure value corresponding to the trace starting from the initial timepoint
<code>temporalNumericMeasureSecondTimepoint</code>	The temporal numeric measure value corresponding to the trace starting from the second timepoint

Definition at line 93 of file ChangeMeasureEvaluator.hpp.

References multiscale::Numeric::division(), and multiscale::ERR\_UNDEFINED\_ENUM\_VALUE.

Referenced by evaluate().

**6.38.2.2 static double multiscale::verification::ChangeMeasureEvaluator::computeTimeValueDifference ( unsigned long *timeValueFirstTimepoint*, unsigned long *timeValueSecondTimepoint* ) [inline, static, private]**

Compute the time value difference considering the given time values.

Time difference = (second timepoint value) - (first timepoint value)

#### Parameters

<i>timeValue-First-Timepoint</i>	The time value corresponding to the first timepoint
<i>timeValue-Second-Timepoint</i>	The time value corresponding to the second timepoint

Definition at line 80 of file ChangeMeasureEvaluator.hpp.

Referenced by evaluate().

**6.38.2.3 static double multiscale::verification::ChangeMeasureEvaluator::evaluate ( const ChangeMeasureType & *changeMeasureType*, double *temporalNumericMeasureCollectionFirstValue*, double *temporalNumericMeasureCollectionSecondValue* ) [inline, static]**

Compute the change measure value considering temporal numeric measure collection and time values.

#### Parameters

<i>change-Measure-Type</i>	The type of the change measure
<i>temporal-Numeric-Measure-Collection-FirstValue</i>	The temporal numeric measure collection value corresponding to the trace starting from the initial timepoint
<i>temporal-Numeric-Measure-Collection-Second-Value</i>	The temporal numeric measure collection value corresponding to the trace starting from the timepoint succeeding the initial timepoint

Definition at line 29 of file ChangeMeasureEvaluator.hpp.

References computeNumericMeasureValueChange().

Referenced by multiscale::verification::NumericMeasureCollectionVisitor::evaluateChangeTemporalNumericCollection().

```
6.38.2.4 static double multiscale::verification::ChangeMeasureEvaluator::evaluate (
    const ChangeMeasureType & changeMeasureType, double temporalNumeric-
    MeasureFirstTimepoint, double temporalNumericMeasureSecondTimepoint,
    unsigned long timeValueFirstTimepoint, unsigned long timeValueSecondTimepoint )
    [inline, static]
```

Compute the change measure value considering the given temporal numeric measure and time values.

#### Parameters

<i>change-     Measure-     Type</i>	The type of the change measure
<i>temporal-     Numeric-     Measure-     First-     Timepoint</i>	The temporal numeric measure value corresponding to the trace starting from the initial timepoint
<i>temporal-     Numeric-     Measure-     Second-     Timepoint</i>	The temporal numeric measure value corresponding to the trace starting from the second timepoint
<i>timeValue-     First-     Timepoint</i>	The time value corresponding to the first timepoint
<i>timeValue-     Second-     Timepoint</i>	The time value corresponding to the second timepoint

Definition at line 49 of file ChangeMeasureEvaluator.hpp.

References computeNumericMeasureValueChange(), computeTimeValueDifference(), and multiscale::Numeric::division().

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

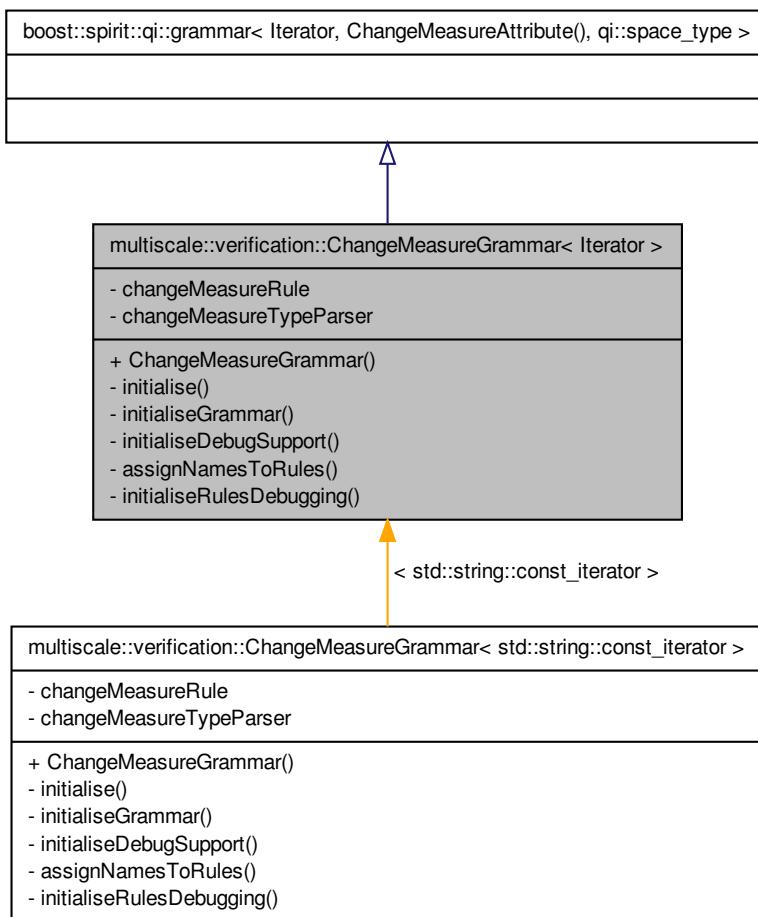
- ChangeMeasureEvaluator.hpp

## 6.39 multiscale::verification::ChangeMeasureGrammar< Iterator > Class Template Reference

The grammar for parsing change measure statements.

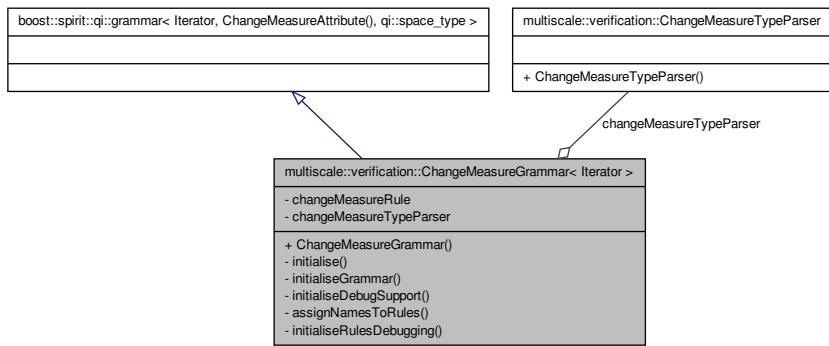
```
#include <ChangeMeasureGrammar.hpp>
```

Inheritance diagram for multiscale::verification::ChangeMeasureGrammar< Iterator >:



Collaboration diagram for multiscale::verification::ChangeMeasureGrammar< Iterator >

>:



## Public Member Functions

- `ChangeMeasureGrammar ()`

## Private Member Functions

- `void initialise ()`  
*Initialisation function.*
- `void initialiseGrammar ()`  
*Initialise the grammar.*
- `void initialiseDebugSupport ()`  
*Initialise debug support.*
- `void assignNamesToRules ()`  
*Assign names to the rules.*
- `void initialiseRulesDebugging ()`  
*Initialise the debugging of rules.*

## Private Attributes

- `qi::rule< Iterator, ChangeMeasureAttribute(), qi::space_type > changeMeasureRule`
- `ChangeMeasureTypeParser changeMeasureTypeParser`

### 6.39.1 Detailed Description

```
template<typename Iterator>class multiscale::verification::ChangeMeasureGrammar< Iterator >
```

The grammar for parsing change measure statements.

Definition at line 24 of file ChangeMeasureGrammar.hpp.

### 6.39.2 Constructor & Destructor Documentation

```
6.39.2.1 template<typename Iterator> multiscale::verification::Change-  
MeasureGrammar< Iterator >::ChangeMeasureGrammar ( )
```

Definition at line 24 of file ChangeMeasureGrammarDefinition.hpp.

References multiscale::verification::ChangeMeasureGrammar< Iterator >::initialise().

### 6.39.3 Member Function Documentation

```
6.39.3.1 template<typename Iterator> void multiscale::verification::Change-  
MeasureGrammar< Iterator >::assignNamesToRules ( ) [private]
```

Assign names to the rules.

Definition at line 63 of file ChangeMeasureGrammarDefinition.hpp.

```
6.39.3.2 template<typename Iterator> void multiscale::verification-  
::ChangeMeasureGrammar< Iterator >::initialise ( ) [private]
```

Initialisation function.

Definition at line 31 of file ChangeMeasureGrammarDefinition.hpp.

Referenced by multiscale::verification::ChangeMeasureGrammar< Iterator >::Change-  
MeasureGrammar().

```
6.39.3.3 template<typename Iterator> void multiscale::verification::Change-  
MeasureGrammar< Iterator >::initialiseDebugSupport ( ) [private]
```

Initialise debug support.

Definition at line 54 of file ChangeMeasureGrammarDefinition.hpp.

---

**6.39.3.4 template<typename Iterator > void multiscale::verification::ChangeMeasureGrammar< Iterator >::initialiseGrammar ( ) [private]**

Initialise the grammar.

Definition at line 38 of file ChangeMeasureGrammarDefinition.hpp.

**6.39.3.5 template<typename Iterator > void multiscale::verification::ChangeMeasureGrammar< Iterator >::initialiseRulesDebugging ( ) [private]**

Initialise the debugging of rules.

Definition at line 69 of file ChangeMeasureGrammarDefinition.hpp.

#### 6.39.4 Member Data Documentation

**6.39.4.1 template<typename Iterator> qi::rule<Iterator, ChangeMeasureAttribute(), qi::space\_type> multiscale::verification::ChangeMeasureGrammar< Iterator >::changeMeasureRule [private]**

The rule for parsing a change measure

Definition at line 30 of file ChangeMeasureGrammar.hpp.

**6.39.4.2 template<typename Iterator> ChangeMeasureTypeParser multiscale::verification::ChangeMeasureGrammar< Iterator >::changeMeasureTypeParser [private]**

The change measure type parser

Definition at line 35 of file ChangeMeasureGrammar.hpp.

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

- ChangeMeasureGrammar.hpp
- ChangeMeasureGrammarDefinition.hpp

### 6.40 multiscale::verification::ChangeMeasureTypeParser Struct Reference

Symbol table and parser for the change measure type.

```
#include <SymbolTables.hpp>
```

## Public Member Functions

- [ChangeMeasureTypeParser \(\)](#)

### 6.40.1 Detailed Description

Symbol table and parser for the change measure type.

Definition at line 70 of file SymbolTables.hpp.

### 6.40.2 Constructor & Destructor Documentation

#### 6.40.2.1 multiscale::verification::ChangeMeasureTypeParser::ChangeMeasureTypeParser( ) [inline]

Definition at line 73 of file SymbolTables.hpp.

References multiscale::verification::Ratio.

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

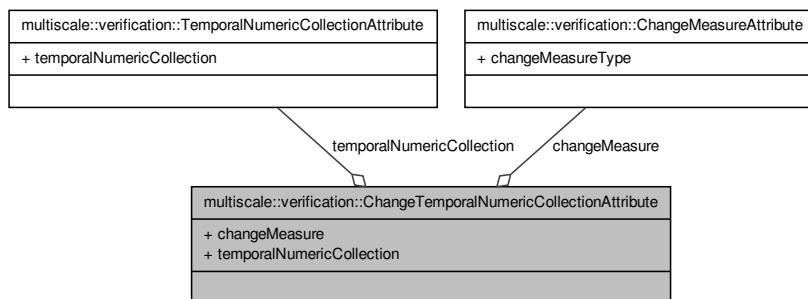
- [SymbolTables.hpp](#)

## 6.41 multiscale::verification::ChangeTemporalNumericCollection-Attribute Class Reference

Class for representing a change temporal numeric collection attribute.

```
#include <ChangeTemporalNumericCollectionAttribute.hpp>
```

Collaboration diagram for multiscale::verification::ChangeTemporalNumericCollection-Attribute:



## Public Attributes

- [ChangeMeasureAttribute changeMeasure](#)
- [TemporalNumericCollectionAttribute temporalNumericCollection](#)

### 6.41.1 Detailed Description

Class for representing a change temporal numeric collection attribute.

Definition at line 16 of file [ChangeTemporalNumericCollectionAttribute.hpp](#).

### 6.41.2 Member Data Documentation

#### 6.41.2.1 ChangeMeasureAttribute multiscale::verification::ChangeTemporal-NumericCollectionAttribute::changeMeasure

The change measure

Definition at line 21 of file [ChangeTemporalNumericCollectionAttribute.hpp](#).

Referenced by [multiscale::verification::NumericMeasureCollectionVisitor::operator\(\)\(\)](#).

#### 6.41.2.2 TemporalNumericCollectionAttribute multiscale::verification::Change-TemporalNumericCollectionAttribute::temporalNumericCollection

The temporal numeric collection

Definition at line 23 of file [ChangeTemporalNumericCollectionAttribute.hpp](#).

Referenced by [multiscale::verification::NumericMeasureCollectionVisitor::operator\(\)\(\)](#).

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

- [ChangeTemporalNumericCollectionAttribute.hpp](#)

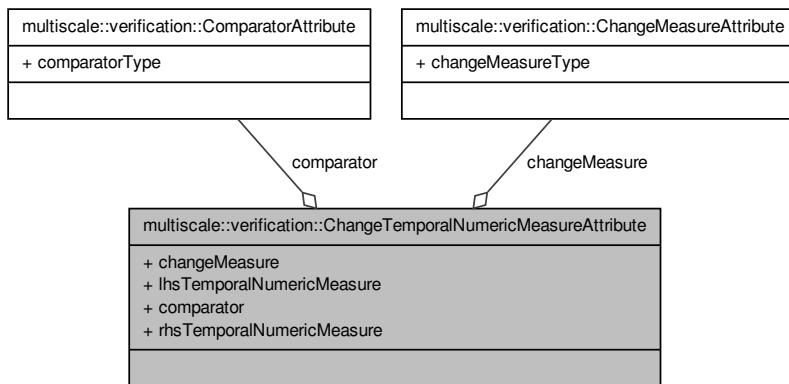
## 6.42 multiscale::verification::ChangeTemporalNumericMeasure-Attribute Class Reference

Class for representing a change temporal numeric measure attribute.

```
#include <ChangeTemporalNumericMeasureAttribute.hpp>
```

Collaboration diagram for [multiscale::verification::ChangeTemporalNumericMeasure-](#)

Attribute:



## Public Attributes

- `ChangeMeasureAttribute changeMeasure`
- `TemporalNumericMeasureType lhsTemporalNumericMeasure`
- `ComparatorAttribute comparator`
- `TemporalNumericMeasureType rhsTemporalNumericMeasure`

### 6.42.1 Detailed Description

Class for representing a change temporal numeric measure attribute.

Definition at line 17 of file `ChangeTemporalNumericMeasureAttribute.hpp`.

### 6.42.2 Member Data Documentation

#### 6.42.2.1 ChangeMeasureAttribute multiscale::verification::ChangeTemporalNumericMeasureAttribute::changeMeasure

The change measure

Definition at line 22 of file `ChangeTemporalNumericMeasureAttribute.hpp`.

Referenced by `multiscale::verification::LogicPropertyVisitor::evaluateChangeLhsTemporalNumericMeasure()`.

#### 6.42.2.2 ComparatorAttribute multiscale::verification::ChangeTemporalNumericMeasureAttribute::comparator

The comparator

Definition at line 26 of file ChangeTemporalNumericMeasureAttribute.hpp.

Referenced by multiscale::verification::LogicPropertyVisitor::evaluateChangeTemporalNumericMeasure().

#### 6.42.2.3 TemporalNumericMeasureType multiscale::verification::ChangeTemporalNumericMeasureAttribute::lhsTemporalNumericMeasure

The left hand side temporal numeric measure

Definition at line 24 of file ChangeTemporalNumericMeasureAttribute.hpp.

Referenced by multiscale::verification::LogicPropertyVisitor::evaluateChangeLhsTemporalNumericMeasure().

#### 6.42.2.4 TemporalNumericMeasureType multiscale::verification::ChangeTemporalNumericMeasureAttribute::rhsTemporalNumericMeasure

The right hand side temporal numeric measure

Definition at line 28 of file ChangeTemporalNumericMeasureAttribute.hpp.

Referenced by multiscale::verification::LogicPropertyVisitor::evaluateChangeTemporalNumericMeasure().

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

- ChangeTemporalNumericMeasureAttribute.hpp

### 6.43 multiscale::analysis::CircularityMeasure< PointType > Class Template Reference

Class for computing the circularity measure for the given collection of points.

```
#include <CircularityMeasure.hpp>
```

#### Static Public Member Functions

- static double **compute** (const std::vector< cv::Point\_< PointType >> &points)  
*Compute circularity measure for the given collection of points.*

#### 6.43.1 Detailed Description

```
template<typename PointType> class multiscale::analysis::CircularityMeasure< PointType >
```

Class for computing the circularity measure for the given collection of points.

Definition at line 16 of file CircularityMeasure.hpp.

## 6.43.2 Member Function Documentation

```
6.43.2.1 template<typename PointType > double CircularityMeasure::compute ( const  
std::vector< cv::Point_< PointType >> & points ) [static]
```

Compute circularity measure for the given collection of points.

The circularity measure is equal to the standard circularity measure described in the following paper:

Joviša Žunić, Kaoru Hirota, Paul L. Rosin, A Hu moment invariant as a shape circularity measure, Pattern Recognition, Volume 43, Issue 1, January 2010, Pages 47-57, ISSN 0031-3203, <http://dx.doi.org/10.1016/j.patcog.2009.06.017>.

Definition at line 7 of file CircularityMeasure.cpp.

References multiscale::Geometry2D::computeConvexHull(), multiscale::Numeric-  
::division(), and multiscale::Geometry2D::PI.

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

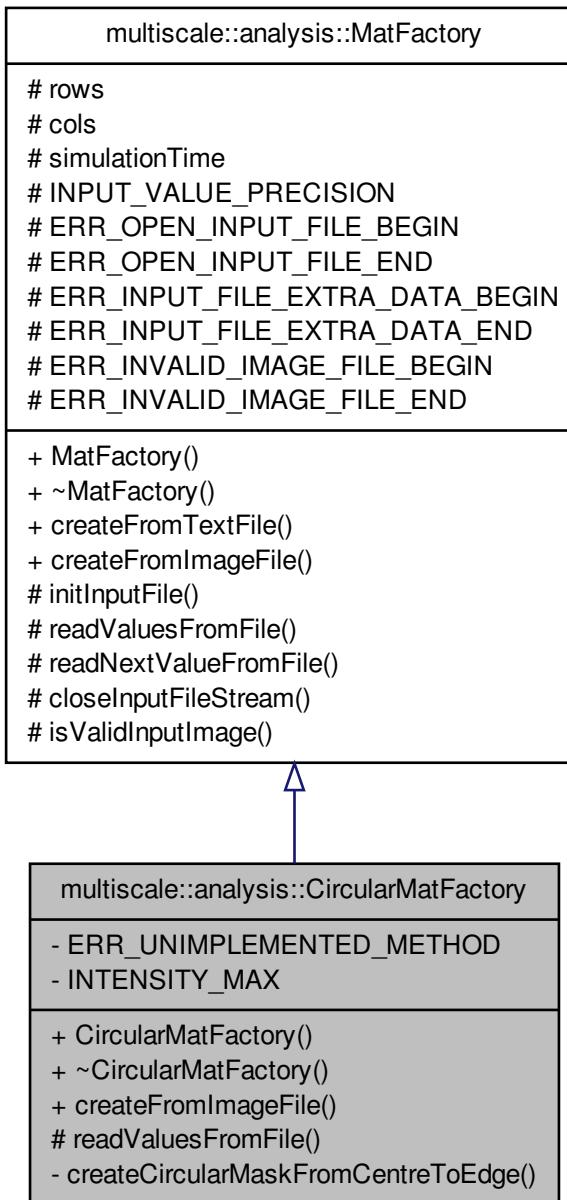
- CircularityMeasure.hpp
- CircularityMeasure.cpp

## 6.44 multiscale::analysis::CircularMatFactory Class Reference

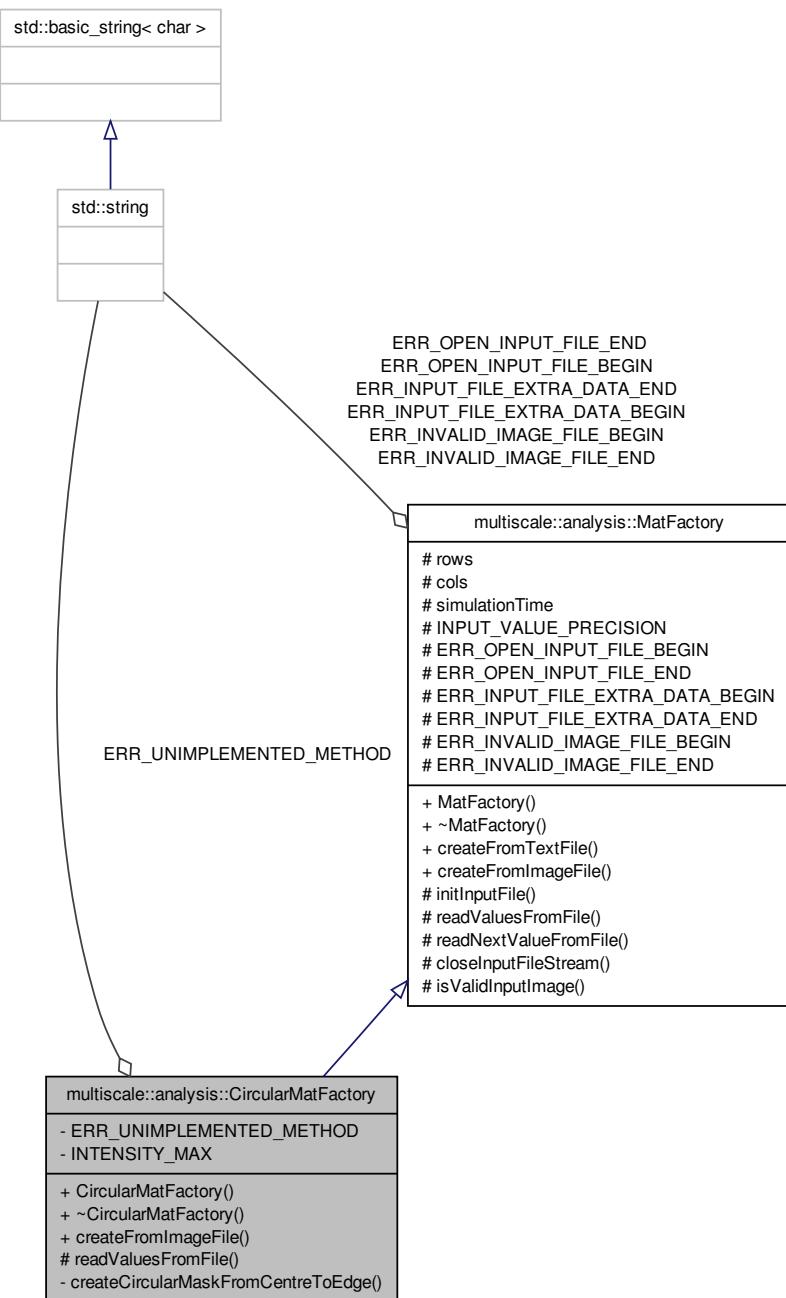
Class for creating a Mat object considering a circular grid.

```
#include <CircularMatFactory.hpp>
```

Inheritance diagram for multiscale::analysis::CircularMatFactory:



Collaboration diagram for multiscale::analysis::CircularMatFactory:



## Public Member Functions

- [CircularMatFactory \(\)](#)
- [~CircularMatFactory \(\)](#)
- cv::Mat [createFromImageFile](#) (const std::string &inputFilePath) override

*Create a Mat object from the provided image file.*

## Protected Member Functions

- void [readValuesFromFile](#) (std::ifstream &fin, cv::Mat &image) override

*Read the values from the input file.*

## Private Member Functions

- cv::Mat [createCircularMaskFromCentreToEdge](#) (const cv::Mat &image)

*Create a circular mask with origin at image centre and tangent to at least two image edges.*

## Static Private Attributes

- static const std::string [ERR\\_UNIMPLEMENTED\\_METHOD](#) = "The method you called is not implemented."
- static const int [INTENSITY\\_MAX](#) = 1

### 6.44.1 Detailed Description

Class for creating a Mat object considering a circular grid.

Definition at line 12 of file CircularMatFactory.hpp.

### 6.44.2 Constructor & Destructor Documentation

#### 6.44.2.1 CircularMatFactory::CircularMatFactory( )

Definition at line 11 of file CircularMatFactory.cpp.

#### 6.44.2.2 CircularMatFactory::~CircularMatFactory( )

Definition at line 13 of file CircularMatFactory.cpp.

### 6.44.3 Member Function Documentation

6.44.3.1 `cv::Mat CircularMatFactory::createCircularMaskFromCentreToEdge ( const cv::Mat & image ) [private]`

Create a circular mask with origin at image centre and tangent to at least two image edges.

Create a mask with 255 intensity pixels inside the circle with origin at image centre and the radius equal to the minimum distance from the image centre to one of the image edges. All the other pixels have intensity zero.

#### Parameters

<i>image</i>	The original image
--------------	--------------------

Definition at line 47 of file CircularMatFactory.cpp.

References INTENSITY\_MAX.

Referenced by createFromImageFile().

6.44.3.2 `cv::Mat CircularMatFactory::createFromImageFile ( const std::string & inputFilePath ) [override, virtual]`

Create a Mat object from the provided image file.

Create a Mat instance from the given image file

#### Parameters

<i>inputFile-Path</i>	The path to the image file
-----------------------	----------------------------

Implements [multiscale::analysis::MatFactory](#).

Definition at line 15 of file CircularMatFactory.cpp.

References createCircularMaskFromCentreToEdge(), and [multiscale::analysis::MatFactory::isValidInputImage\(\)](#).

6.44.3.3 `void CircularMatFactory::readValuesFromFile ( std::ifstream & fin, cv::Mat & image ) [override, protected, virtual]`

Read the values from the input file.

REMARK: This method is not implemented and throws an error when called.

#### Parameters

<i>fin</i>	Input file stream from which the values are read
<i>image</i>	The image to which the values are written

Implements [multiscale::analysis::MatFactory](#).

Definition at line 40 of file CircularMatFactory.cpp.

References `ERR_UNIMPLEMENTED_METHOD`.

#### 6.44.4 Member Data Documentation

**6.44.4.1 const std::string CircularMatFactory::ERR\_UNIMPLEMENTED\_METHOD =**  
"The method you called is not implemented." [static, private]

Definition at line 51 of file CircularMatFactory.hpp.

Referenced by `readValuesFromFile()`.

**6.44.4.2 const int CircularMatFactory::INTENSITY\_MAX = 1** [static, private]

Definition at line 53 of file CircularMatFactory.hpp.

Referenced by `createCircularMaskFromCentreToEdge()`.

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

- CircularMatFactory.hpp
- CircularMatFactory.cpp

### 6.45 multiscale::analysis::Cluster Class Reference

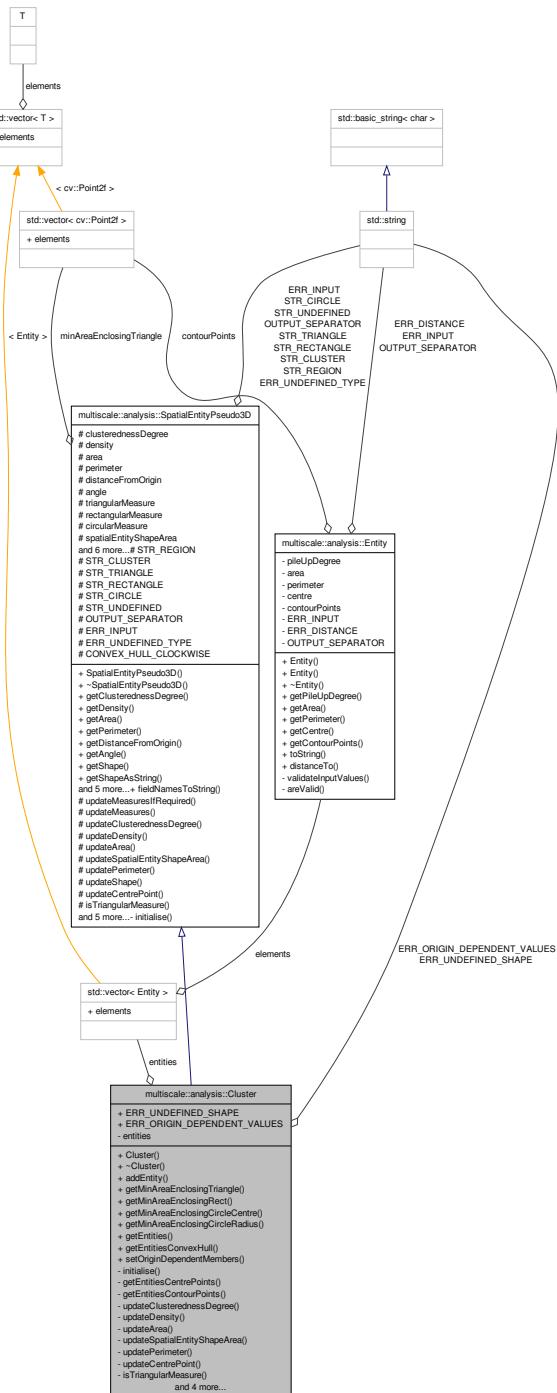
Class for representing a cluster of entities in an image.

```
#include <Cluster.hpp>
```

Inheritance diagram for multiscale::analysis::Cluster:



Collaboration diagram for multiscale::analysis::Cluster:



## Public Member Functions

- `Cluster ()`
- `~Cluster ()`
- `void addEntity (const Entity &entity)`  
`Add a new entity to the cluster.`
- `std::vector< cv::Point2f > getMinAreaEnclosingTriangle ()`  
`Get the minimum area enclosing triangle.`
- `cv::RotatedRect getMinAreaEnclosingRect ()`  
`Get the minimum area enclosing rectangle.`
- `cv::Point2f getMinAreaEnclosingCircleCentre ()`  
`Get the minimum area enclosing circle centre.`
- `float getMinAreaEnclosingCircleRadius ()`  
`Get the minimum area enclosing circle radius.`
- `std::vector< Entity > getEntities () const`  
`Get the collection of underlying entities.`
- `std::vector< cv::Point2f > getEntitiesConvexHull ()`  
`Get the convex hull enclosing the collection of entities' contour points.`
- `void setOriginDependentMembers (double distanceFromOrigin, double angleWrtOrigin)`  
`Set the values of the origin dependent members.`

## Static Public Attributes

- `static const std::string ERR_UNDEFINED_SHAPE = "The shape of the given cluster is undefined."`
- `static const std::string ERR_ORIGIN_DEPENDENT_VALUES = "The origin dependent values are invalid (i.e. negative)."`

## Private Member Functions

- `void initialise ()`  
`Initialisation function for the class.`
- `std::vector< cv::Point2f > getEntitiesCentrePoints ()`  
`Get the collection of entities' centres.`
- `std::vector< cv::Point2f > getEntitiesContourPoints ()`  
`Get the collection of entities' contour points.`
- `void updateClusterednessDegree () override`  
`Update the value of the clusteredness degree.`
- `void updateDensity () override`  
`Update the value of the pile up degree.`
- `void updateArea () override`  
`Update the value of the area.`
- `void updateSpatialEntityShapeArea () override`

- void `updatePerimeter ()` override
  - Update the value of the spatial entity shape.*
- void `updateCentrePoint ()` override
  - Update the value of the perimeter.*
- double `isTriangularMeasure ()` override
  - Get the measure that the cluster has a triangular shape.*
- double `isRectangularMeasure ()` override
  - Get the measure that the cluster has a rectangular shape.*
- double `isCircularMeasure ()` override
  - Get the measure that the cluster has a circular shape.*
- `SpatialEntityPseudo3DType type ()` override
  - Return the type of the pseudo 3D spatial entity.*
- void `validateOriginDependentValues (double distanceFromOrigin, double angleWrtOrigin)`
  - Validate the origin dependent values (i.e. non-negative)*
- bool `isValidOriginDependentValues (double distanceFromOrigin, double angleWrtOrigin)`
  - Check if the origin dependent values are valid (i.e. non-negative)*

## Private Attributes

- `std::vector< Entity > entities`

### 6.45.1 Detailed Description

Class for representing a cluster of entities in an image.

Definition at line 19 of file analysis/spatial/include/multiscale/analysis/spatial/model/-Cluster.hpp.

### 6.45.2 Constructor & Destructor Documentation

#### 6.45.2.1 Cluster::Cluster ( )

Definition at line 11 of file Cluster.cpp.

References `initialise()`.

Referenced by `type()`.

#### 6.45.2.2 Cluster::~Cluster ( )

Definition at line 15 of file Cluster.cpp.

### 6.45.3 Member Function Documentation

#### 6.45.3.1 void Cluster::addEntity ( const Entity & entity )

Add a new entity to the cluster.

Definition at line 17 of file Cluster.cpp.

References entities, and multiscale::analysis::SpatialEntityPseudo3D::updateFlag.

#### 6.45.3.2 bool Cluster::isValidOriginDependentValues ( double *distanceFromOrigin*, double *angleWrtOrigin* ) [private]

Check if the origin dependent values are valid (i.e. non-negative)

##### Parameters

<i>distance-FromOrigin</i>	Distance from the origin
<i>angleWrt-Origin</i>	Angle with respect to the origin

Definition at line 219 of file Cluster.cpp.

References multiscale::Numeric::greaterOrEqual(), and multiscale::Numeric::lessOrEqual().

Referenced by validateOriginDependentValues().

#### 6.45.3.3 std::vector< Entity > Cluster::getEntities ( ) const

Get the collection of underlying entities.

Definition at line 47 of file Cluster.cpp.

References entities.

Referenced by multiscale::analysis::SimulationClusterDetector::outputClusterToImage().

#### 6.45.3.4 std::vector< cv::Point2f > Cluster::getEntitiesCentrePoints ( ) [private]

Get the collection of entities' centres.

Definition at line 87 of file Cluster.cpp.

References entities.

Referenced by getEntitiesConvexHull().

**6.45.3.5 std::vector< cv::Point2f > Cluster::getEntitiesContourPoints( )**  
[private]

Get the collection of entities' contour points.

Definition at line 97 of file Cluster.cpp.

References entities.

**6.45.3.6 std::vector< cv::Point2f > Cluster::getEntitiesConvexHull( )**

Get the convex hull enclosing the collection of entities' contour points.

Definition at line 51 of file Cluster.cpp.

References multiscale::Geometry2D::computeConvexHull(), multiscale::analysis::SpatialEntityPseudo3D::CONVEX\_HULL\_CLOCKWISE, entities, and getEntitiesCentrePoints().

Referenced by isCircularMeasure(), isRectangularMeasure(), isTriangularMeasure(), updateArea(), updateCentrePoint(), updatePerimeter(), and updateSpatialEntityShapeArea().

**6.45.3.7 cv::Point2f Cluster::getMinAreaEnclosingCircleCentre( )**

Get the minimum area enclosing circle centre.

Definition at line 35 of file Cluster.cpp.

References multiscale::analysis::SpatialEntityPseudo3D::minAreaEnclosingCircleCentre, and multiscale::analysis::SpatialEntityPseudo3D::updateMeasuresIfRequired().

Referenced by multiscale::analysis::SimulationClusterDetector::outputClusterCircularShape().

**6.45.3.8 float Cluster::getMinAreaEnclosingCircleRadius( )**

Get the minimum area enclosing circle radius.

Definition at line 41 of file Cluster.cpp.

References multiscale::analysis::SpatialEntityPseudo3D::minAreaEnclosingCircleRadius, and multiscale::analysis::SpatialEntityPseudo3D::updateMeasuresIfRequired().

Referenced by multiscale::analysis::SimulationClusterDetector::outputClusterCircularShape().

**6.45.3.9 cv::RotatedRect Cluster::getMinAreaEnclosingRect( )**

Get the minimum area enclosing rectangle.

Definition at line 29 of file Cluster.cpp.

References `multiscale::analysis::SpatialEntityPseudo3D::minAreaEnclosingRect`, and `multiscale::analysis::SpatialEntityPseudo3D::updateMeasuresIfRequired()`.

Referenced by `multiscale::analysis::SimulationClusterDetector::outputClusterRectangularShape()`.

#### 6.45.3.10 `std::vector< cv::Point2f > Cluster::getMinAreaEnclosingTriangle( )`

Get the minimum area enclosing triangle.

Definition at line 23 of file `Cluster.cpp`.

References `multiscale::analysis::SpatialEntityPseudo3D::minAreaEnclosingTriangle`, and `multiscale::analysis::SpatialEntityPseudo3D::updateMeasuresIfRequired()`.

Referenced by `multiscale::analysis::SimulationClusterDetector::outputClusterTriangularShape()`.

#### 6.45.3.11 `void Cluster::initialise( ) [private]`

Initialisation function for the class.

Reimplemented from [multiscale::analysis::SpatialEntityPseudo3D](#).

Definition at line 72 of file `Cluster.cpp`.

References `multiscale::analysis::SpatialEntityPseudo3D::angle`, `multiscale::analysis::SpatialEntityPseudo3D::clusterednessDegree`, `multiscale::analysis::SpatialEntityPseudo3D::density`, `multiscale::analysis::SpatialEntityPseudo3D::distanceFromOrigin`, `entities`, `multiscale::analysis::SpatialEntityPseudo3D::minAreaEnclosingCircleRadius`, `multiscale::analysis::SpatialEntityPseudo3D::minAreaEnclosingTriangle`, and `multiscale::analysis::SpatialEntityPseudo3D::updateFlag`.

Referenced by `Cluster()`.

#### 6.45.3.12 `double Cluster::isCircularMeasure( ) [override, private, virtual]`

Get the measure that the cluster has a circular shape.

Implements [multiscale::analysis::SpatialEntityPseudo3D](#).

Definition at line 198 of file `Cluster.cpp`.

References `getEntitiesConvexHull()`, `multiscale::analysis::SpatialEntityPseudo3D::minAreaEnclosingCircleCentre`, `multiscale::analysis::SpatialEntityPseudo3D::minAreaEnclosingCircleRadius`, `multiscale::analysis::SpatialEntityPseudo3D::normalisedShapeMeasure()`, and `multiscale::Geometry2D::PI`.

---

**6.45.3.13 double Cluster::isRectangularMeasure( ) [override, private, virtual]**

Get the measure that the cluster has a rectangular shape.

Implements [multiscale::analysis::SpatialEntityPseudo3D](#).

Definition at line 187 of file Cluster.cpp.

References [getEntitiesConvexHull\(\)](#), [multiscale::analysis::SpatialEntityPseudo3D::minAreaEnclosingRect](#), and [multiscale::analysis::SpatialEntityPseudo3D::normalisedShapeMeasure\(\)](#).

**6.45.3.14 double Cluster::isTriangularMeasure( ) [override, private, virtual]**

Get the measure that the cluster has a triangular shape.

Implements [multiscale::analysis::SpatialEntityPseudo3D](#).

Definition at line 171 of file Cluster.cpp.

References [multiscale::MinEnclosingTriangleFinder::find\(\)](#), [getEntitiesConvexHull\(\)](#), [multiscale::analysis::SpatialEntityPseudo3D::minAreaEnclosingTriangle](#), and [multiscale::analysis::SpatialEntityPseudo3D::normalisedShapeMeasure\(\)](#).

**6.45.3.15 void Cluster::setOriginDependentMembers ( double *distanceFromOrigin*, double *angleWrtOrigin* )**

Set the values of the origin dependent members.

#### Parameters

<i>distance-FromOrigin</i>	Distance from the origin
<i>angleWrt-Origin</i>	Angle with respect to the origin

Definition at line 65 of file Cluster.cpp.

References [multiscale::analysis::SpatialEntityPseudo3D::angle](#), [multiscale::analysis::SpatialEntityPseudo3D::distanceFromOrigin](#), and [validateOriginDependentValues\(\)](#).

Referenced by [multiscale::analysis::ClusterDetector::updateClusterOriginDependentValues\(\)](#).

**6.45.3.16 SpatialEntityPseudo3DType Cluster::type( ) [override, private, virtual]**

Return the type of the pseudo 3D spatial entity.

Implements [multiscale::analysis::SpatialEntityPseudo3D](#).

Definition at line 209 of file Cluster.cpp.

References Cluster().

#### 6.45.3.17 void Cluster::updateArea( ) [override, private, virtual]

Update the value of the area.

Implements [multiscale::analysis::SpatialEntityPseudo3D](#).

Definition at line 139 of file Cluster.cpp.

References [multiscale::analysis::SpatialEntityPseudo3D::area](#), [multiscale::analysis::SpatialMeasureCalculator::computePolygonArea\(\)](#), and [getEntitiesConvexHull\(\)](#).

#### 6.45.3.18 void Cluster::updateCentrePoint( ) [override, private, virtual]

Update the point defining the centre of the cluster.

Implements [multiscale::analysis::SpatialEntityPseudo3D](#).

Definition at line 165 of file Cluster.cpp.

References [multiscale::analysis::SpatialEntityPseudo3D::centre](#), [multiscale::Geometry2D::centroid\(\)](#), and [getEntitiesConvexHull\(\)](#).

#### 6.45.3.19 void Cluster::updateClusterednessDegree( ) [override, private, virtual]

Update the value of the clusteredness degree.

Implements [multiscale::analysis::SpatialEntityPseudo3D](#).

Definition at line 109 of file Cluster.cpp.

References [multiscale::analysis::SpatialEntityPseudo3D::clusterednessDegree](#), [multiscale::Numeric::division\(\)](#), and [entities](#).

#### 6.45.3.20 void Cluster::updateDensity( ) [override, private, virtual]

Update the value of the pile up degree.

Implements [multiscale::analysis::SpatialEntityPseudo3D](#).

Definition at line 129 of file Cluster.cpp.

References [multiscale::analysis::SpatialEntityPseudo3D::density](#), [multiscale::Numeric::division\(\)](#), and [entities](#).

---

**6.45.3.21 void Cluster::updatePerimeter( ) [override, private, virtual]**

Update the value of the perimeter.

Implements [multiscale::analysis::SpatialEntityPseudo3D](#).

Definition at line 155 of file Cluster.cpp.

References multiscale::analysis::SpatialMeasureCalculator::computePolygonPerimeter(), getEntitiesConvexHull(), and multiscale::analysis::SpatialEntityPseudo3D::perimeter.

**6.45.3.22 void Cluster::updateSpatialEntityShapeArea( ) [override, private, virtual]**

Update the value of the spatial entity shape.

Implements [multiscale::analysis::SpatialEntityPseudo3D](#).

Definition at line 149 of file Cluster.cpp.

References getEntitiesConvexHull(), and multiscale::analysis::SpatialEntityPseudo3D::spatialEntityShapeArea.

**6.45.3.23 void Cluster::validateOriginDependentValues( double *distanceFromOrigin*, double *angleWrtOrigin* ) [private]**

Validate the origin dependent values (i.e. non-negative)

#### Parameters

<i>distance-FromOrigin</i>	Distance from the origin
<i>angleWrt-Origin</i>	Angle with respect to the origin

Definition at line 213 of file Cluster.cpp.

References areValidOriginDependentValues(), and ERR\_ORIGIN\_DEPENDENT\_VALUES.

Referenced by setOriginDependentMembers().

## 6.45.4 Member Data Documentation

**6.45.4.1 std::vector<Entity> multiscale::analysis::Cluster::entities [private]**

Entities which belong to this cluster

Definition at line 24 of file analysis/spatial/include/multiscale/analysis/spatial/model/Cluster.hpp.

Referenced by addEntity(), getEntities(), getEntitiesCentrePoints(), getEntitiesContourPoints(), getEntitiesConvexHull(), initialise(), updateClusterednessDegree(), and updateDensity().

#### 6.45.4.2 const std::string Cluster::ERR\_ORIGIN\_DEPENDENT\_VALUES = "The origin dependent values are invalid (i.e. negative)." [static]

Definition at line 118 of file analysis/spatial/include/multiscale/analysis/spatial/model/-Cluster.hpp.

Referenced by validateOriginDependentValues().

#### 6.45.4.3 const std::string Cluster::ERR\_UNDEFINED\_SHAPE = "The shape of the given cluster is undefined." [static]

Definition at line 117 of file analysis/spatial/include/multiscale/analysis/spatial/model/-Cluster.hpp.

Referenced by multiscale::analysis::SimulationClusterDetector::outputClusterShape().

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

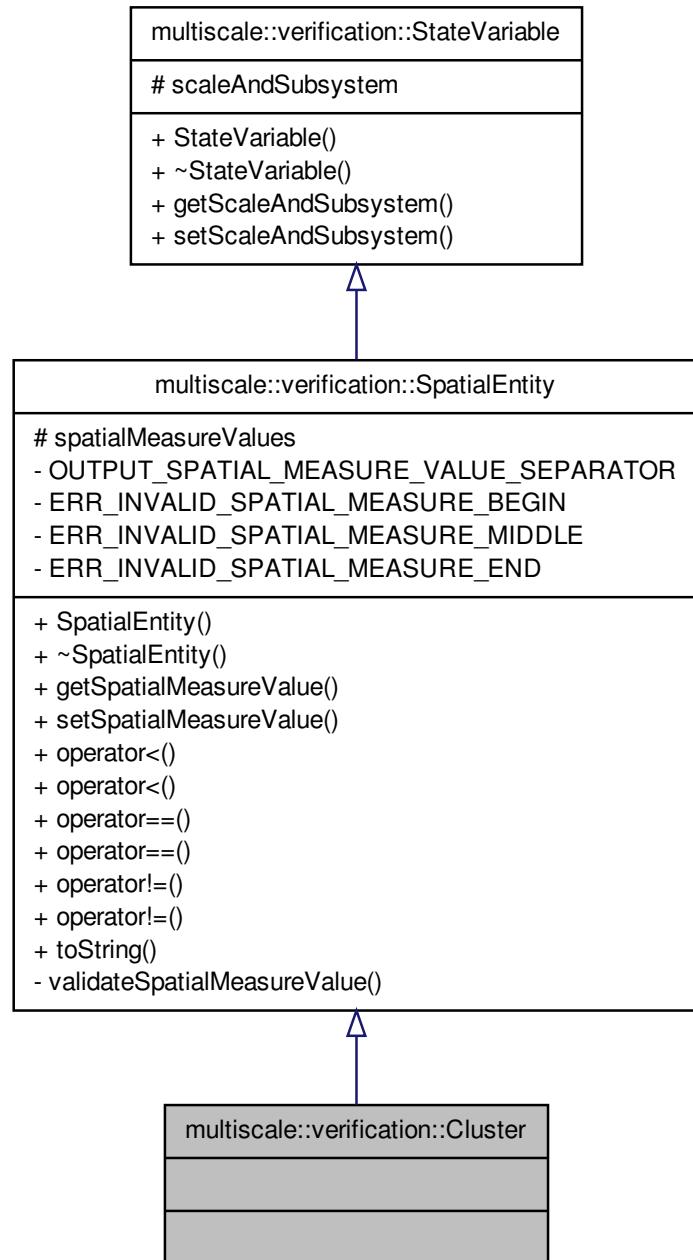
- analysis/spatial/include/multiscale/analysis/spatial/model/Cluster.hpp
- Cluster.cpp

## 6.46 multiscale::verification::Cluster Class Reference

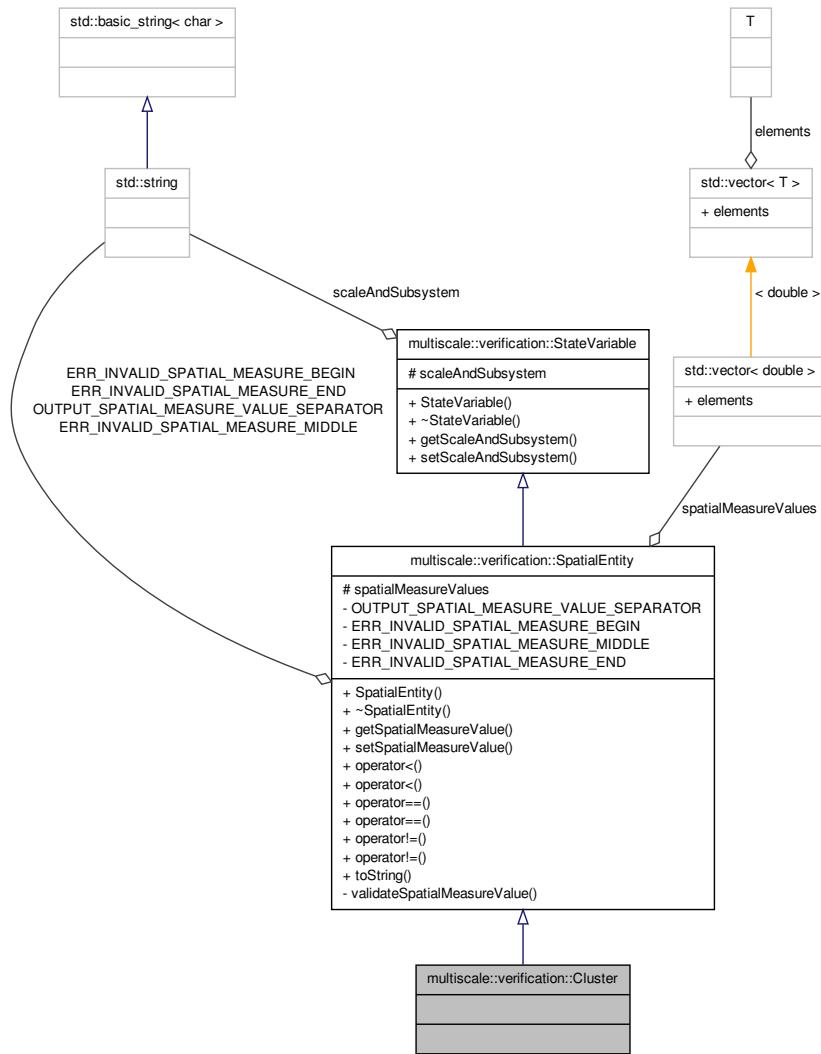
Class for representing a cluster.

```
#include <Cluster.hpp>
```

Inheritance diagram for multiscale::verification::Cluster:



Collaboration diagram for multiscale::verification::Cluster:



### 6.46.1 Detailed Description

Class for representing a cluster.

Definition at line 21 of file verification/spatial-temporal/include/multiscale/verification/spatial-temporal/model/Cluster.hpp.

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

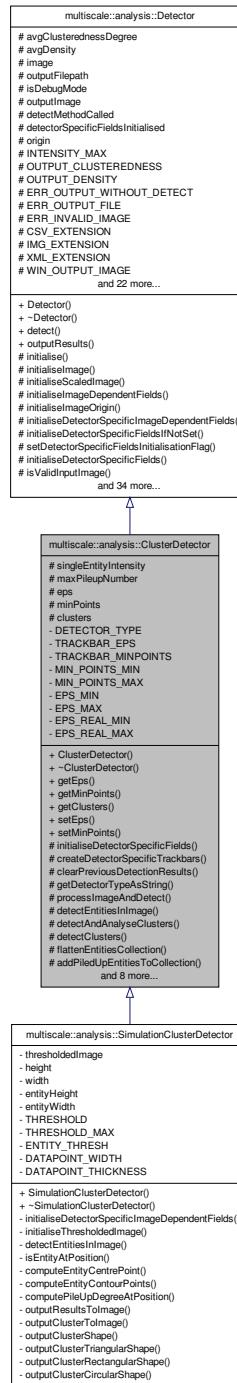
- verification/spatial-temporal/include/multiscale/verification/spatial-temporal/model/-Cluster.hpp

## 6.47 multiscale::analysis::ClusterDetector Class Reference

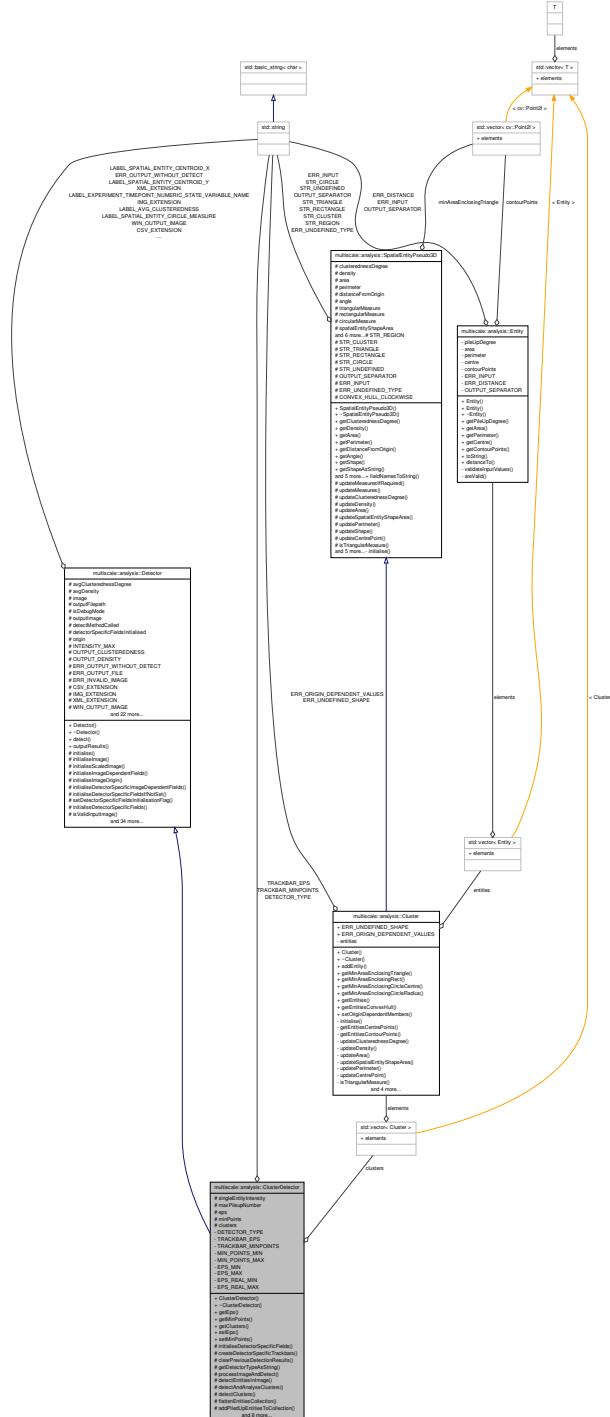
Class for detecting clusters in 2D images.

```
#include <ClusterDetector.hpp>
```

Inheritance diagram for multiscale::analysis::ClusterDetector:



## Collaboration diagram for multiscale::analysis::ClusterDetector:



## Public Member Functions

- `ClusterDetector` (`unsigned int maxPileupNumber, bool isDebugEnabled=false`)  
`virtual ~ClusterDetector ()`  
`double getEps ()`  
    *Get the value of the clustering algorithm parameter eps.*
- `int getMinPoints ()`  
    *Get the value of the clustering algorithm parameter MinPoints.*
- `std::vector< Cluster > const & getClusters ()`  
    *Get a const reference to the vector of detected clusters.*
- `void setEps (double eps)`  
    *Set the value of the clustering algorithm parameter eps.*
- `void setMinPoints (int minPoints)`  
    *Set the value of the clustering algorithm parameter MinPoints.*

## Protected Member Functions

- `void initialiseDetectorSpecificFields () override`  
    *Initialise clustering values.*
- `void createDetectorSpecificTrackbars () override`  
    *Create the trackbars.*
- `void clearPreviousDetectionResults () override`  
    *Clear the clusters from the previous detection.*
- `std::string getDetectorTypeAsString () override`  
    *Get the type of the detector as a string.*
- `void processImageAndDetect () override`  
    *Process the provided image and detect clusters in it.*
- `virtual void detectEntitiesInImage (std::vector< Entity > &entities)=0`  
    *Detect the entities in the image.*
- `void detectAndAnalyseClusters (const std::vector< Entity > &entities, std::vector< Cluster > &clusters)`  
    *Detect and analyse the clusters of entities in the image.*
- `void detectClusters (const std::vector< Entity > &entities, std::vector< int > &clusterIndexes, std::size_t &nrOfClusters)`  
    *Detect the clusters of entities in the image.*
- `std::vector< Entity > flattenEntitiesCollection (const std::vector< Entity > &entities)`  
    *Flatten the collection of entities.*
- `void addPiledUpEntitiesToCollection (const std::vector< Entity > &entities, std::vector< Entity > &flattenedEntities)`  
    *Add the piled up entities to the flattened collection of entities.*
- `void addEntitiesToClusters (const std::vector< Entity > &entities, const std::vector< int > &clusterIndexes, std::size_t nrOfClusters, std::vector< Cluster > &clusters)`

*Add the entities to the clusters as indicated by the clusterIndexes parameter.*

- void `analyseClusters` (`std::vector< Cluster > &clusters`)
 

*Analyse the clusters.*
- void `analyseClustersOriginDependentValues` (`std::vector< Cluster > &clusters`)
 

*Analyse the clusters and compute the origin dependent values.*
- void `updateClusterOriginDependentValues` (`Cluster &cluster, const std::vector< cv::Point2f > &clusterConvexHull`)
 

*Update the cluster and compute the origin dependent values considering the convex hull.*
- double `computeClusterednessIndex` (`const std::vector< Cluster > &clusters`)
 

*Compute the clusteredness index for all the entities detected in the image.*
- double `computeAveragePileUpDegree` (`std::vector< Cluster > &clusters`)
 

*Compute the average pile up degree for all entities in the image.*
- `std::vector< std::shared_ptr < SpatialEntityPseudo3D > > getCollectionOfSpatialEntityPseudo3D () override`

*Get the collection of clusters detected in the image.*
- double `convertEpsValue ()`

*Convert the value of eps from integer to double.*
- int `getValidMinPointsValue ()`

*Return non-zero value for minPoints.*

### Protected Attributes

- double `singleEntityIntensity`
- unsigned int `maxPileupNumber`
- int `eps`
- int `minPoints`
- `std::vector< Cluster > clusters`

### Static Private Attributes

- static const `std::string DETECTOR_TYPE` = "Clusters"
- static const `std::string TRACKBAR_EPS` = "Eps (Multiplied by 10)"
- static const `std::string TRACKBAR_MINPOINTS` = "Minimum number of points"
- static const `int MIN_POINTS_MIN` = 0
- static const `int MIN_POINTS_MAX` = 100
- static const `int EPS_MIN` = 0
- static const `int EPS_MAX` = 10000
- static const `int EPS_REAL_MIN` = 0
- static const `int EPS_REAL_MAX` = 1000

### 6.47.1 Detailed Description

Class for detecting clusters in 2D images.

Definition at line 17 of file ClusterDetector.hpp.

### 6.47.2 Constructor & Destructor Documentation

**6.47.2.1 ClusterDetector::ClusterDetector ( *unsigned int maxPileupNumber, bool isDebugEnabled = false* )**

Definition at line 15 of file ClusterDetector.cpp.

References multiscale::analysis::Detector::avgClusterednessDegree, multiscale::analysis::Detector::avgDensity, multiscale::Numeric::division(), eps, multiscale::analysis::Detector::INTENSITY\_MAX, minPoints, and singleEntityIntensity.

**6.47.2.2 ClusterDetector::~ClusterDetector ( ) [virtual]**

Definition at line 31 of file ClusterDetector.cpp.

### 6.47.3 Member Function Documentation

**6.47.3.1 void ClusterDetector::addEntitiesToClusters ( *const std::vector< Entity > & entities, const std::vector< int > & clusterIndexes, std::size\_t nrOfClusters, std::vector< Cluster > & clusters* ) [protected]**

Add the entities to the clusters as indicated by the clusterIndexes parameter.

Add the entities to the clusters as indicated by the clusterIndexes parameter

The "noise" cluster will be ignored.

#### Parameters

<i>entities</i>	Entities detected in the image
<i>cluster-Indexes</i>	Indexes to which cluster each entity belongs
<i>nrOfClusters</i>	Total number of clusters
<i>clusters</i>	Collection of clusters, each one with the updated measures

Definition at line 128 of file ClusterDetector.cpp.

Referenced by detectAndAnalyseClusters().

**6.47.3.2 void ClusterDetector::addPiledUpEntitiesToCollection ( *const std::vector< Entity > & entities, std::vector< Entity > & flattenedEntities* ) [protected]**

Add the piled up entities to the flattened collection of entities.

**Parameters**

<i>entities</i>	Entities detected in the image
<i>flattened- Entities</i>	Flattened collection of entities

Definition at line 116 of file ClusterDetector.cpp.

Referenced by flattenEntitiesCollection().

**6.47.3.3 void ClusterDetector::analyseClusters ( std::vector< Cluster > & *clusters* ) [protected]**

Analyse the clusters.

Analyse the clusters and compute the angle and distance from the centre, average clusteredness degree and pile up degree

**Parameters**

<i>clusters</i>	Collection of clusters, each one with the updated measures
-----------------	--

Definition at line 147 of file ClusterDetector.cpp.

References analyseClustersOriginDependentValues(), multiscale::analysis::Detector::avgClusterednessDegree, multiscale::analysis::Detector::avgDensity, computeAveragePileUpDegree(), and computeClusterednessIndex().

Referenced by detectAndAnalyseClusters().

**6.47.3.4 void ClusterDetector::analyseClustersOriginDependentValues ( std::vector< Cluster > & *clusters* ) [protected]**

Analyse the clusters and compute the origin dependent values.

The values which depend on the origin point are the distance of the cluster from the centre and the angle

**Parameters**

<i>clusters</i>	Collection of clusters, each one with the updated measures
-----------------	--

Definition at line 154 of file ClusterDetector.cpp.

References updateClusterOriginDependentValues().

Referenced by analyseClusters().

**6.47.3.5 void ClusterDetector::clearPreviousDetectionResults ( ) [override, protected, virtual]**

Clear the clusters from the previous detection.

Implements [multiscale::analysis::Detector](#).

Definition at line 69 of file ClusterDetector.cpp.

References clusters.

#### 6.47.3.6 double ClusterDetector::computeAveragePileUpDegree ( std::vector< Cluster > & *clusters* ) [protected]

Compute the average pile up degree for all entities in the image.

Compute the average pile up degree for all entities in the image as the sum of the average pile up degrees of all clusters divided by the number of clusters

##### Parameters

<i>clusters</i>	Clusters of entities detected in the image
-----------------	--

Definition at line 178 of file ClusterDetector.cpp.

References multiscale::Numeric::division().

Referenced by analyseClusters().

#### 6.47.3.7 double ClusterDetector::computeClusterednessIndex ( const std::vector< Cluster > & *clusters* ) [protected]

Compute the clusteredness index for all the entities detected in the image.

Compute the clusteredness index for all the entities detected in the image using - [Silhouette](#) cluster validity index

##### Parameters

<i>clusters</i>	Collection of clusters, each one with the updated measures
-----------------	--

Definition at line 173 of file ClusterDetector.cpp.

References multiscale::analysis::Silhouette::computeOverallAverageMeasure().

Referenced by analyseClusters().

#### 6.47.3.8 double ClusterDetector::convertEpsValue ( ) [protected]

Convert the value of eps from integer to double.

Definition at line 203 of file ClusterDetector.cpp.

References eps, EPS\_MAX, EPS\_MIN, EPS\_REAL\_MAX, and EPS\_REAL\_MIN.

Referenced by detectClusters(), and getEps().

---

**6.47.3.9 void ClusterDetector::createDetectorSpecificTrackbars ( )**  
[override, protected, virtual]

Create the trackbars.

Implements [multiscale::analysis::Detector](#).

Definition at line 64 of file ClusterDetector.cpp.

References eps, EPS\_MAX, MIN\_POINTS\_MAX, minPoints, TRACKBAR\_EPS, TRACKBAR\_MINPOINTS, and [multiscale::analysis::Detector::WIN\\_OUTPUT\\_IMAGE](#).

**6.47.3.10 void ClusterDetector::detectAndAnalyseClusters ( const std::vector< Entity > & entities, std::vector< Cluster > & clusters ) [protected]**

Detect and analyse the clusters of entities in the image.

Detect and analyse the clusters of entities in the image

Remark: The "noise" cluster will be ignored.

#### Parameters

<i>entities</i>	Entities detected in the image
<i>clusters</i>	Clusters of entities detected in the image

Definition at line 84 of file ClusterDetector.cpp.

References [addEntitiesToClusters\(\)](#), [analyseClusters\(\)](#), and [detectClusters\(\)](#).

Referenced by [processImageAndDetect\(\)](#).

**6.47.3.11 void ClusterDetector::detectClusters ( const std::vector< Entity > & entities, std::vector< int > & clusterIndexes, std::size\_t & nrOfClusters ) [protected]**

Detect the clusters of entities in the image.

Detect the clusters of entities in the image using Density Based scan (DBscan) clustering algorithm Clusters start from index 1, because cluster 0 contains only noise data-points.

#### Parameters

<i>entities</i>	Entities detected in the image
<i>cluster-Indexes</i>	Indexes to which cluster each entity belongs
<i>nrOfClusters</i>	Total number of clusters

Definition at line 94 of file ClusterDetector.cpp.

References [convertEpsValue\(\)](#), [flattenEntitiesCollection\(\)](#), and [getValidMinPointsValue\(\)](#).

Referenced by [detectAndAnalyseClusters\(\)](#).

6.47.3.12 `virtual void multiscale::analysis::ClusterDetector::detectEntitiesInImage ( std::vector< Entity > & entities )` [protected, pure virtual]

Detect the entities in the image.

Detect the entities in the image, compute their centre point and degree of pile up

Parameters

<code>entities</code>	Entities detected in the image
-----------------------	--------------------------------

Implemented in [multiscale::analysis::SimulationClusterDetector](#).

Referenced by `processImageAndDetect()`.

6.47.3.13 `std::vector< Entity > ClusterDetector::flattenEntitiesCollection ( const std::vector< Entity > & entities )` [protected]

Flatten the collection of entities.

Add each piled up entity in the original collection as a single standing entity in the resulting collection. The first entities in the resulting collection are the entities in the given collection. All piled up entities are added afterwards in the resulting collection.

Parameters

<code>entities</code>	Entities detected in the image
-----------------------	--------------------------------

Definition at line 106 of file `ClusterDetector.cpp`.

References `addPiledUpEntitiesToCollection()`.

Referenced by `detectClusters()`.

6.47.3.14 `std::vector< Cluster > const & ClusterDetector::getClusters ( )`

Get a const reference to the vector of detected clusters.

Definition at line 41 of file `ClusterDetector.cpp`.

References `clusters`.

6.47.3.15 `std::vector< std::shared_ptr< SpatialEntityPseudo3D > > ClusterDetector::getCollectionOfSpatialEntityPseudo3D ( )` [override, protected, virtual]

Get the collection of clusters detected in the image.

Implements [multiscale::analysis::Detector](#).

Definition at line 193 of file `ClusterDetector.cpp`.

References `multiscale::analysis::Cluster`, and `clusters`.

6.47.3.16 `std::string ClusterDetector::getDetectorTypeAsString( )` [override, protected, virtual]

Get the type of the detector as a string.

Implements [multiscale::analysis::Detector](#).

Definition at line 73 of file ClusterDetector.cpp.

References DETECTOR\_TYPE.

6.47.3.17 `double ClusterDetector::getEps( )`

Get the value of the clustering algorithm parameter eps.

Definition at line 33 of file ClusterDetector.cpp.

References convertEpsValue().

6.47.3.18 `int ClusterDetector::getMinPoints( )`

Get the value of the clustering algorithm parameter MinPoints.

Definition at line 37 of file ClusterDetector.cpp.

References minPoints.

6.47.3.19 `int ClusterDetector::getValidMinPointsValue( )` [protected]

Return non-zero value for minPoints.

Definition at line 211 of file ClusterDetector.cpp.

References minPoints.

Referenced by detectClusters().

6.47.3.20 `void ClusterDetector::initialiseDetectorSpecificFields( )` [override, protected, virtual]

Initialise clustering values.

Implements [multiscale::analysis::Detector](#).

Definition at line 59 of file ClusterDetector.cpp.

References eps, and minPoints.

6.47.3.21 `void ClusterDetector::processImageAndDetect( )` [override, protected, virtual]

Process the provided image and detect clusters in it.

Implements [multiscale::analysis::Detector](#).

Definition at line 77 of file ClusterDetector.cpp.

References clusters, detectAndAnalyseClusters(), and detectEntitiesInImage().

#### 6.47.3.22 void ClusterDetector::setEps ( double eps )

Set the value of the clustering algorithm parameter eps.

##### Parameters

<code>eps</code>	Value of the clustering algorithm parameter eps
------------------	---

Definition at line 45 of file ClusterDetector.cpp.

References `eps`, EPS\_MAX, EPS\_MIN, EPS\_REAL\_MAX, EPS\_REAL\_MIN, and [multiscale::analysis::Detector::setDetectorSpecificFieldsInitialisationFlag\(\)](#).

#### 6.47.3.23 void ClusterDetector::setMinPoints ( int minPoints )

Set the value of the clustering algorithm parameter MinPoints.

##### Parameters

<code>minPoints</code>	Value of the clustering algorithm parameter MinPoints
------------------------	---

Definition at line 53 of file ClusterDetector.cpp.

References `minPoints`, and [multiscale::analysis::Detector::setDetectorSpecificFieldsInitialisationFlag\(\)](#).

#### 6.47.3.24 void ClusterDetector::updateClusterOriginDependentValues ( Cluster & cluster, const std::vector< cv::Point2f > & clusterConvexHull ) [protected]

Update the cluster and compute the origin dependent values considering the convex hull.

The values which depend on the origin point are the distance of the cluster from the centre and the angle

##### Parameters

<code>cluster</code>	<a href="#">Cluster</a>
<code>cluster-ConvexHull</code>	Convex hull of the cluster

Definition at line 164 of file ClusterDetector.cpp.

References [multiscale::analysis::Detector::computeDistanceFromOrigin\(\)](#), [multiscale::analysis::Detector::computePolygonAngle\(\)](#), and [multiscale::analysis::Cluster::set-](#)

OriginDependentMembers().

Referenced by analyseClustersOriginDependentValues().

#### 6.47.4 Member Data Documentation

**6.47.4.1 std::vector<Cluster> multiscale::analysis::ClusterDetector::clusters [protected]**

Clusters found in the image

Definition at line 32 of file ClusterDetector.hpp.

Referenced by clearPreviousDetectionResults(), getClusters(), getCollectionOfSpatialEntityPseudo3D(), multiscale::analysis::SimulationClusterDetector::outputResultsToImage(), and processImageAndDetect().

**6.47.4.2 const std::string ClusterDetector::DETECTOR\_TYPE = "Clusters" [static, private]**

Definition at line 195 of file ClusterDetector.hpp.

Referenced by getDetectorTypeAsString().

**6.47.4.3 int multiscale::analysis::ClusterDetector::eps [protected]**

**DBSCAN** algorithm parameter for specifying the maximum radius of the neighbourhood

Definition at line 27 of file ClusterDetector.hpp.

Referenced by ClusterDetector(), convertEpsValue(), createDetectorSpecificTrackbars(), initialiseDetectorSpecificFields(), and setEps().

**6.47.4.4 const int ClusterDetector::EPS\_MAX = 10000 [static, private]**

Definition at line 204 of file ClusterDetector.hpp.

Referenced by convertEpsValue(), createDetectorSpecificTrackbars(), and setEps().

**6.47.4.5 const int ClusterDetector::EPS\_MIN = 0 [static, private]**

Definition at line 203 of file ClusterDetector.hpp.

Referenced by convertEpsValue(), and setEps().

**6.47.4.6 const int ClusterDetector::EPS\_REAL\_MAX = 1000 [static, private]**

Definition at line 206 of file ClusterDetector.hpp.

Referenced by convertEpsValue(), and setEps().

**6.47.4.7 const int ClusterDetector::EPS\_REAL\_MIN = 0 [static, private]**

Definition at line 205 of file ClusterDetector.hpp.

Referenced by convertEpsValue(), and setEps().

**6.47.4.8 unsigned int multiscale::analysis::ClusterDetector::maxPileupNumber [protected]**

The maximum number of entities which can occupy the same grid position

Definition at line 24 of file ClusterDetector.hpp.

Referenced by multiscale::analysis::SimulationClusterDetector::computePileUpDegreeAtPosition().

**6.47.4.9 const int ClusterDetector::MIN\_POINTS\_MAX = 100 [static, private]**

Definition at line 201 of file ClusterDetector.hpp.

Referenced by createDetectorSpecificTrackbars().

**6.47.4.10 const int ClusterDetector::MIN\_POINTS\_MIN = 0 [static, private]**

Definition at line 200 of file ClusterDetector.hpp.

**6.47.4.11 int multiscale::analysis::ClusterDetector::minPoints [protected]**

**DBSCAN** algorithm parameter for specifying the minimum number of points in an epsilon-neighbourhood of that point

Definition at line 29 of file ClusterDetector.hpp.

Referenced by ClusterDetector(), createDetectorSpecificTrackbars(), getMinPoints(), getValidMinPointsValue(), initialiseDetectorSpecificFields(), and setMinPoints().

**6.47.4.12 double multiscale::analysis::ClusterDetector::singleEntityIntensity [protected]**

The intensity (pile up degree) of a grid position occupied by a single entity

Definition at line 21 of file ClusterDetector.hpp.

Referenced by ClusterDetector(), and multiscale::analysis::SimulationClusterDetector::computePileUpDegreeAtPosition().

---

6.47.4.13 `const std::string ClusterDetector::TRACKBAR_EPS = "Eps (Multiplied by 10)"`  
[static, private]

Definition at line 197 of file ClusterDetector.hpp.

Referenced by `createDetectorSpecificTrackbars()`.

6.47.4.14 `const std::string ClusterDetector::TRACKBAR_MINPOINTS = "Minimum number of points"` [static, private]

Definition at line 198 of file ClusterDetector.hpp.

Referenced by `createDetectorSpecificTrackbars()`.

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

- ClusterDetector.hpp
  
- ClusterDetector.cpp

## 6.48 multiscale::verification::CommandLineModelChecking Class - Reference

Class for running model checkers from the command line.

```
#include <CommandLineModelChecking.hpp>
```

## 6.48 multiscale::verification::CommandLineModelChecking Class Reference 247

Collaboration diagram for multiscale::verification::CommandLineModelChecking:



## Public Member Functions

- `CommandLineModelChecking ()`
- `~CommandLineModelChecking ()`
- `void initialise (int argc, char **argv)`

*Initialise the class with the given command line arguments.*
- `void execute ()`

*Execute the model checking task.*

## Private Member Functions

- `bool areValidArguments (int argc, char **argv)`

*Check if the provided command line arguments are valid.*
- `void initialiseAllowedArgumentsConfiguration ()`

*Initialise the configuration of allowed command line arguments.*
- `void initialiseRequiredArgumentsConfiguration ()`

*Initialise the configuration of required command line arguments.*
- `void initialiseOptionalArgumentsConfiguration ()`

*Initialise the configuration of optional command line arguments.*
- `void initialiseModelCheckerTypeSpecificArgumentsConfiguration ()`

*Initialise the configuration of model checker type specific command line arguments.*
- `po::options_description initialiseStatisticalModelCheckerArgumentsConfiguration ()`

*Initialise the configuration of the statistical model checker command line arguments.*
- `po::options_description initialiseApproximateProbabilisticModelCheckerArgumentsConfiguration ()`

*Initialise the configuration of the approximate probabilistic model checker command line arguments.*
- `po::options_description initialiseBayesianModelCheckerArgumentsConfiguration ()`

*Initialise the configuration of the Bayesian model checker command line arguments.*
- `po::options_description initialiseApproximateBayesianModelCheckerArgumentsConfiguration ()`

*Initialise the configuration of the approximate Bayesian model checker command line arguments.*
- `bool areValidArgumentsConsideringConfiguration (int argc, char **argv)`

*Check if the provided command line arguments are valid.*
- `po::parsed_options parseAndStoreArgumentsValues (int argc, char **argv)`

*Parse and store the command line arguments' values in a variables map.*
- `bool areInvalidExecutionArguments (const po::parsed_options &parsedArguments)`

*Check if any invalid execution arguments were provided.*
- `bool isHelpArgumentPresent ()`

*Check if the help command line argument is present.*

- void `handleHelpRequest ()`  
*Handle the help request i.e. if the --help flag was provided.*
- void `printHelpMessage ()`  
*Print help message to the console.*
- void `printHelpIntroMessage ()`  
*Print the help intro message to the console.*
- void `printHelpContentsMessage ()`  
*Print the help contents message to the console.*
- void `printHelpClosingMessage ()`  
*Print the help closing message to the console.*
- bool `areUnrecognizedArgumentsPresent (const po::parsed_options &parsedArguments)`  
*Check if any unrecognized command line arguments are present.*
- bool `areInvalidModelCheckingArguments ()`  
*Check if any invalid model checker type dependent arguments are present.*
- bool `areInvalidModelCheckingArgumentsPresent ()`  
*Check if any model checker type dependent arguments are invalid.*
- void `removeRequiredArguments (po::variables_map &variablesMap)`  
*Remove the required arguments from the given variables\_map.*
- void `removeOptionalArguments (po::variables_map &variablesMap)`  
*Remove the optional arguments from the given variables\_map.*
- bool `areInvalidModelCheckingTypeSpecificArguments (unsigned int modelCheckerType, po::variables_map &variablesMap)`  
*Check if the model checking type specific arguments from the given variables\_map are invalid.*
- bool `areModelCheckingTypeSpecificArgumentsPresent (unsigned int modelCheckerType, const po::variables_map &variablesMap)`  
*Check if all model checking type specific arguments are present.*
- bool `areStatisticalModelCheckingArgumentsPresent (const po::variables_map &variablesMap)`  
*Check if the arguments specific to statistical model checking are present.*
- bool `areApproximateProbabilisticModelCheckingArgumentsPresent (const po::variables_map &variablesMap)`  
*Check if the arguments specific to approximate probabilistic model checking are present.*
- bool `areBayesianModelCheckingArgumentsPresent (const po::variables_map &variablesMap)`  
*Check if the arguments specific to Bayesian model checking are present.*
- bool `areApproximateBayesianModelCheckingArgumentsPresent (const po::variables_map &variablesMap)`  
*Check if the arguments specific to approximate Bayesian model checking are present.*
- void `removeModelCheckingTypeSpecificArguments (unsigned int modelCheckerType, po::variables_map &variablesMap)`  
*Remove the model checking type specific arguments from the given variables\_map.*

- void `removeStatisticalModelCheckingArguments` (po::variables\_map &variables-Map)  
*Remove the statistical model checking arguments from the given variables\_map.*
- void `removeApproximateProbabilisticModelCheckingArguments` (po::variables\_map &variablesMap)  
*Remove the approximate probabilistic model checking arguments from the given variables\_map.*
- void `removeBayesianModelCheckingArguments` (po::variables\_map &variables-Map)  
*Remove the Bayesian model checking arguments from the given variables\_map.*
- void `removeApproximateBayesianModelCheckingArguments` (po::variables\_map &variablesMap)  
*Remove the approximate Bayesian model checking arguments from the given variables\_map.*
- void `initialiseClassMembers` ()  
*Initialise the class members using the command line arguments.*
- void `initialiseRequiredArgumentsDependentClassMembers` ()  
*Initialise the class members dependent on required command line arguments.*
- void `initialiseOptionalArgumentsDependentClassMembers` ()  
*Initialise the class members dependent on optional command line arguments.*
- void `initialiseModelCheckerTypeDependentClassMembers` ()  
*Initialise the class members dependent on the model checker type.*
- void `initialiseModelChecker` ()  
*Initialise the model checker.*
- void `initialiseProbabilisticBlackBoxModelChecker` ()  
*Initialise the probabilistic black box model checker.*
- void `initialiseStatisticalModelChecker` ()  
*Initialise the statistical model checker.*
- void `initialiseApproximateProbabilisticModelChecker` ()  
*Initialise the approximate probabilistic model checker.*
- void `initialiseBayesianModelChecker` ()  
*Initialise the Bayesian model checker.*
- void `initialiseApproximateBayesianModelChecker` ()  
*Initialise the approximate Bayesian model checker.*
- void `initialiseModelCheckingManager` ()  
*Initialise the model checking manager.*
- void `printModelCheckingInitialisationMessage` ()  
*Print the model checking initialisation message.*

### Private Attributes

- std::string logicQueriesFilepath
- std::string tracesFolderPath
- unsigned int modelCheckerType
- unsigned long extraEvaluationTime
- std::string extraEvaluationProgramPath
- std::string multiscaleArchitectureGraphFilepath
- bool shouldVerboseDetailedResults
- po::variables\_map variablesMap
- po::options\_description allowedArguments
- po::options\_description requiredArguments
- po::options\_description optionalArguments
- po::options\_description modelCheckerTypeSpecificArguments
- std::string modelCheckerTypeName
- std::string modelCheckerParameters
- std::shared\_ptr < ModelCheckerFactory > modelCheckerFactory
- std::shared\_ptr < ModelCheckingManager > modelCheckingManager

### Static Private Attributes

- static const std::string **ERR\_INVALID\_COMMAND\_LINE\_ARGUMENTS** = "- Invalid command line arguments were provided and the model checker execution was stopped."
- static const std::string **ERR\_INVALID\_MODEL\_CHECKING\_ARGUMENTS** = "- The command line arguments provided for the chosen model checking type are invalid. Please run Mule with the --help flag to determine which arguments you should use."
- static const std::string **ERR\_INVALID\_MODEL\_CHECKING\_TYPE** = "The provided model checking type is invalid. Please run Mule with the --help flag to determine which values you can use."
- static const std::string **ARG\_LOGIC\_QUERIES\_NAME\_LONG** = "logic-queries"
- static const std::string **ARG\_LOGIC\_QUERIES\_NAME\_BOTH** = ",q"
- static const std::string **ARG\_LOGIC\_QUERIES\_DESCRIPTION** = "the path to the spatio-temporal queries input file"
- static const std::string **ARG\_SPATIAL\_TEMPORAL\_TRACES\_NAME\_LONG** = "spatial-temporal-traces"
- static const std::string **ARG\_SPATIAL\_TEMPORAL\_TRACES\_NAME\_BOTH** = ",t"
- static const std::string **ARG\_SPATIAL\_TEMPORAL\_TRACES\_DESCRIPTION** = "the path to the folder containing spatio-temporal traces"
- static const std::string **ARG\_EXTRA\_EVALUATION\_TIME\_NAME\_LONG** = "extra-evaluation-time"
- static const std::string **ARG\_EXTRA\_EVALUATION\_TIME\_NAME\_BOTH** = ",e"
- static const std::string **ARG\_EXTRA\_EVALUATION\_TIME\_DESCRIPTION** = "the maximum number of minutes the application can wait before finishing evaluation"

- static const std::string `ARG_MODEL_CHECKER_TYPE_NAME_LONG` = "model-checker-type"
- static const std::string `ARG_MODEL_CHECKER_TYPE_NAME_BOTH` = ",m"
- static const std::string `ARG_MODEL_CHECKER_TYPE_DESCRIPTION` = "the type of the model checker (0 = Probabilistic black-box, 1 = Frequentist statistical, 2 = Frequentist approximate probabilistic (Chernoff-Hoeffding), 3 = Bayesian (statistical hypothesis testing), 4 = Approximate Bayesian (mean and variance estimation))"
- static const std::string `ARG_HELP_NAME_LONG` = "help"
- static const std::string `ARG_HELP_NAME_BOTH` = ",h"
- static const std::string `ARG_HELP_DESCRIPTION` = "display help message (describing the meaning and usage of each command line argument)"
- static const std::string `ARG_EXTRA_EVALUATION_PROGRAM_NAME_LONG` = "extra-evaluation-program"
- static const std::string `ARG_EXTRA_EVALUATION_PROGRAM_NAME_BOTH` = ",p"
- static const std::string `ARG_EXTRA_EVALUATION_PROGRAM_DESCRIPTION` = "the program which will be executed whenever extra evaluation (and input traces) is required"
- static const std::string `ARG_MULTISCALE_ARCHITECTURE_GRAPH_NAME_LONG` = "multiscale-architecture-graph"
- static const std::string `ARG_MULTISCALE_ARCHITECTURE_GRAPH_NAME_BOTH` = ",a"
- static const std::string `ARG_MULTISCALE_ARCHITECTURE_GRAPH_DESCRIPTION` = "the multiscale architecture graph encoding the hierarchical structure of the considered system"
- static const std::string `ARG_VERBOSE_NAME_LONG` = "verbose"
- static const std::string `ARG_VERBOSE_NAME_BOTH` = ",v"
- static const std::string `ARG_VERBOSE_DESCRIPTION` = "if this flag is set detailed evaluation results will be displayed"
- static const std::string `ARG_TYPE_I_ERROR_NAME_LONG` = "type-I-error"
- static const std::string `ARG_TYPE_I_ERROR_DESCRIPTION` = "the probability of type I errors"
- static const std::string `ARG_TYPE_II_ERROR_NAME_LONG` = "type-II-error"
- static const std::string `ARG_TYPE_II_ERROR_DESCRIPTION` = "the probability of type II errors"
- static const std::string `ARG_DELTA_NAME_LONG` = "delta"
- static const std::string `ARG_DELTA_DESCRIPTION` = "the upper bound on the probability to deviate from the true probability"
- static const std::string `ARG_EPSILON_NAME_LONG` = "epsilon"
- static const std::string `ARG_EPSILON_DESCRIPTION` = "the considered deviation from the true probability"
- static const std::string `ARG_BAYESIAN_ALPHA_NAME_LONG` = "bayesian-alpha"
- static const std::string `ARG_BAYESIAN_ALPHA_DESCRIPTION` = "the alpha shape parameter of the Beta distribution prior"
- static const std::string `ARG_BAYESIAN_BETA_NAME_LONG` = "bayesian-beta"

- static const std::string `ARG_BAYESIAN_BETA_DESCRIPTION` = "the beta shape parameter of the Beta distribution prior"
- static const std::string `ARG_BAYES_FACTOR_THRESHOLD_NAME_LONG` = "bayes-factor-threshold"
- static const std::string `ARG_BAYES_FACTOR_THRESHOLD_DESCRIPTION` = "the Bayes factor threshold used to fix the confidence level of the answer"
- static const std::string `ARG_APPROXIMATE_BAYESIAN_ALPHA_NAME_LONG` = "approximate-bayesian-alpha"
- static const std::string `ARG_APPROXIMATE_BAYESIAN_ALPHA_DESCRIPTION` = "the alpha shape parameter of the Beta distribution prior"
- static const std::string `ARG_APPROXIMATE_BAYESIAN_BETA_NAME_LONG` = "approximate-bayesian-beta"
- static const std::string `ARG_APPROXIMATE_BAYESIAN_BETA_DESCRIPTION` = "the beta shape parameter of the Beta distribution prior"
- static const std::string `ARG_VARIANCE_THRESHOLD_NAME_LONG` = "variance-threshold"
- static const std::string `ARG_VARIANCE_THRESHOLD_DESCRIPTION` = "the variance threshold used to fix the confidence level of the answer"
- static const std::string `HELP_NAME_LABEL` = "NAME:"
- static const std::string `HELP_NAME_MSG` = " Mule - Multiscale multidimensional meta-model checker"
- static const std::string `HELP_USAGE_LABEL` = "USAGE:"
- static const std::string `HELP_USAGE_MSG` = " Mule <required-arguments> [<optional-arguments>] <model-checking-type-specific-arguments>"
- static const std::string `HELP_DESCRIPTION_LABEL` = "DESCRIPTION:"
- static const std::string `HELP_DESCRIPTION_MSG` = " Mule is a multiscale multidimensional (spatial-temporal) approximate probabilistic meta-model checker. It can be used for two different types of applications. First of all Mule can be employed to validate logic properties against multidimensional multiscale models. Secondly it can be used in reverse mode as a method to query time series data generated by in vivo/vitro experiments. Properties of interest are formalised using a multiscale spatio-temporal logic and their validity is checked using Mule."
- static const std::string `HELP_AUTHOR_LABEL` = "AUTHOR:"
- static const std::string `HELP_AUTHOR_MSG` = " The author of this software is Ovidiu Parvu."
- static const std::string `HELP_COPYRIGHT_LABEL` = "COPYRIGHT:"
- static const std::string `HELP_COPYRIGHT_MSG` = " Copyright Ovidiu Parvu 2014."
- static const std::string `HELP_REPORTING_BUGS_LABEL` = "REPORTING BUGS:"
- static const std::string `HELP_REPORTING_BUGS_MSG` = " Please send requests for fixing bugs or recommendations to <ovidiu.parvu[AT]gmail.com>."
- static const std::string `MSG_MODEL_CHECKING_HELP_REQUESTED` = "A request for displaying help information was issued."
- static const unsigned int `MODEL_CHECKER_TYPE_PROBABILISTIC_BLACK_BOX` = 0
- static const unsigned int `MODEL_CHECKER_TYPE_STATISTICAL` = 1

- static const unsigned int `MODEL_CHECKER_TYPE_APPROXIMATE_PROBABILISTIC` = 2
- static const unsigned int `MODEL_CHECKER_TYPE_BAYESIAN` = 3
- static const unsigned int `MODEL_CHECKER_TYPE_APPROXIMATE_BAYESIAN` = 4
- static const std::string `MODEL_CHECKER_PROBABILISTIC_BLACK_BOX_NAME` = "Probabilistic black-box"
- static const std::string `MODEL_CHECKER_PROBABILISTIC_BLACK_BOX_PARAMETERS` = "None"
- static const std::string `MODEL_CHECKER_STATISTICAL_NAME` = "Frequentist statistical"
- static const std::string `MODEL_CHECKER_STATISTICAL_PARAMETERS_BEGIN` = "Probability of type I errors (false negatives) = "
- static const std::string `MODEL_CHECKER_STATISTICAL_PARAMETERS_MIDDLE` = " and of type II errors (false positives) = "
- static const std::string `MODEL_CHECKER_STATISTICAL_PARAMETERS_END` = ":"
- static const std::string `MODEL_CHECKER_APPROXIMATE_PROBABILISTIC_NAME` = "Frequentist approximate probabilistic (Chernoff-Hoeffding)"
- static const std::string `MODEL_CHECKER_APPROXIMATE_PROBABILISTIC_PARAMETERS_BEGIN` = "Upper bound on probability to deviate more than epsilon = "
- static const std::string `MODEL_CHECKER_APPROXIMATE_PROBABILISTIC_PARAMETERS_MIDDLE` = " from the true probability is delta = "
- static const std::string `MODEL_CHECKER_APPROXIMATE_PROBABILISTIC_PARAMETERS_END` = ":"
- static const std::string `MODEL_CHECKER_BAYESIAN_NAME` = "Bayesian (statistical hypothesis testing)"
- static const std::string `MODEL_CHECKER_BAYESIAN_PARAMETERS_BEGIN` = "Beta distribution prior shape parameters alpha = "
- static const std::string `MODEL_CHECKER_BAYESIAN_PARAMETERS_MIDDLE1` = " and beta = "
- static const std::string `MODEL_CHECKER_BAYESIAN_PARAMETERS_MIDDLE2` = ". Bayes factor threshold = "
- static const std::string `MODEL_CHECKER_BAYESIAN_PARAMETERS_END` = ":"
- static const std::string `MODEL_CHECKER_APPROXIMATE_BAYESIAN_NAME` = "Approximate Bayesian (mean and variance estimate)"
- static const std::string `MODEL_CHECKER_APPROXIMATE_BAYESIAN_PARAMETERS_BEGIN` = "Beta distribution prior shape parameters alpha = "
- static const std::string `MODEL_CHECKER_APPROXIMATE_BAYESIAN_PARAMETERS_MIDDLE1` = " and beta = "
- static const std::string `MODEL_CHECKER_APPROXIMATE_BAYESIAN_PARAMETERS_MIDDLE2` = ". Variance threshold = "
- static const std::string `MODEL_CHECKER_APPROXIMATE_BAYESIAN_PARAMETERS_END` = ":"
- static const std::string `CONFIG_CAPTION_ALLOWED_ARGUMENTS` = ""

- static const std::string **CONFIG\_CAPTION\_REQUIRED\_ARGUMENTS** = "REQUIRED ARGUMENTS"
- static const std::string **CONFIG\_CAPTION\_OPTIONAL\_ARGUMENTS** = "OPTIONAL ARGUMENTS"
- static const std::string **CONFIG\_CAPTION\_MODEL\_CHECKER\_TYPE\_SPECIFIC\_ARGUMENTS** = "MODEL CHECKING TYPE SPECIFIC ARGUMENTS"
- static const std::string **CONFIG\_CAPTION\_PROBABILISTIC\_BLACK\_BOX\_MODEL\_CHECKER\_ARGUMENTS** = MODEL\_CHECKER\_PROBABILISTIC\_BLACK\_BOX\_NAME
- static const std::string **CONFIG\_CAPTION\_STATISTICAL\_MODEL\_CHECKER\_ARGUMENTS** = MODEL\_CHECKER\_STATISTICAL\_NAME
- static const std::string **CONFIG\_CAPTION\_APPROXIMATE\_PROBABILISTIC\_MODEL\_CHECKER\_ARGUMENTS** = MODEL\_CHECKER\_APPROXIMATE\_PROBABILISTIC\_NAME
- static const std::string **CONFIG\_CAPTION\_BAYESIAN\_MODEL\_CHECKER\_ARGUMENTS** = MODEL\_CHECKER\_BAYESIAN\_NAME
- static const std::string **CONFIG\_CAPTION\_APPROXIMATE\_BAYESIAN\_MODEL\_CHECKER\_ARGUMENTS** = MODEL\_CHECKER\_APPROXIMATE\_BAYESIAN\_NAME

### **6.48.1 Detailed Description**

Class for running model checkers from the command line.

Definition at line 22 of file CommandLineModelChecking.hpp.

### **6.48.2 Constructor & Destructor Documentation**

#### **6.48.2.1 CommandLineModelChecking::CommandLineModelChecking( )**

Definition at line 25 of file CommandLineModelChecking.cpp.

#### **6.48.2.2 CommandLineModelChecking::~CommandLineModelChecking( )**

Definition at line 34 of file CommandLineModelChecking.cpp.

### **6.48.3 Member Function Documentation**

#### **6.48.3.1 bool CommandLineModelChecking::areApproximateBayesianModelCheckingArgumentsPresent( const po::variables\_map & variablesMap ) [private]**

Check if the arguments specific to approximate Bayesian model checking are present.

**Parameters**

<b>variables-Map</b>	The map containing all parsed command line arguments
----------------------	--

Definition at line 314 of file CommandLineModelChecking.cpp.

References ARG\_APPROXIMATE\_BAYESIAN\_ALPHA\_NAME\_LONG, ARG\_APPROXIMATE\_BAYESIAN\_BETA\_NAME\_LONG, and ARG\_VARIANCE\_THRESHOLD\_NAME\_LONG.

Referenced by areModelCheckingTypeSpecificArgumentsPresent().

```
6.48.3.2 bool CommandLineModelChecking::areApproximateProbabilisticModel-
    CheckingArgumentsPresent ( const po::variables_map & variablesMap )
    [private]
```

Check if the arguments specific to approximate probabilistic model checking are present.

**Parameters**

<b>variables-Map</b>	The map containing all parsed command line arguments
----------------------	--

Definition at line 299 of file CommandLineModelChecking.cpp.

References ARG\_DELTA\_NAME\_LONG, and ARG\_EPSILON\_NAME\_LONG.

Referenced by areModelCheckingTypeSpecificArgumentsPresent().

```
6.48.3.3 bool CommandLineModelChecking::areBayesianModelChecking-
    ArgumentsPresent ( const po::variables_map & variablesMap )
    [private]
```

Check if the arguments specific to Bayesian model checking are present.

**Parameters**

<b>variables-Map</b>	The map containing all parsed command line arguments
----------------------	--

Definition at line 306 of file CommandLineModelChecking.cpp.

References ARG\_BAYES\_FACTOR\_THRESHOLD\_NAME\_LONG, ARG\_BAYESIAN\_ALPHA\_NAME\_LONG, and ARG\_BAYESIAN\_BETA\_NAME\_LONG.

Referenced by areModelCheckingTypeSpecificArgumentsPresent().

**6.48.3.4 bool CommandLineModelChecking::areInvalidExecutionArguments ( const po::parsed\_options & *parsedArguments* ) [private]**

Check if any invalid execution arguments were provided.

**Parameters**

<i>parsed- Arguments</i>	The parsed command line arguments
------------------------------	-----------------------------------

Definition at line 153 of file CommandLineModelChecking.cpp.

References areUnrecognizedArgumentsPresent(), and isHelpArgumentPresent().

Referenced by areValidArgumentsConsideringConfiguration().

**6.48.3.5 bool CommandLineModelChecking::areInvalidModelCheckingArguments ( ) [private]**

Check if any invalid model checker type dependent arguments are present.

Definition at line 212 of file CommandLineModelChecking.cpp.

References areInvalidModelCheckingArgumentsPresent(), and ERR\_INVALID\_MODEL\_CHECKING\_ARGUMENTS.

Referenced by areValidArgumentsConsideringConfiguration().

**6.48.3.6 bool CommandLineModelChecking::areInvalidModelChecking-  
ArgumentsPresent ( ) [private]**

Check if any model checker type dependent arguments are invalid.

Definition at line 220 of file CommandLineModelChecking.cpp.

References areInvalidModelCheckingTypeSpecificArguments(), ARG\_MODEL\_CHECKER\_TYPE\_NAME\_LONG, modelCheckerType, removeOptionalArguments(), removeRequiredArguments(), and variablesMap.

Referenced by areInvalidModelCheckingArguments().

**6.48.3.7 bool CommandLineModelChecking::areInvalidModelCheckingType-  
SpecificArguments ( unsigned int *modelCheckerType*, po::variables\_map &  
*variablesMap* ) [private]**

Check if the model checking type specific arguments from the given variables\_map are invalid.

**Parameters**

<i>model- Checker- Type</i>	The type of the model checker
-------------------------------------	-------------------------------

<i>variables-Map</i>	The map containing all parsed command line arguments
----------------------	--

Definition at line 255 of file CommandLineModelChecking.cpp.

References `areModelCheckingTypeSpecificArgumentsPresent()`, and `removeModelCheckingTypeSpecificArguments()`.

Referenced by `areInvalidModelCheckingArgumentsPresent()`.

```
6.48.3.8 bool CommandLineModelChecking::areModelCheckingTypeSpecific-
ArgumentsPresent ( unsigned int modelCheckerType, const po::variables_map &
variablesMap ) [private]
```

Check if all model checking type specific arguments are present.

#### Parameters

<i>model-Checker-Type</i>	The type of the model checker
<i>variables-Map</i>	The map containing all parsed command line arguments

Definition at line 266 of file CommandLineModelChecking.cpp.

References `areApproximateBayesianModelCheckingArgumentsPresent()`, `areApproximateProbabilisticModelCheckingArgumentsPresent()`, `areBayesianModelCheckingArgumentsPresent()`, `areStatisticalModelCheckingArgumentsPresent()`, `ERR_INVALID_MODEL_CHECKING_TYPE`, `MODEL_CHECKER_TYPE_APPROXIMATE_BAYESIAN`, `MODEL_CHECKER_TYPE_APPROXIMATE_PROBABILISTIC`, `MODEL_CHECKER_TYPE_BAYESIAN`, `MODEL_CHECKER_TYPE_PROBABILISTIC_BLACK_BOX`, and `MODEL_CHECKER_TYPE_STATISTICAL`.

Referenced by `areInvalidModelCheckingTypeSpecificArguments()`.

```
6.48.3.9 bool CommandLineModelChecking::areStatisticalModelChecking-
ArgumentsPresent ( const po::variables_map & variablesMap ) [private]
```

Check if the arguments specific to statistical model checking are present.

#### Parameters

<i>variables-Map</i>	The map containing all parsed command line arguments
----------------------	--

Definition at line 292 of file CommandLineModelChecking.cpp.

## **6.48 multiscale::verification::CommandLineModelChecking Class Reference 259**

---

References ARG\_TYPE\_I\_ERROR\_NAME\_LONG, and ARG\_TYPE\_II\_ERROR\_NAME\_LONG.

Referenced by areModelCheckingTypeSpecificArgumentsPresent().

**6.48.3.10 bool CommandLineModelChecking::areUnrecognizedArgumentsPresent ( const po::parsed\_options & *parsedArguments* ) [private]**

Check if any unrecognized command line arguments are present.

### Parameters

<i>parsed-Arguments</i>	The parsed command line arguments
-------------------------	-----------------------------------

Definition at line 205 of file CommandLineModelChecking.cpp.

Referenced by areInvalidExecutionArguments().

**6.48.3.11 bool CommandLineModelChecking::isValidArguments ( int *argc*, char \*\* *argv* ) [private]**

Check if the provided command line arguments are valid.

### Parameters

<i>argc</i>	The number of provided command line arguments
<i>argv</i>	The collection of command line arguments

Definition at line 51 of file CommandLineModelChecking.cpp.

References isValidArgumentsConsideringConfiguration(), and initialiseAllowedArgumentsConfiguration().

Referenced by initialise().

**6.48.3.12 bool CommandLineModelChecking::isValidArgumentsConsideringConfiguration ( int *argc*, char \*\* *argv* ) [private]**

Check if the provided command line arguments are valid.

### Parameters

<i>argc</i>	The number of provided command line arguments
<i>argv</i>	The collection of command line arguments

Definition at line 131 of file CommandLineModelChecking.cpp.

References `areInvalidExecutionArguments()`, `areInvalidModelCheckingArguments()`, `parseAndStoreArgumentsValues()`, and `variablesMap`.

Referenced by `areValidArguments()`.

#### 6.48.3.13 void `CommandLineModelChecking::execute( )`

Execute the model checking task.

Definition at line 47 of file `CommandLineModelChecking.cpp`.

References `modelCheckerFactory`, and `modelCheckingManager`.

#### 6.48.3.14 void `CommandLineModelChecking::handleHelpRequest( )` [private]

Handle the help request i.e. if the `--help` flag was provided.

Definition at line 164 of file `CommandLineModelChecking.cpp`.

References `MSG_MODEL_CHECKING_HELP_REQUESTED`, and `printHelpMessage()`.

Referenced by `initialise()`.

#### 6.48.3.15 void `CommandLineModelChecking::initialise( int argc, char ** argv )`

Initialise the class with the given command line arguments.

##### Parameters

<code>argc</code>	The number of provided command line arguments
<code>argv</code>	The collection of command line arguments

Definition at line 36 of file `CommandLineModelChecking.cpp`.

References `areValidArguments()`, `ERR_INVALID_COMMAND_LINE_ARGUMENTS`, `handleHelpRequest()`, `initialiseClassMembers()`, `isHelpArgumentPresent()`, and `printModelCheckingInitialisationMessage()`.

#### 6.48.3.16 void `CommandLineModelChecking::initialiseAllowedArgumentsConfiguration( )` [private]

Initialise the configuration of allowed command line arguments.

Definition at line 57 of file `CommandLineModelChecking.cpp`.

References `allowedArguments`, `initialiseModelCheckerTypeSpecificArgumentsConfiguration()`, `initialiseOptionalArgumentsConfiguration()`, `initialiseRequiredArgumentsConfiguration()`, `modelCheckerTypeSpecificArguments`, `optionalArguments`, and `requiredArguments`.

Referenced by `isValidArguments()`.

**6.48.3.17 void `CommandLineModelChecking::initialiseApproximateBayesianModelChecker( )` [private]**

Initialise the approximate Bayesian model checker.

Definition at line 489 of file `CommandLineModelChecking.cpp`.

References `ARG_APPROXIMATE_BAYESIAN_ALPHA_NAME_LONG`, `ARG_APPROXIMATE_BAYESIAN_BETA_NAME_LONG`, `ARG_VARIANCE_THRESHOLD_NAME_LONG`, `MODEL_CHECKER_APPROXIMATE_BAYESIAN_NAME`, `MODEL_CHECKER_APPROXIMATE_BAYESIAN_PARAMETERS_BEGIN`, `MODEL_CHECKER_APPROXIMATE_BAYESIAN_PARAMETERS_END`, `MODEL_CHECKER_APPROXIMATE_BAYESIAN_PARAMETERS_MIDDLE1`, `MODEL_CHECKER_APPROXIMATE_BAYESIAN_PARAMETERS_MIDDLE2`, `modelCheckerFactory`, `modelCheckerParameters`, `modelCheckerTypeName`, `multiscale::StringManipulator::toString()`, and `variablesMap`.

Referenced by `initialiseModelChecker()`.

**6.48.3.18 `po::options_description CommandLineModelChecking::initialiseApproximateBayesianModelCheckerArgumentsConfiguration( )` [private]**

Initialise the configuration of the approximate Bayesian model checker command line arguments.

Definition at line 121 of file `CommandLineModelChecking.cpp`.

References `ARG_APPROXIMATE_BAYESIAN_ALPHA_NAME_LONG`, `ARG_APPROXIMATE_BAYESIAN_BETA_NAME_LONG`, `ARG_BAYESIAN_ALPHA_DESCRIPTION`, `ARG_BAYESIAN_BETA_DESCRIPTION`, `ARG_VARIANCE_THRESHOLD_DESCRIPTION`, `ARG_VARIANCE_THRESHOLD_NAME_LONG`, and `CONFIG_CAPTURE_APPROXIMATE_BAYESIAN_MODEL_CHECKER_ARGUMENTS`.

Referenced by `initialiseModelCheckerTypeSpecificArgumentsConfiguration()`.

**6.48.3.19 void `CommandLineModelChecking::initialiseApproximateProbabilisticModelChecker( )` [private]**

Initialise the approximate probabilistic model checker.

Definition at line 454 of file `CommandLineModelChecking.cpp`.

References `ARG_DELTA_NAME_LONG`, `ARG_EPSILON_NAME_LONG`, `MODEL_CHECKER_APPROXIMATE_PROBABILISTIC_NAME`, `MODEL_CHECKER_APPROXIMATE_PROBABILISTIC_PARAMETERS_BEGIN`, `MODEL_CHECKER_APPROXIMATE_PROBABILISTIC_PARAMETERS_END`, `MODEL_CHECKER_APPROXIMATE_PROBABILISTIC_PARAMETERS_MIDDLE`, `modelCheckerFactory`, `modelCheckerParameters`, `modelCheckerTypeName`, `multiscale::StringManipulator::toString()`, and `variablesMap`.

Referenced by initialiseModelChecker().

**6.48.3.20 po::options\_description CommandLineModelChecking::initialiseApproximateProbabilisticModelCheckerArgumentsConfiguration( ) [private]**

Initialise the configuration of the approximate probabilistic model checker command line arguments.

Definition at line 102 of file CommandLineModelChecking.cpp.

References ARG\_DELTA\_DESCRIPTION, ARG\_DELTA\_NAME\_LONG, ARG\_EPSILON\_DESCRIPTION, ARG\_EPSILON\_NAME\_LONG, and CONFIG\_CAPTION\_APPROXIMATE\_PROBABILISTIC\_MODEL\_CHECKER\_ARGUMENTS.

Referenced by initialiseModelCheckerTypeSpecificArgumentsConfiguration().

**6.48.3.21 void CommandLineModelChecking::initialiseBayesianModelChecker( ) [private]**

Initialise the Bayesian model checker.

Definition at line 470 of file CommandLineModelChecking.cpp.

References ARG\_BAYES\_FACTOR\_THRESHOLD\_NAME\_LONG, ARG\_BAYESIAN\_ALPHA\_NAME\_LONG, ARG\_BAYESIAN\_BETA\_NAME\_LONG, MODEL\_CHECKER\_BAYESIAN\_NAME, MODEL\_CHECKER\_BAYESIAN\_PARAMETERS\_BEGIN, MODEL\_CHECKER\_BAYESIAN\_PARAMETERS\_END, MODEL\_CHECKER\_BAYESIAN\_PARAMETERS\_MIDDLE1, MODEL\_CHECKER\_BAYESIAN\_PARAMETERS\_MIDDLE2, modelCheckerFactory, modelCheckerParameters, modelCheckerTypeName, multiscale::StringManipulator::toString(), and variablesMap.

Referenced by initialiseModelChecker().

**6.48.3.22 po::options\_description CommandLineModelChecking::initialiseBayesianModelCheckerArgumentsConfiguration( ) [private]**

Initialise the configuration of the Bayesian model checker command line arguments.

Definition at line 111 of file CommandLineModelChecking.cpp.

References ARG\_BAYES\_FACTOR\_THRESHOLD\_DESCRIPTION, ARG\_BAYES\_FACTOR\_THRESHOLD\_NAME\_LONG, ARG\_BAYESIAN\_ALPHA\_DESCRIPTION, ARG\_BAYESIAN\_ALPHA\_NAME\_LONG, ARG\_BAYESIAN\_BETA\_DESCRIPTION, ARG\_BAYESIAN\_BETA\_NAME\_LONG, and CONFIG\_CAPTION\_BAYESIAN\_MODEL\_CHECKER\_ARGUMENTS.

Referenced by initialiseModelCheckerTypeSpecificArgumentsConfiguration().

**6.48.3.23 void CommandLineModelChecking::initialiseClassMembers ( )**  
[private]

Initialise the class members using the command line arguments.

Definition at line 371 of file CommandLineModelChecking.cpp.

References initialiseModelCheckerTypeDependentClassMembers(), initialiseOptionalArgumentsDependentClassMembers(), and initialiseRequiredArgumentsDependentClassMembers().

Referenced by initialise().

**6.48.3.24 void CommandLineModelChecking::initialiseModelChecker ( )**  
[private]

Initialise the model checker.

Definition at line 404 of file CommandLineModelChecking.cpp.

References ERR\_INVALID\_MODEL\_CHECKING\_TYPE, initialiseApproximateBayesianModelChecker(), initialiseApproximateProbabilisticModelChecker(), initialiseBayesianModelChecker(), initialiseProbabilisticBlackBoxModelChecker(), initialiseStatisticalModelChecker(), MODEL\_CHECKER\_TYPE\_APPROXIMATE\_BAYESIAN, MODEL\_CHECKER\_TYPE\_APPROXIMATE\_PROBABILISTIC, MODEL\_CHECKER\_TYPE\_BAYESIAN, MODEL\_CHECKER\_TYPE\_PROBABILISTIC\_BLACK\_BOX, MODEL\_CHECKER\_TYPE\_STATISTICAL, and modelCheckerType.

Referenced by initialiseModelCheckerTypeDependentClassMembers().

**6.48.3.25 void CommandLineModelChecking::initialise-  
ModelCheckerTypeDependentClassMembers ( )**  
[private]

Initialise the class members dependent on the model checker type.

Definition at line 399 of file CommandLineModelChecking.cpp.

References initialiseModelChecker(), and initialiseModelCheckingManager().

Referenced by initialiseClassMembers().

**6.48.3.26 void CommandLineModelChecking::initialiseModel-  
CheckerTypeSpecificArgumentsConfiguration ( )**  
[private]

Initialise the configuration of model checker type specific command line arguments.

Definition at line 81 of file CommandLineModelChecking.cpp.

References initialiseApproximateBayesianModelCheckerArgumentsConfiguration(), initialiseApproximateProbabilisticModelCheckerArgumentsConfiguration(), initialise-

BayesianModelCheckerArgumentsConfiguration(), initialiseStatisticalModelCheckerArgumentsConfiguration(), and modelCheckerTypeSpecificArguments.

Referenced by initialiseAllowedArgumentsConfiguration().

**6.48.3.27 void CommandLineModelChecking::initialiseModelCheckingManager ( ) [private]**

Initialise the model checking manager.

Definition at line 510 of file CommandLineModelChecking.cpp.

References extraEvaluationProgramPath, extraEvaluationTime, logicQueriesFilepath, modelCheckingManager, multiscaleArchitectureGraphFilepath, shouldVerboseDetailedResults, and tracesFolderPath.

Referenced by initialiseModelCheckerTypeDependentClassMembers().

**6.48.3.28 void CommandLineModelChecking::initialiseOptionalArgumentsConfiguration( ) [private]**

Initialise the configuration of optional command line arguments.

Definition at line 74 of file CommandLineModelChecking.cpp.

References ARG\_EXTRA\_EVALUATION\_PROGRAM\_DESCRIPTION, ARG\_EXTRA\_EVALUATION\_PROGRAM\_NAME\_BOTH, ARG\_HELP\_DESCRIPTION, ARG\_HELP\_NAME\_BOTH, ARG\_MULTISCALE\_ARCHITECTURE\_GRAPH\_DESCRIPTION, ARG\_MULTISCALE\_ARCHITECTURE\_GRAPH\_NAME\_BOTH, ARG\_VERBOSE\_DESCRIPTION, ARG\_VERBOSE\_NAME\_BOTH, and optionalArguments.

Referenced by initialiseAllowedArgumentsConfiguration().

**6.48.3.29 void CommandLineModelChecking::initialiseOptionalArgumentsDependentClassMembers( ) [private]**

Initialise the class members dependent on optional command line arguments.

Definition at line 384 of file CommandLineModelChecking.cpp.

References ARG\_EXTRA\_EVALUATION\_PROGRAM\_NAME\_LONG, ARG\_MULTISCALE\_ARCHITECTURE\_GRAPH\_NAME\_LONG, ARG\_VERBOSE\_NAME\_LONG, extraEvaluationProgramPath, multiscaleArchitectureGraphFilepath, shouldVerboseDetailedResults, and variablesMap.

Referenced by initialiseClassMembers().

**6.48.3.30 void CommandLineModelChecking::initialiseProbabilisticBlackBoxModelChecker( ) [private]**

Initialise the probabilistic black box model checker.

## **6.48 multiscale::verification::CommandLineModelChecking Class Reference 265**

---

Definition at line 431 of file CommandLineModelChecking.cpp.

References MODEL\_CHECKER\_PROBABILISTIC\_BLACK\_BOX\_NAME, MODEL\_CHECKER\_PROBABILISTIC\_BLACK\_BOX\_PARAMETERS, modelCheckerFactory, modelCheckerParameters, and modelCheckerTypeName.

Referenced by initialiseModelChecker().

### **6.48.3.31 void CommandLineModelChecking::initialiseRequiredArgumentsConfiguration( ) [private]**

Initialise the configuration of required command line arguments.

Definition at line 67 of file CommandLineModelChecking.cpp.

References ARG\_EXTRA\_EVALUATION\_TIME\_DESCRIPTION, ARG\_EXTRA\_EVALUATION\_TIME\_NAME\_BOTH, ARG\_LOGIC\_QUERIES\_DESCRIPTION, ARG\_LOGIC\_QUERIES\_NAME\_BOTH, ARG\_MODEL\_CHECKER\_TYPE\_DESCRIPTION, ARG\_MODEL\_CHECKER\_TYPE\_NAME\_BOTH, ARG\_SPATIAL\_TEMPORAL\_TRACES\_DESCRIPTION, ARG\_SPATIAL\_TEMPORAL\_TRACES\_NAME\_BOTH, and requiredArguments.

Referenced by initialiseAllowedArgumentsConfiguration().

### **6.48.3.32 void CommandLineModelChecking::initialiseRequiredArgumentsDependentClassMembers( ) [private]**

Initialise the class members dependent on required command line arguments.

Definition at line 377 of file CommandLineModelChecking.cpp.

References ARG\_EXTRA\_EVALUATION\_TIME\_NAME\_LONG, ARG\_LOGIC\_QUERIES\_NAME\_LONG, ARG\_MODEL\_CHECKER\_TYPE\_NAME\_LONG, ARG\_SPATIAL\_TEMPORAL\_TRACES\_NAME\_LONG, extraEvaluationTime, logicQueriesFilepath, modelCheckerType, tracesFolderPath, and variablesMap.

Referenced by initialiseClassMembers().

### **6.48.3.33 void CommandLineModelChecking::initialiseStatisticalModelChecker( ) [private]**

Initialise the statistical model checker.

Definition at line 438 of file CommandLineModelChecking.cpp.

References ARG\_TYPE\_I\_ERROR\_NAME\_LONG, ARG\_TYPE\_II\_ERROR\_NAME\_LONG, MODEL\_CHECKER\_STATISTICAL\_NAME, MODEL\_CHECKER\_STATISTICAL\_PARAMETERS\_BEGIN, MODEL\_CHECKER\_STATISTICAL\_PARAMETERS\_END, MODEL\_CHECKER\_STATISTICAL\_PARAMETERS\_MIDDLE, modelCheckerFactory, modelCheckerParameters, modelCheckerTypeName, multiscale::StringManipulator::toString(), and variablesMap.

Referenced by initialiseModelChecker().

**6.48.3.34 po::options\_description CommandLineModelChecking::initialiseStatisticalModelCheckerArgumentsConfiguration ( )**  
[private]

Initialise the configuration of the statistical model checker command line arguments.

Definition at line 93 of file CommandLineModelChecking.cpp.

References ARG\_TYPE\_I\_ERROR\_DESCRIPTION, ARG\_TYPE\_I\_ERROR\_NAME\_LONG, ARG\_TYPE\_II\_ERROR\_DESCRIPTION, ARG\_TYPE\_II\_ERROR\_NAME\_LONG, and CONFIG\_CAPTION\_STATISTICAL\_MODEL\_CHECKER\_ARGUMENTS.

Referenced by initialiseModelCheckerTypeSpecificArgumentsConfiguration().

**6.48.3.35 bool CommandLineModelChecking::isHelpArgumentPresent ( )**  
[private]

Check if the help command line argument is present.

Definition at line 160 of file CommandLineModelChecking.cpp.

References ARG\_HELP\_NAME\_LONG, and variablesMap.

Referenced by areInvalidExecutionArguments(), and initialise().

**6.48.3.36 po::parsed\_options CommandLineModelChecking::parseAndStoreArgumentsValues ( int argc, char \*\* argv )**  
[private]

Parse and store the command line arguments' values in a variables map.

#### Parameters

<i>argc</i>	The number of provided command line arguments
<i>argv</i>	The collection of command line arguments

Definition at line 144 of file CommandLineModelChecking.cpp.

References allowedArguments, and variablesMap.

Referenced by areValidArgumentsConsideringConfiguration().

**6.48.3.37 void CommandLineModelChecking::printHelpClosingMessage ( )**  
[private]

Print the help closing message to the console.

Definition at line 192 of file CommandLineModelChecking.cpp.

References HELP\_AUTHOR\_LABEL, HELP\_AUTHOR\_MSG, HELP\_COPYRIGHT\_L-

## **6.48 multiscale::verification::CommandLineModelChecking Class Reference 267**

---

ABEL, HELP\_COPYRIGHT\_MSG, HELP\_REPORTING\_BUGS\_LABEL, and HELP\_R-EPORTING\_BUGS\_MSG.

Referenced by printHelpMessage().

**6.48.3.38 void CommandLineModelChecking::printHelpContentsMessage( )**  
[private]

Print the help contents message to the console.

Definition at line 188 of file CommandLineModelChecking.cpp.

References allowedArguments.

Referenced by printHelpMessage().

**6.48.3.39 void CommandLineModelChecking::printHelpIntroMessage( )**  
[private]

Print the help intro message to the console.

Definition at line 176 of file CommandLineModelChecking.cpp.

References HELP\_DESCRIPTION\_LABEL, HELP\_DESCRIPTION\_MSG, HELP\_NA-ME\_LABEL, HELP\_NAME\_MSG, HELP\_USAGE\_LABEL, and HELP\_USAGE\_MSG.

Referenced by printHelpMessage().

**6.48.3.40 void CommandLineModelChecking::printHelpMessage( )**  
[private]

Print help message to the console.

Definition at line 170 of file CommandLineModelChecking.cpp.

References printHelpClosingMessage(), printHelpContentsMessage(), and printHelp-IntroMessage().

Referenced by handleHelpRequest().

**6.48.3.41 void CommandLineModelChecking::printModelCheckingInitialisation-  
Message( ) [private]**

Print the model checking initialisation message.

Definition at line 520 of file CommandLineModelChecking.cpp.

References extraEvaluationTime, logicQueriesFilepath, modelCheckerParameters, modelCheckerTypeName, multiscale::verification::ModelCheckingOutputWriter::print-InitialisationMessage(), multiscale::verification::ModelCheckingOutputWriter::print-IntroductionMessage(), and tracesFolderPath.

Referenced by initialise().

---

**6.48.3.42 void CommandLineModelChecking::removeApproximateBayesianModelCheckingArguments ( po::variables\_map & variablesMap ) [private]**

Remove the approximate Bayesian model checking arguments from the given variables\_map.

**Parameters**

<b>variables-Map</b>	The map containing all parsed command line arguments
----------------------	--

Definition at line 365 of file CommandLineModelChecking.cpp.

References ARG\_APPROXIMATE\_BAYESIAN\_ALPHA\_NAME\_LONG, ARG\_APPROXIMATE\_BAYESIAN\_BETA\_NAME\_LONG, and ARG\_VARIANCE\_THRESHOLD\_NAME\_LONG.

Referenced by removeModelCheckingTypeSpecificArguments().

---

**6.48.3.43 void CommandLineModelChecking::removeApproximateProbabilisticModelCheckingArguments ( po::variables\_map & variablesMap ) [private]**

Remove the approximate probabilistic model checking arguments from the given variables\_map.

**Parameters**

<b>variables-Map</b>	The map containing all parsed command line arguments
----------------------	--

Definition at line 354 of file CommandLineModelChecking.cpp.

References ARG\_DELTA\_NAME\_LONG, and ARG\_EPSILON\_NAME\_LONG.

Referenced by removeModelCheckingTypeSpecificArguments().

---

**6.48.3.44 void CommandLineModelChecking::removeBayesianModelCheckingArguments ( po::variables\_map & variablesMap ) [private]**

Remove the Bayesian model checking arguments from the given variables\_map.

**Parameters**

<b>variables-Map</b>	The map containing all parsed command line arguments
----------------------	--

Definition at line 359 of file CommandLineModelChecking.cpp.

## **6.48 multiscale::verification::CommandLineModelChecking Class Reference 269**

References ARG\_BAYES\_FACTOR\_THRESHOLD\_NAME\_LONG, ARG\_BAYESIAN\_ALPHA\_NAME\_LONG, and ARG\_BAYESIAN\_BETA\_NAME\_LONG.

Referenced by removeModelCheckingTypeSpecificArguments().

**6.48.3.45 void CommandLineModelChecking::removeModelCheckingType-SpecificArguments ( unsigned int *modelCheckerType*, po::variables\_map & *variablesMap* ) [private]**

Remove the model checking type specific arguments from the given variables\_map.

### Parameters

<i>model- Checker- Type</i>	The type of the model checker
<i>variables- Map</i>	The map containing all parsed command line arguments

Definition at line 322 of file CommandLineModelChecking.cpp.

References ERR\_INVALID\_MODEL\_CHECKING\_TYPE, MODEL\_CHECKER\_TYPE\_APPROXIMATE\_BAYESIAN, MODEL\_CHECKER\_TYPE\_APPROXIMATE\_PROBABILISTIC, MODEL\_CHECKER\_TYPE\_BAYESIAN, MODEL\_CHECKER\_TYPE\_PROBABILISTIC\_BLACK\_BOX, MODEL\_CHECKER\_TYPE\_STATISTICAL, removeApproximateBayesianModelCheckingArguments(), removeApproximateProbabilisticModelCheckingArguments(), removeBayesianModelCheckingArguments(), and removeStatisticalModelCheckingArguments().

Referenced by areInvalidModelCheckingTypeSpecificArguments().

**6.48.3.46 void CommandLineModelChecking::removeOptionalArguments ( po::variables\_map & *variablesMap* ) [private]**

Remove the optional arguments from the given variables\_map.

### Parameters

<i>variables- Map</i>	The map containing all parsed command line arguments
---------------------------	--

Definition at line 237 of file CommandLineModelChecking.cpp.

References ARG\_EXTRA\_EVALUATION\_PROGRAM\_NAME\_LONG, ARG\_HELP\_NAME\_LONG, ARG\_MULTISCALE\_ARCHITECTURE\_GRAPH\_NAME\_LONG, and ARG\_VERBOSE\_NAME\_LONG.

Referenced by areInvalidModelCheckingArgumentsPresent().

---

**6.48.3.47 void CommandLineModelChecking::removeRequiredArguments ( po::variables\_map & variablesMap ) [private]**

Remove the required arguments from the given variables\_map.

**Parameters**

<i>variables- Map</i>	The map containing all parsed command line arguments
---------------------------	--

Definition at line 230 of file CommandLineModelChecking.cpp.

References ARG\_EXTRA\_EVALUATION\_TIME\_NAME\_LONG, ARG\_LOGIC\_QUESTIONS\_NAME\_LONG, ARG\_MODEL\_CHECKER\_TYPE\_NAME\_LONG, and ARG\_SPATIAL\_TEMPORAL\_TRACES\_NAME\_LONG.

Referenced by areInvalidModelCheckingArgumentsPresent().

**6.48.3.48 void CommandLineModelChecking::removeStatistical-  
ModelCheckingArguments ( po::variables\_map & variablesMap ) [private]**

Remove the statistical model checking arguments from the given variables\_map.

**Parameters**

<i>variables- Map</i>	The map containing all parsed command line arguments
---------------------------	--

Definition at line 349 of file CommandLineModelChecking.cpp.

References ARG\_TYPE\_I\_ERROR\_NAME\_LONG, and ARG\_TYPE\_II\_ERROR\_NAME\_LONG.

Referenced by removeModelCheckingTypeSpecificArguments().

#### 6.48.4 Member Data Documentation

**6.48.4.1 po::options\_description multiscale::verification::CommandLineModel-  
Checking::allowedArguments [private]**

The configuration indicating which command line arguments are allowed

Definition at line 52 of file CommandLineModelChecking.hpp.

Referenced by initialiseAllowedArgumentsConfiguration(), parseAndStoreArgumentsValues(), and printHelpContentsMessage().

6.48.4.2 `const std::string CommandLineModelChecking::ARG_APPROXIMATE_B-  
AYESIAN_ALPHA_DESCRIPTION = "the alpha shape parameter of the Beta  
distribution prior" [static, private]`

Definition at line 353 of file CommandLineModelChecking.hpp.

6.48.4.3 `const std::string CommandLineModelChecking::ARG_APPROXIMATE_BA-  
YESIAN_ALPHA_NAME_LONG = "approximate-bayesian-alpha" [static,  
private]`

Definition at line 352 of file CommandLineModelChecking.hpp.

Referenced by `areApproximateBayesianModelCheckingArgumentsPresent()`, `initialise-  
ApproximateBayesianModelChecker()`, `initialiseApproximateBayesianModelChecker-  
ArgumentsConfiguration()`, and `removeApproximateBayesianModelCheckingArguments()`.

6.48.4.4 `const std::string CommandLineModelChecking::ARG_APPROXIMATE_-  
BAYESIAN_BETA_DESCRIPTION = "the beta shape parameter of the Beta  
distribution prior" [static, private]`

Definition at line 356 of file CommandLineModelChecking.hpp.

6.48.4.5 `const std::string CommandLineModelChecking::ARG_APPROXIMATE_BA-  
YESIAN_BETA_NAME_LONG = "approximate-bayesian-beta" [static,  
private]`

Definition at line 355 of file CommandLineModelChecking.hpp.

Referenced by `areApproximateBayesianModelCheckingArgumentsPresent()`, `initialise-  
ApproximateBayesianModelChecker()`, `initialiseApproximateBayesianModelChecker-  
ArgumentsConfiguration()`, and `removeApproximateBayesianModelCheckingArguments()`.

6.48.4.6 `const std::string CommandLineModelChecking::ARG_BAYES_FACTOR_-  
_THRESHOLD_DESCRIPTION = "the Bayes factor threshold used to fix the  
confidence level of the answer" [static, private]`

Definition at line 350 of file CommandLineModelChecking.hpp.

Referenced by `initialiseBayesianModelCheckerArgumentsConfiguration()`.

6.48.4.7 `const std::string CommandLineModelChecking::ARG_BAYES_FACTOR_-  
_THRESHOLD_NAME_LONG = "bayes-factor-threshold" [static,  
private]`

Definition at line 349 of file CommandLineModelChecking.hpp.

Referenced by areBayesianModelCheckingArgumentsPresent(), initialiseBayesianModelChecker(), initialiseBayesianModelCheckerArgumentsConfiguration(), and removeBayesianModelCheckingArguments().

6.48.4.8 `const std::string CommandLineModelChecking::ARG_BAYESIAN_ALPHA_DESCRIPTION = "the alpha shape parameter of the Beta distribution prior"`  
[static, private]

Definition at line 344 of file CommandLineModelChecking.hpp.

Referenced by initialiseApproximateBayesianModelCheckerArgumentsConfiguration(), and initialiseBayesianModelCheckerArgumentsConfiguration().

6.48.4.9 `const std::string CommandLineModelChecking::ARG_BAYESIAN_ALPHA_NAME_LONG = "bayesian-alpha"` [static, private]

Definition at line 343 of file CommandLineModelChecking.hpp.

Referenced by areBayesianModelCheckingArgumentsPresent(), initialiseBayesianModelChecker(), initialiseBayesianModelCheckerArgumentsConfiguration(), and removeBayesianModelCheckingArguments().

6.48.4.10 `const std::string CommandLineModelChecking::ARG_BAYESIAN_BETA_DESCRIPTION = "the beta shape parameter of the Beta distribution prior"`  
[static, private]

Definition at line 347 of file CommandLineModelChecking.hpp.

Referenced by initialiseApproximateBayesianModelCheckerArgumentsConfiguration(), and initialiseBayesianModelCheckerArgumentsConfiguration().

6.48.4.11 `const std::string CommandLineModelChecking::ARG_BAYESIAN_BETA_NAME_LONG = "bayesian-beta"` [static, private]

Definition at line 346 of file CommandLineModelChecking.hpp.

Referenced by areBayesianModelCheckingArgumentsPresent(), initialiseBayesianModelChecker(), initialiseBayesianModelCheckerArgumentsConfiguration(), and removeBayesianModelCheckingArguments().

6.48.4.12 `const std::string CommandLineModelChecking::ARG_DELTA_DESCRIPTION = "the upper bound on the probability to deviate from the true probability"`  
[static, private]

Definition at line 338 of file CommandLineModelChecking.hpp.

Referenced by initialiseApproximateProbabilisticModelCheckerArgumentsConfiguration().

**6.48.4.13 const std::string CommandLineModelChecking::ARG\_DELTA\_NAME\_LONG = "delta" [static, private]**

Definition at line 337 of file CommandLineModelChecking.hpp.

Referenced by areApproximateProbabilisticModelCheckingArgumentsPresent(), initialiseApproximateProbabilisticModelChecker(), initialiseApproximateProbabilisticModelCheckerArgumentsConfiguration(), and removeApproximateProbabilisticModelCheckingArguments().

**6.48.4.14 const std::string CommandLineModelChecking::ARG\_EPSILON\_DESC-RIPTION = "the considered deviation from the true probability" [static, private]**

Definition at line 341 of file CommandLineModelChecking.hpp.

Referenced by initialiseApproximateProbabilisticModelCheckerArgumentsConfiguration().

**6.48.4.15 const std::string CommandLineModelChecking::ARG\_EPSILON\_NAME\_LONG = "epsilon" [static, private]**

Definition at line 340 of file CommandLineModelChecking.hpp.

Referenced by areApproximateProbabilisticModelCheckingArgumentsPresent(), initialiseApproximateProbabilisticModelChecker(), initialiseApproximateProbabilisticModelCheckerArgumentsConfiguration(), and removeApproximateProbabilisticModelCheckingArguments().

**6.48.4.16 const std::string CommandLineModelChecking::ARG\_EXTRA\_EVALUATION\_PROGRAM\_DESCRIPTION = "the program which will be executed whenever extra evaluation (and input traces) is required" [static, private]**

Definition at line 321 of file CommandLineModelChecking.hpp.

Referenced by initialiseOptionalArgumentsConfiguration().

**6.48.4.17 const std::string CommandLineModelChecking::ARG\_EXTRA\_EVALUATION\_PROGRAM\_NAME\_BOTH = ",p" [static, private]**

Definition at line 320 of file CommandLineModelChecking.hpp.

Referenced by initialiseOptionalArgumentsConfiguration().

---

6.48.4.18 `const std::string CommandLineModelChecking::ARG_EXTRA_EVALUATION_PROGRAM_NAME_LONG = "extra-evaluation-program" [static, private]`

Definition at line 319 of file CommandLineModelChecking.hpp.

Referenced by initialiseOptionalArgumentsDependentClassMembers(), and removeOptionalArguments().

6.48.4.19 `const std::string CommandLineModelChecking::ARG_EXTRA_EVALUATION_TIME_DESCRIPTION = "the maximum number of minutes the application can wait before finishing evaluation" [static, private]`

Definition at line 309 of file CommandLineModelChecking.hpp.

Referenced by initialiseRequiredArgumentsConfiguration().

6.48.4.20 `const std::string CommandLineModelChecking::ARG_EXTRA_EVALUATION_TIME_NAME_BOTH = ",e" [static, private]`

Definition at line 308 of file CommandLineModelChecking.hpp.

Referenced by initialiseRequiredArgumentsConfiguration().

6.48.4.21 `const std::string CommandLineModelChecking::ARG_EXTRA_EVALUATION_TIME_NAME_LONG = "extra-evaluation-time" [static, private]`

Definition at line 307 of file CommandLineModelChecking.hpp.

Referenced by initialiseRequiredArgumentsDependentClassMembers(), and removeRequiredArguments().

6.48.4.22 `const std::string CommandLineModelChecking::ARG_HELP_DESCRIPTION = "display help message (describing the meaning and usage of each command line argument)" [static, private]`

Definition at line 317 of file CommandLineModelChecking.hpp.

Referenced by initialiseOptionalArgumentsConfiguration().

6.48.4.23 `const std::string CommandLineModelChecking::ARG_HELP_NAME_BOTH = ",h" [static, private]`

Definition at line 316 of file CommandLineModelChecking.hpp.

Referenced by initialiseOptionalArgumentsConfiguration().

```
6.48.4.24 const std::string CommandLineModelChecking::-
ARG_HELP_NAME_LONG = "help" [static,
private]
```

Definition at line 315 of file CommandLineModelChecking.hpp.

Referenced by isHelpArgumentPresent(), and removeOptionalArguments().

```
6.48.4.25 const std::string CommandLineModelChecking::ARG_LOGIC_QUERIES_-
DESCRIPTION = "the path to the spatio-temporal queries input file" [static,
private]
```

Definition at line 301 of file CommandLineModelChecking.hpp.

Referenced by initialiseRequiredArgumentsConfiguration().

```
6.48.4.26 const std::string CommandLineModelChecking::ARG-
_LOGIC_QUERIES_NAME_BOTH = ",q" [static,
private]
```

Definition at line 300 of file CommandLineModelChecking.hpp.

Referenced by initialiseRequiredArgumentsConfiguration().

```
6.48.4.27 const std::string CommandLineModelChecking::ARG_LOG-
IC_QUERIES_NAME_LONG = "logic-queries" [static,
private]
```

Definition at line 299 of file CommandLineModelChecking.hpp.

Referenced by initialiseRequiredArgumentsDependentClassMembers(), and removeRequiredArguments().

```
6.48.4.28 const std::string CommandLineModelChecking::ARG_MODEL_CHECKE-
R_TYPE_DESCRIPTION = "the type of the model checker (0 = Probabilistic
black-box, 1 = Frequentist statistical, 2 = Frequentist approximate probabilistic
(Chernoff-Hoeffding), 3 = Bayesian (statistical hypothesis testing), 4 = Approximate
Bayesian (mean and variance estimation))" [static, private]
```

Definition at line 313 of file CommandLineModelChecking.hpp.

Referenced by initialiseRequiredArgumentsConfiguration().

```
6.48.4.29 const std::string CommandLineModelChecking::ARG_MO-
DEL_CHECKER_TYPE_NAME_BOTH = ",m" [static,
private]
```

Definition at line 312 of file CommandLineModelChecking.hpp.

---

Referenced by initialiseRequiredArgumentsConfiguration().

```
6.48.4.30 const std::string CommandLineModelChecking::ARG_MODEL_CHECKER_TYPE_NAME_LONG = "model-checker-type" [static, private]
```

Definition at line 311 of file CommandLineModelChecking.hpp.

Referenced by areInvalidModelCheckingArgumentsPresent(), initialiseRequiredArgumentsDependentClassMembers(), and removeRequiredArguments().

```
6.48.4.31 const std::string CommandLineModelChecking::ARG_MULTISCALE_ARCHITECTURE_GRAPH_DESCRIPTION = "the multiscale architecture graph encoding the hierarchical structure of the considered system" [static, private]
```

Definition at line 325 of file CommandLineModelChecking.hpp.

Referenced by initialiseOptionalArgumentsConfiguration().

```
6.48.4.32 const std::string CommandLineModelChecking::ARG_MULTISCALE_ARCHITECTURE_GRAPH_NAME_BOTH = ",a" [static, private]
```

Definition at line 324 of file CommandLineModelChecking.hpp.

Referenced by initialiseOptionalArgumentsConfiguration().

```
6.48.4.33 const std::string CommandLineModelChecking::ARG_MULTISCALE_ARCHITECTURE_GRAPH_NAME_LONG = "multiscale-architecture-graph" [static, private]
```

Definition at line 323 of file CommandLineModelChecking.hpp.

Referenced by initialiseOptionalArgumentsDependentClassMembers(), and removeOptionalArguments().

```
6.48.4.34 const std::string CommandLineModelChecking::ARG_SPATIAL_TEMPORAL_TRACES_DESCRIPTION = "the path to the folder containing spatio-temporal traces" [static, private]
```

Definition at line 305 of file CommandLineModelChecking.hpp.

Referenced by initialiseRequiredArgumentsConfiguration().

```
6.48.4.35 const std::string CommandLineModelChecking::ARG_SPATIAL_TEMPORAL_TRACES_NAME_BOTH = ",t" [static, private]
```

Definition at line 304 of file CommandLineModelChecking.hpp.

Referenced by initialiseRequiredArgumentsConfiguration().

```
6.48.4.36 const std::string CommandLineModelChecking::ARG_SPATIAL_TEMPORAL_TRACES_NAME_LONG = "spatial-temporal-traces" [static, private]
```

Definition at line 303 of file CommandLineModelChecking.hpp.

Referenced by initialiseRequiredArgumentsDependentClassMembers(), and removeRequiredArguments().

```
6.48.4.37 const std::string CommandLineModelChecking::ARG_TYPE_I_ERROR_DESCRIPTION = "the probability of type I errors" [static, private]
```

Definition at line 332 of file CommandLineModelChecking.hpp.

Referenced by initialiseStatisticalModelCheckerArgumentsConfiguration().

```
6.48.4.38 const std::string CommandLineModelChecking::ARG_TYPE_I_ERROR_NAME_LONG = "type-I-error" [static, private]
```

Definition at line 331 of file CommandLineModelChecking.hpp.

Referenced by areStatisticalModelCheckingArgumentsPresent(), initialiseStatisticalModelChecker(), initialiseStatisticalModelCheckerArgumentsConfiguration(), and removeStatisticalModelCheckingArguments().

```
6.48.4.39 const std::string CommandLineModelChecking::ARG_TYPE_II_ERROR_DESCRIPTION = "the probability of type II errors" [static, private]
```

Definition at line 335 of file CommandLineModelChecking.hpp.

Referenced by initialiseStatisticalModelCheckerArgumentsConfiguration().

```
6.48.4.40 const std::string CommandLineModelChecking::ARG_TYPE_II_ERROR_NAME_LONG = "type-II-error" [static, private]
```

Definition at line 334 of file CommandLineModelChecking.hpp.

Referenced by `areStatisticalModelCheckingArgumentsPresent()`, `initialiseStatisticalModelChecker()`, `initialiseStatisticalModelCheckerArgumentsConfiguration()`, and `removeStatisticalModelCheckingArguments()`.

**6.48.4.41** `const std::string CommandLineModelChecking::ARG_VARIANCE_THRESHOLD_DESCRIPTION = "the variance threshold used to fix the confidence level of the answer" [static, private]`

Definition at line 359 of file `CommandLineModelChecking.hpp`.

Referenced by `initialiseApproximateBayesianModelCheckerArgumentsConfiguration()`.

**6.48.4.42** `const std::string CommandLineModelChecking::ARG_VARIANCE_THRESHOLD_NAME_LONG = "variance-threshold" [static, private]`

Definition at line 358 of file `CommandLineModelChecking.hpp`.

Referenced by `areApproximateBayesianModelCheckingArgumentsPresent()`, `initialiseApproximateBayesianModelChecker()`, `initialiseApproximateBayesianModelCheckerArgumentsConfiguration()`, and `removeApproximateBayesianModelCheckingArguments()`.

**6.48.4.43** `const std::string CommandLineModelChecking::ARG_VERBOSE_DESCRIPTION = "if this flag is set detailed evaluation results will be displayed" [static, private]`

Definition at line 329 of file `CommandLineModelChecking.hpp`.

Referenced by `initialiseOptionalArgumentsConfiguration()`.

**6.48.4.44** `const std::string CommandLineModelChecking::ARG_VERBOSE_NAME_BOTH = "v" [static, private]`

Definition at line 328 of file `CommandLineModelChecking.hpp`.

Referenced by `initialiseOptionalArgumentsConfiguration()`.

**6.48.4.45** `const std::string CommandLineModelChecking::ARG_VERBOSE_NAME_LONG = "verbose" [static, private]`

Definition at line 327 of file `CommandLineModelChecking.hpp`.

Referenced by `initialiseOptionalArgumentsDependentClassMembers()`, and `removeOptionalArguments()`.

```
6.48.4.46 const std::string CommandLineModelChecking::CONFIG_
    _CAPTION_ALLOWED_ARGUMENTS = "" [static,
    private]
```

Definition at line 407 of file CommandLineModelChecking.hpp.

```
6.48.4.47 const std::string CommandLineModelChecking::CONFIG_CAPTION_-
    APPROXIMATE_BAYESIAN_MODEL_CHECKER_ARGUMENTS =
    MODEL_CHECKER_APPROXIMATE_BAYESIAN_NAME [static,
    private]
```

Definition at line 416 of file CommandLineModelChecking.hpp.

Referenced by initialiseApproximateBayesianModelCheckerArgumentsConfiguration().

```
6.48.4.48 const std::string CommandLineModelChecking::CONFIG_CAPTION_A-
    PPROXIMATE_PROBABILISTIC_MODEL_CHECKER_ARGUMENTS
    = MODEL_CHECKER_APPROXIMATE_PROBABILISTIC_NAME
    [static, private]
```

Definition at line 414 of file CommandLineModelChecking.hpp.

Referenced by initialiseApproximateProbabilisticModelCheckerArgumentsConfiguration().

```
6.48.4.49 const std::string CommandLineModelChecking::CONFIG_C-
    APTION_BAYESIAN_MODEL_CHECKER_ARGUMENTS =
    MODEL_CHECKER_BAYESIAN_NAME [static, private]
```

Definition at line 415 of file CommandLineModelChecking.hpp.

Referenced by initialiseBayesianModelCheckerArgumentsConfiguration().

```
6.48.4.50 const std::string CommandLineModelChecking::CONFIG_CAPTION_MOD-
    EL_CHECKER_TYPE_SPECIFIC_ARGUMENTS = "MODEL CHECKING TYPE
    SPECIFIC ARGUMENTS" [static, private]
```

Definition at line 410 of file CommandLineModelChecking.hpp.

```
6.48.4.51 const std::string CommandLineModelChecking::CONFIG_CAPTION_-
    OPTIONAL_ARGUMENTS = "OPTIONAL ARGUMENTS" [static,
    private]
```

Definition at line 409 of file CommandLineModelChecking.hpp.

```
6.48.4.52 const std::string CommandLineModelChecking::CONFIG_CAPTION_-  
PROBABILISTIC_BLACK_BOX_MODEL_CHECKER_ARGUMENTS  
= MODEL_CHECKER_PROBABILISTIC_BLACK_BOX_NAME  
[static, private]
```

Definition at line 412 of file CommandLineModelChecking.hpp.

```
6.48.4.53 const std::string CommandLineModelChecking::CONFIG_CAPTION_-  
REQUIRED_ARGUMENTS = "REQUIRED ARGUMENTS" [static,  
private]
```

Definition at line 408 of file CommandLineModelChecking.hpp.

```
6.48.4.54 const std::string CommandLineModelChecking::CONFIG_CA-  
PTION_STATISTICAL_MODEL_CHECKER_ARGUMENTS =  
MODEL_CHECKER_STATISTICAL_NAME [static, private]
```

Definition at line 413 of file CommandLineModelChecking.hpp.

Referenced by initialiseStatisticalModelCheckerArgumentsConfiguration().

```
6.48.4.55 const std::string CommandLineModelChecking::ERR_INVALID_COMMAN-  
D_LINE_ARGUMENTS = "Invalid command line arguments were provided and the  
model checker execution was stopped." [static, private]
```

Definition at line 294 of file CommandLineModelChecking.hpp.

Referenced by initialise().

```
6.48.4.56 const std::string CommandLineModelChecking::ERR_INVALID_MODE-  
L_CHECKING_ARGUMENTS = "The command line arguments provided for  
the chosen model checking type are invalid. Please run Mule with the --help flag to  
determine which arguments you should use." [static, private]
```

Definition at line 295 of file CommandLineModelChecking.hpp.

Referenced by areInvalidModelCheckingArguments().

```
6.48.4.57 const std::string CommandLineModelChecking::ERR_INVALID_MODEL_-  
CHECKING_TYPE = "The provided model checking type is invalid. Please run  
Mule with the --help flag to determine which values you can use." [static,  
private]
```

Definition at line 297 of file CommandLineModelChecking.hpp.

Referenced by areModelCheckingTypeSpecificArgumentsPresent(), initialiseModel-  
Checker(), and removeModelCheckingTypeSpecificArguments().

**6.48.4.58 std::string multiscale::verification::CommandLine-  
ModelChecking::extraEvaluationProgramPath  
[private]**

The path to the program which will be executed whenever more traces are required

Definition at line 38 of file CommandLineModelChecking.hpp.

Referenced by initialiseModelCheckingManager(), and initialiseOptionalArguments-DependentClassMembers().

**6.48.4.59 unsigned long multiscale::verification::CommandLineModelChecking-  
::extraEvaluationTime [private]**

The number of minutes for which the application waits for new traces to be produced

Definition at line 35 of file CommandLineModelChecking.hpp.

Referenced by initialiseModelCheckingManager(), initialiseRequiredArguments-DependentClassMembers(), and printModelCheckingInitialisationMessage().

**6.48.4.60 const std::string CommandLineModelChecking::HELP\_AUTHOR\_LABEL =  
"AUTHOR:" [static, private]**

Definition at line 367 of file CommandLineModelChecking.hpp.

Referenced by printHelpClosingMessage().

**6.48.4.61 const std::string CommandLineModelChecking::HELP\_AUTHOR\_MSG =  
"The author of this software is Ovidiu Parvu." [static, private]**

Definition at line 368 of file CommandLineModelChecking.hpp.

Referenced by printHelpClosingMessage().

**6.48.4.62 const std::string CommandLineModelChecking::HEL-  
P\_COPYRIGHT\_LABEL = "COPYRIGHT:" [static,  
private]**

Definition at line 369 of file CommandLineModelChecking.hpp.

Referenced by printHelpClosingMessage().

**6.48.4.63 const std::string CommandLineModelChecking::HELP\_COPYRIGHT\_MSG  
= " Copyright Ovidiu Parvu 2014." [static, private]**

Definition at line 370 of file CommandLineModelChecking.hpp.

Referenced by printHelpClosingMessage().

---

```
6.48.4.64 const std::string CommandLineModelChecking::HELP_-  
DESCRIPTION_LABEL = "DESCRIPTION:" [static,  
private]
```

Definition at line 365 of file CommandLineModelChecking.hpp.

Referenced by printHelpIntroMessage().

```
6.48.4.65 const std::string CommandLineModelChecking::HELP_DESCRIPTION_-  
_MSG = " Mule is a multiscale multidimensional (spatial-temporal) approximate  
probabilistic meta-model checker. It can be used for two different types of applications.  
First of all Mule can be employed to validate logic properties against multidimensional  
multiscale models. Secondly it can be used in reverse mode as a method to query  
time series data generated by in vivo/vitro experiments. Properties of interest are  
formalised using a multiscale spatio-temporal logic and their validity is checked using  
Mule." [static, private]
```

Definition at line 366 of file CommandLineModelChecking.hpp.

Referenced by printHelpIntroMessage().

```
6.48.4.66 const std::string CommandLineModelChecking::HELP_NAME_LABEL =  
"NAME:" [static, private]
```

Definition at line 361 of file CommandLineModelChecking.hpp.

Referenced by printHelpIntroMessage().

```
6.48.4.67 const std::string CommandLineModelChecking::HELP_NAME_MSG  
= " Mule - Multiscale multidimensional meta-model checker" [static,  
private]
```

Definition at line 362 of file CommandLineModelChecking.hpp.

Referenced by printHelpIntroMessage().

```
6.48.4.68 const std::string CommandLineModelChecking::HELP_REPORTING_BUGS_LABEL =  
"REPORTING BUGS:" [static,  
private]
```

Definition at line 371 of file CommandLineModelChecking.hpp.

Referenced by printHelpClosingMessage().

```
6.48.4.69 const std::string CommandLineModelChecking::HELP_REPORTING_
    _BUGS_MSG = "Please send requests for fixing bugs or recommendations to
    <ovidiu.parvu[AT]gmail.com>" [static, private]
```

Definition at line 372 of file CommandLineModelChecking.hpp.

Referenced by printHelpClosingMessage().

```
6.48.4.70 const std::string CommandLineModelChecking::HELP_USAGE_LABEL =
    "USAGE:" [static, private]
```

Definition at line 363 of file CommandLineModelChecking.hpp.

Referenced by printHelpIntroMessage().

```
6.48.4.71 const std::string CommandLineModelChecking::HELP_USAGE_MSG
    = "Mule <required-arguments> [<optional-arguments>]
    <model-checking-type-specific-arguments>" [static, private]
```

Definition at line 364 of file CommandLineModelChecking.hpp.

Referenced by printHelpIntroMessage().

```
6.48.4.72 std::string multiscale::verification::CommandLineModelChecking::logic-
    QueriesFilepath [private]
```

The path to the logic queries file

Definition at line 27 of file CommandLineModelChecking.hpp.

Referenced by initialiseModelCheckingManager(), initialiseRequiredArgumentsDependentClassMembers(), and printModelCheckingInitialisationMessage().

```
6.48.4.73 const std::string CommandLineModelChecking::MODEL_CHECKER_APP-
    ROXIMATE_BAYESIAN_NAME = "Approximate Bayesian (mean and variance
    estimate)" [static, private]
```

Definition at line 401 of file CommandLineModelChecking.hpp.

Referenced by initialiseApproximateBayesianModelChecker().

```
6.48.4.74 const std::string CommandLineModelChecking::MODEL_CHECKER_APP-
    ROXIMATE_BAYESIAN_PARAMETERS_BEGIN = "Beta distribution prior
    shape parameters alpha = " [static, private]
```

Definition at line 402 of file CommandLineModelChecking.hpp.

Referenced by initialiseApproximateBayesianModelChecker().

```
6.48.4.75 const std::string CommandLineModelChecking::MODEL_CHECKER_-  
APPROXIMATE_BAYESIAN_PARAMETERS_END = "." [static,  
private]
```

Definition at line 405 of file CommandLineModelChecking.hpp.

Referenced by initialiseApproximateBayesianModelChecker().

```
6.48.4.76 const std::string CommandLineModelChecking::MODEL_CHECKER_A-  
PPROXIMATE_BAYESIAN_PARAMETERS_MIDDLE1 = " and beta = "  
[static, private]
```

Definition at line 403 of file CommandLineModelChecking.hpp.

Referenced by initialiseApproximateBayesianModelChecker().

```
6.48.4.77 const std::string CommandLineModelChecking::MODEL_CHECKER_APP-  
ROXIMATE_BAYESIAN_PARAMETERS_MIDDLE2 = ". Variance threshold = "  
[static, private]
```

Definition at line 404 of file CommandLineModelChecking.hpp.

Referenced by initialiseApproximateBayesianModelChecker().

```
6.48.4.78 const std::string CommandLineModelChecking::MODEL_CHECKER_APP-  
ROXIMATE_PROBABILISTIC_NAME = "Frequentist approximate probabilistic  
(Chernoff-Hoeffding)" [static, private]
```

Definition at line 390 of file CommandLineModelChecking.hpp.

Referenced by initialiseApproximateProbabilisticModelChecker().

```
6.48.4.79 const std::string CommandLineModelChecking::MODEL_CHECKER_APP-  
ROXIMATE_PROBABILISTIC_PARAMETERS_BEGIN = "Upper bound on  
probability to deviate more than epsilon = " [static, private]
```

Definition at line 391 of file CommandLineModelChecking.hpp.

Referenced by initialiseApproximateProbabilisticModelChecker().

```
6.48.4.80 const std::string CommandLineModelChecking::MODEL_CHECKER_APP-  
ROXIMATE_PROBABILISTIC_PARAMETERS_END = "." [static,  
private]
```

Definition at line 393 of file CommandLineModelChecking.hpp.

Referenced by initialiseApproximateProbabilisticModelChecker().

## **6.48 multiscale::verification::CommandLineModelChecking Class Reference 285**

---

```
6.48.4.81 const std::string CommandLineModelChecking::MODEL_CHECKER_APP-
ROXIMATE_PROBABILISTIC_PARAMETERS_MIDDLE = "from the true
probability is delta = " [static, private]
```

Definition at line 392 of file CommandLineModelChecking.hpp.

Referenced by initialiseApproximateProbabilisticModelChecker().

```
6.48.4.82 const std::string CommandLineModelChecking::MODEL_CHECKER_B-
AYESIAN_NAME = "Bayesian (statistical hypothesis testing)" [static,
private]
```

Definition at line 395 of file CommandLineModelChecking.hpp.

Referenced by initialiseBayesianModelChecker().

```
6.48.4.83 const std::string CommandLineModelChecking::MODEL_CHECKER_BAY-
ESIAN_PARAMETERS_BEGIN = "Beta distribution prior shape parameters alpha
= " [static, private]
```

Definition at line 396 of file CommandLineModelChecking.hpp.

Referenced by initialiseBayesianModelChecker().

```
6.48.4.84 const std::string CommandLineModelChecking::MODEL_CH-
ECKER_BAYESIAN_PARAMETERS_END = ":" [static,
private]
```

Definition at line 399 of file CommandLineModelChecking.hpp.

Referenced by initialiseBayesianModelChecker().

```
6.48.4.85 const std::string CommandLineModelChecking::MODEL_CHECKER_B-
AYESIAN_PARAMETERS_MIDDLE1 = " and beta = " [static,
private]
```

Definition at line 397 of file CommandLineModelChecking.hpp.

Referenced by initialiseBayesianModelChecker().

```
6.48.4.86 const std::string CommandLineModelChecking::MODEL_CHECKER_BAY-
ESIAN_PARAMETERS_MIDDLE2 = ". Bayes factor threshold = " [static,
private]
```

Definition at line 398 of file CommandLineModelChecking.hpp.

Referenced by initialiseBayesianModelChecker().

```
6.48.4.87 const std::string CommandLineModelChecking::MODEL_CHECKER_PRO-
BABILISTIC_BLACK_BOX_NAME = "Probabilistic black-box" [static,
private]
```

Definition at line 382 of file CommandLineModelChecking.hpp.

Referenced by initialiseProbabilisticBlackBoxModelChecker().

```
6.48.4.88 const std::string CommandLineModelChecking::MODEL_CHECKER_P-
ROBABILISTIC_BLACK_BOX_PARAMETERS = "None" [static,
private]
```

Definition at line 383 of file CommandLineModelChecking.hpp.

Referenced by initialiseProbabilisticBlackBoxModelChecker().

```
6.48.4.89 const std::string CommandLineModelChecking::MODEL_CHECK-
ER_STATISTICAL_NAME = "Frequentist statistical" [static,
private]
```

Definition at line 385 of file CommandLineModelChecking.hpp.

Referenced by initialiseStatisticalModelChecker().

```
6.48.4.90 const std::string CommandLineModelChecking::MODEL_CHECKER_STA-
TISTICAL_PARAMETERS_BEGIN = "Probability of type I errors (false negatives)
= " [static, private]
```

Definition at line 386 of file CommandLineModelChecking.hpp.

Referenced by initialiseStatisticalModelChecker().

```
6.48.4.91 const std::string CommandLineModelChecking::MODEL_CHECK-
ER_STATISTICAL_PARAMETERS_END = ":" [static,
private]
```

Definition at line 388 of file CommandLineModelChecking.hpp.

Referenced by initialiseStatisticalModelChecker().

```
6.48.4.92 const std::string CommandLineModelChecking::MODEL_CHECKER_STA-
TISTICAL_PARAMETERS_MIDDLE = " and of type II errors (false positives) = "
[static, private]
```

Definition at line 387 of file CommandLineModelChecking.hpp.

Referenced by initialiseStatisticalModelChecker().

```
6.48.4.93 const unsigned int multiscale::verification::CommandLineModel-
    Checking::MODEL_CHECKER_TYPE_APPROXIMATE_BAYESIAN = 4
    [static, private]
```

Definition at line 380 of file CommandLineModelChecking.hpp.

Referenced by areModelCheckingTypeSpecificArgumentsPresent(), initialiseModelChecker(), and removeModelCheckingTypeSpecificArguments().

```
6.48.4.94 const unsigned int multiscale::verification::CommandLineModelChecking-
    ::MODEL_CHECKER_TYPE_APPROXIMATE_PROBABILISTIC = 2
    [static, private]
```

Definition at line 378 of file CommandLineModelChecking.hpp.

Referenced by areModelCheckingTypeSpecificArgumentsPresent(), initialiseModelChecker(), and removeModelCheckingTypeSpecificArguments().

```
6.48.4.95 const unsigned int multiscale::verification::CommandLineModel-
    Checking::MODEL_CHECKER_TYPE_BAYESIAN = 3 [static,
    private]
```

Definition at line 379 of file CommandLineModelChecking.hpp.

Referenced by areModelCheckingTypeSpecificArgumentsPresent(), initialiseModelChecker(), and removeModelCheckingTypeSpecificArguments().

```
6.48.4.96 const unsigned int multiscale::verification::CommandLineModelChecking-
    ::MODEL_CHECKER_TYPE_PROBABILISTIC_BLACK_BOX = 0
    [static, private]
```

Definition at line 376 of file CommandLineModelChecking.hpp.

Referenced by areModelCheckingTypeSpecificArgumentsPresent(), initialiseModelChecker(), and removeModelCheckingTypeSpecificArguments().

```
6.48.4.97 const unsigned int multiscale::verification::CommandLineModel-
    Checking::MODEL_CHECKER_TYPE_STATISTICAL = 1 [static,
    private]
```

Definition at line 377 of file CommandLineModelChecking.hpp.

Referenced by areModelCheckingTypeSpecificArgumentsPresent(), initialiseModelChecker(), and removeModelCheckingTypeSpecificArguments().

**6.48.4.98 std::shared\_ptr<ModelCheckerFactory> multiscale::verification-  
::CommandLineModelChecking::modelCheckerFactory  
[private]**

The model checker

Definition at line 70 of file CommandLineModelChecking.hpp.

Referenced by execute(), initialiseApproximateBayesianModelChecker(), initialise-  
ApproximateProbabilisticModelChecker(), initialiseBayesianModelChecker(), initialise-  
ProbabilisticBlackBoxModelChecker(), and initialiseStatisticalModelChecker().

**6.48.4.99 std::string multiscale::verification::CommandLineModelChecking-  
::modelCheckerParameters [private]**

The parameters specific to the model checker

Definition at line 67 of file CommandLineModelChecking.hpp.

Referenced by initialiseApproximateBayesianModelChecker(), initialiseApproximate-  
ProbabilisticModelChecker(), initialiseBayesianModelChecker(), initialiseProbabilistic-  
BlackBoxModelChecker(), initialiseStatisticalModelChecker(), and printModelChecking-  
InitialisationMessage().

**6.48.4.100 unsigned int multiscale::verification::CommandLineModelChecking-  
::modelCheckerType [private]**

The type of the model checker

Definition at line 32 of file CommandLineModelChecking.hpp.

Referenced by areInvalidModelCheckingArgumentsPresent(), initialiseModelChecker(),  
and initialiseRequiredArgumentsDependentClassMembers().

**6.48.4.101 std::string multiscale::verification::CommandLineModelChecking-  
::modelCheckerTypeName [private]**

The name of the model checker type

Definition at line 65 of file CommandLineModelChecking.hpp.

Referenced by initialiseApproximateBayesianModelChecker(), initialiseApproximate-  
ProbabilisticModelChecker(), initialiseBayesianModelChecker(), initialiseProbabilistic-  
BlackBoxModelChecker(), initialiseStatisticalModelChecker(), and printModelChecking-  
InitialisationMessage().

**6.48.4.102 po::options\_description multiscale::verification::CommandLine-  
ModelChecking::modelCheckerTypeSpecificArguments  
[private]**

The configuration indicating which command line arguments are allowed

## **6.48 multiscale::verification::CommandLineModelChecking Class Reference 289**

---

Definition at line 61 of file CommandLineModelChecking.hpp.

Referenced by initialiseAllowedArgumentsConfiguration(), and initialiseModelCheckerTypeSpecificArgumentsConfiguration().

**6.48.4.103 std::shared\_ptr<ModelCheckingManager> multiscale::verification::CommandLineModelChecking::modelCheckingManager  
[private]**

The model checking task manager

Definition at line 72 of file CommandLineModelChecking.hpp.

Referenced by execute(), and initialiseModelCheckingManager().

**6.48.4.104 const std::string CommandLineModelChecking::MSG\_MODEL\_CHECKING\_HELP\_REQUESTED = "A request for displaying help information was issued" [static, private]**

Definition at line 374 of file CommandLineModelChecking.hpp.

Referenced by handleHelpRequest().

**6.48.4.105 std::string multiscale::verification::CommandLineModelChecking::multiscaleArchitectureGraphFilepath  
[private]**

The path to the multiscale architecture graph

Definition at line 42 of file CommandLineModelChecking.hpp.

Referenced by initialiseModelCheckingManager(), and initialiseOptionalArgumentsDependentClassMembers().

**6.48.4.106 po::options\_description multiscale::verification::CommandLineModelChecking::optionalArguments  
[private]**

The configuration indicating which command line arguments are allowed

Definition at line 58 of file CommandLineModelChecking.hpp.

Referenced by initialiseAllowedArgumentsConfiguration(), and initialiseOptionalArgumentsConfiguration().

**6.48.4.107 po::options\_description multiscale::verification::CommandLineModelChecking::requiredArguments  
[private]**

The configuration indicating which command line arguments are allowed

Definition at line 55 of file CommandLineModelChecking.hpp.

Referenced by initialiseAllowedArgumentsConfiguration(), and initialiseRequiredArgumentsConfiguration().

#### **6.48.4.108 bool multiscale::verification::CommandLineModelChecking::shouldVerboseDetailedResults [private]**

The flag indicating if detailed results should be printed out

Definition at line 45 of file CommandLineModelChecking.hpp.

Referenced by initialiseModelCheckingManager(), and initialiseOptionalArgumentsDependentClassMembers().

#### **6.48.4.109 std::string multiscale::verification::CommandLineModelChecking::tracesFolderPath [private]**

The path to the folder containing traces

Definition at line 29 of file CommandLineModelChecking.hpp.

Referenced by initialiseModelCheckingManager(), initialiseRequiredArgumentsDependentClassMembers(), and printModelCheckingInitialisationMessage().

#### **6.48.4.110 po::variables\_map multiscale::verification::CommandLineModelChecking::variablesMap [private]**

The map containing  $\langle a, v \rangle$  pairs where  $a$  = command line argument and  $v$  = value

Definition at line 49 of file CommandLineModelChecking.hpp.

Referenced by areInvalidModelCheckingArgumentsPresent(), areValidArgumentsConsideringConfiguration(), initialiseApproximateBayesianModelChecker(), initialiseApproximateProbabilisticModelChecker(), initialiseBayesianModelChecker(), initialiseOptionalArgumentsDependentClassMembers(), initialiseRequiredArgumentsDependentClassMembers(), initialiseStatisticalModelChecker(), isHelpArgumentPresent(), and parseAndStoreArgumentsValues().

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

- CommandLineModelChecking.hpp
- CommandLineModelChecking.cpp

### **6.49 multiscale::verification::ComparatorAttribute Class Reference**

Class for representing a comparator attribute.

```
#include <ComparatorAttribute.hpp>
```

### Public Attributes

- `ComparatorType comparatorType`

#### 6.49.1 Detailed Description

Class for representing a comparator attribute.

Definition at line 31 of file `ComparatorAttribute.hpp`.

#### 6.49.2 Member Data Documentation

##### 6.49.2.1 `ComparatorType multiscale::verification::ComparatorAttribute- ::comparatorType`

The comparator type

Definition at line 35 of file `ComparatorAttribute.hpp`.

Referenced by `multiscale::verification::LogicPropertyVisitor::evaluateChangeTemporal-NumericMeasure()`, `multiscale::verification::LogicPropertyVisitor::evaluateTemporal-NumericComparison()`, `multiscale::verification::ProbabilisticLogicPropertyAttribute::get-Comparator()`, and `multiscale::verification::ConstraintVisitor::operator()()`.

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

- `ComparatorAttribute.hpp`

## 6.50 multiscale::verification::ComparatorEvaluator Class Reference

Class for evaluating comparison expressions.

```
#include <ComparatorEvaluator.hpp>
```

### Static Public Member Functions

- template<typename T >  
`static bool evaluate (T lhsElement, const ComparatorType &comparator, T rhs-  
Element)`

*Compare two real valued elements considering a given comparator.*

- static bool `evaluate (const std::string &lhsScaleAndSubsystem, const -  
ComparatorType &comparator, const std::string &rhsScaleAndSubsystem, const  
MultiscaleArchitectureGraph &multiscaleArchitectureGraph)`

*Compare two scales and subsystems considering a given comparator and multiscale  
architecture graph.*

### 6.50.1 Detailed Description

Class for evaluating comparison expressions.

Definition at line 14 of file ComparatorEvaluator.hpp.

### 6.50.2 Member Function Documentation

**6.50.2.1 template<typename T > static bool multiscale::verification::ComparatorEvaluator::evaluate ( T *lhsElement*, const ComparatorType & *comparator*, T *rhsElement* ) [inline, static]**

Compare two real valued elements considering a given comparator.

#### Parameters

<i>lhsElement</i>	The element which is on the left hand side of the comparator
<i>comparator</i>	The comparator type used to compare the elements
<i>rhsElement</i>	The element which is on the right hand side of the comparator

Definition at line 25 of file ComparatorEvaluator.hpp.

References multiscale::Numeric::almostEqual(), multiscale::ERR\_UNDEFINED\_ENUM\_VALUE, multiscale::Numeric::greaterOrEqual(), and multiscale::Numeric::lessOrEqual().

Referenced by multiscale::verification::ConstraintVisitor::filterSpatialEntitiesWrtScaleAndSubsystemConsideringNonEqualComparator(), and multiscale::verification::ConstraintVisitor::filterSpatialEntitiesWrtSpatialMeasure().

**6.50.2.2 static bool multiscale::verification::ComparatorEvaluator::evaluate ( const std::string & *lhsScaleAndSubsystem*, const ComparatorType & *comparator*, const std::string & *rhsScaleAndSubsystem*, const MultiscaleArchitectureGraph & *multiscaleArchitectureGraph* ) [inline, static]**

Compare two scales and subsystems considering a given comparator and multiscale architecture graph.

#### Parameters

<i>lhsScale-And-Subsystem</i>	The left hand side scale and subsystem
<i>comparator</i>	The comparator type used to compare the elements
<i>rhsScale-And-Subsystem</i>	The right hand side scale and subsystem
<i>multiscale-Architecture-Graph</i>	The considered multiscale architecture graph which encodes relations between scales and subsystems

Definition at line 58 of file ComparatorEvaluator.hpp.

References multiscale::verification::ScaleAndSubsystemEvaluator::areEqualScalesAndSubsystems(), multiscale::ERR\_UNDEFINED\_ENUM\_VALUE, and multiscale::verification::MultiscaleArchitectureGraph::isScaleAndSubsystemSmallerThan().

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

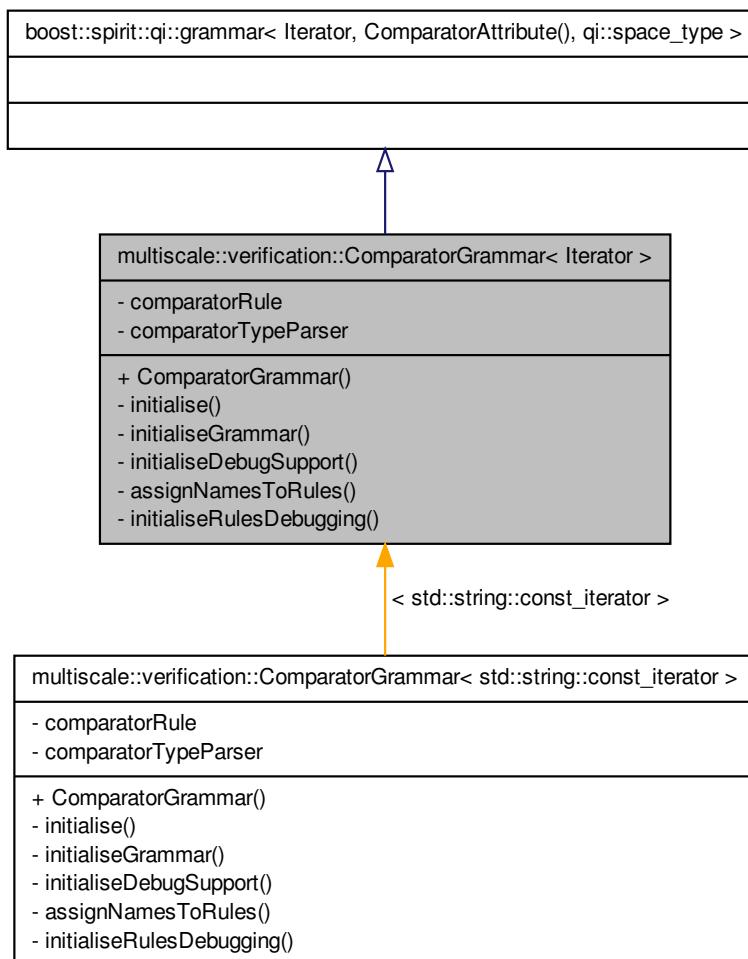
- ComparatorEvaluator.hpp

## **6.51 multiscale::verification::ComparatorGrammar< Iterator > - Class Template Reference**

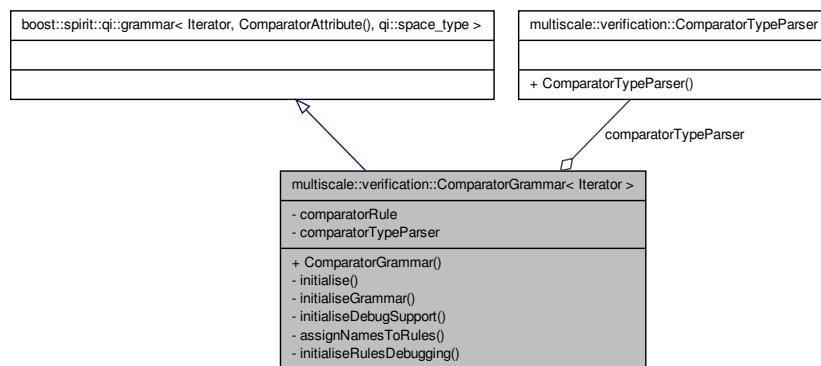
The grammar for parsing comparator statements.

```
#include <ComparatorGrammar.hpp>
```

Inheritance diagram for multiscale::verification::ComparatorGrammar< Iterator >:



Collaboration diagram for multiscale::verification::ComparatorGrammar< Iterator >:



## Public Member Functions

- [ComparatorGrammar \(\)](#)

## Private Member Functions

- void [initialise \(\)](#)  
*Initialisation function.*
- void [initialiseGrammar \(\)](#)  
*Initialise the grammar.*
- void [initialiseDebugSupport \(\)](#)  
*Initialise debug support.*
- void [assignNamesToRules \(\)](#)  
*Assign names to the rules.*
- void [initialiseRulesDebugging \(\)](#)  
*Initialise the debugging of rules.*

## Private Attributes

- [qi::rule< Iterator, ComparatorAttribute\(\), qi::space\\_type > comparatorRule](#)
- [ComparatorTypeParser comparatorTypeParser](#)

### 6.51.1 Detailed Description

```
template<typename Iterator>class multiscale::verification::ComparatorGrammar< Iterator >
```

The grammar for parsing comparator statements.

Definition at line 24 of file ComparatorGrammar.hpp.

### 6.51.2 Constructor & Destructor Documentation

```
6.51.2.1 template<typename Iterator > multiscale::verification::-
    ComparatorGrammar< Iterator >::ComparatorGrammar (
    )
```

Definition at line 24 of file ComparatorGrammarDefinition.hpp.

References multiscale::verification::ComparatorGrammar< Iterator >::initialise().

### 6.51.3 Member Function Documentation

```
6.51.3.1 template<typename Iterator > void multiscale::verification::-
    ComparatorGrammar< Iterator >::assignNamesToRules ( )
    [private]
```

Assign names to the rules.

Definition at line 54 of file ComparatorGrammarDefinition.hpp.

```
6.51.3.2 template<typename Iterator > void multiscale::verification::-
    ComparatorGrammar< Iterator >::initialise ( )
    [private]
```

Initialisation function.

Definition at line 31 of file ComparatorGrammarDefinition.hpp.

Referenced by multiscale::verification::ComparatorGrammar< Iterator >::ComparatorGrammar().

```
6.51.3.3 template<typename Iterator > void multiscale::verification::-
    ComparatorGrammar< Iterator >::initialiseDebugSupport ( )
    [private]
```

Initialise debug support.

Definition at line 45 of file ComparatorGrammarDefinition.hpp.

## **6.52 multiscale::verification::ComparatorNonEqualTypeParser Struct Reference**

---

**6.51.3.4 template<typename Iterator > void multiscale::verification::- ComparatorGrammar< Iterator >::initialiseGrammar( ) [private]**

Initialise the grammar.

Definition at line 38 of file ComparatorGrammarDefinition.hpp.

**6.51.3.5 template<typename Iterator > void multiscale::verification::- ComparatorGrammar< Iterator >::initialiseRulesDebugging( ) [private]**

Initialise the debugging of rules.

Definition at line 60 of file ComparatorGrammarDefinition.hpp.

### **6.51.4 Member Data Documentation**

**6.51.4.1 template<typename Iterator> qi::rule<Iterator, ComparatorAttribute(), qi::space\_type> multiscale::verification::ComparatorGrammar< Iterator >::comparatorRule [private]**

The rule for parsing a comparator

Definition at line 30 of file ComparatorGrammar.hpp.

**6.51.4.2 template<typename Iterator> ComparatorTypeParser multiscale::verification::ComparatorGrammar< Iterator >::comparatorTypeParser [private]**

The comparator type parser

Definition at line 35 of file ComparatorGrammar.hpp.

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

- ComparatorGrammar.hpp
- ComparatorGrammarDefinition.hpp

## **6.52 multiscale::verification::ComparatorNonEqualTypeParser - Struct Reference**

Symbol table and parser for the comparator type which does not accept the "=" symbol.

```
#include <SymbolTables.hpp>
```

## Public Member Functions

- [ComparatorNonEqualTypeParser \(\)](#)

### 6.52.1 Detailed Description

Symbol table and parser for the comparator type which does not accept the "=" symbol.  
Definition at line 83 of file SymbolTables.hpp.

### 6.52.2 Constructor & Destructor Documentation

#### 6.52.2.1 multiscale::verification::ComparatorNonEqual- TypeParser::ComparatorNonEqualTypeParser ( ) [inline]

Definition at line 86 of file SymbolTables.hpp.

References multiscale::verification::GreaterThanOrEqual, and multiscale::verification::-  
LessThanOrEqual.

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

- [SymbolTables.hpp](#)

## 6.53 multiscale::verification::ComparatorTypeParser Struct - Reference

Symbol table and parser for the comparator type.

```
#include <SymbolTables.hpp>
```

## Public Member Functions

- [ComparatorTypeParser \(\)](#)

### 6.53.1 Detailed Description

Symbol table and parser for the comparator type.

Definition at line 98 of file SymbolTables.hpp.

### 6.53.2 Constructor & Destructor Documentation

6.53.2.1 **multiscale::verification::ComparatorTypeParser::ComparatorTypeParser**  
( ) [inline]

Definition at line 101 of file SymbolTables.hpp.

References multiscale::verification::GreaterThanOrEqual, and multiscale::verification::LessThanOrEqual.

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

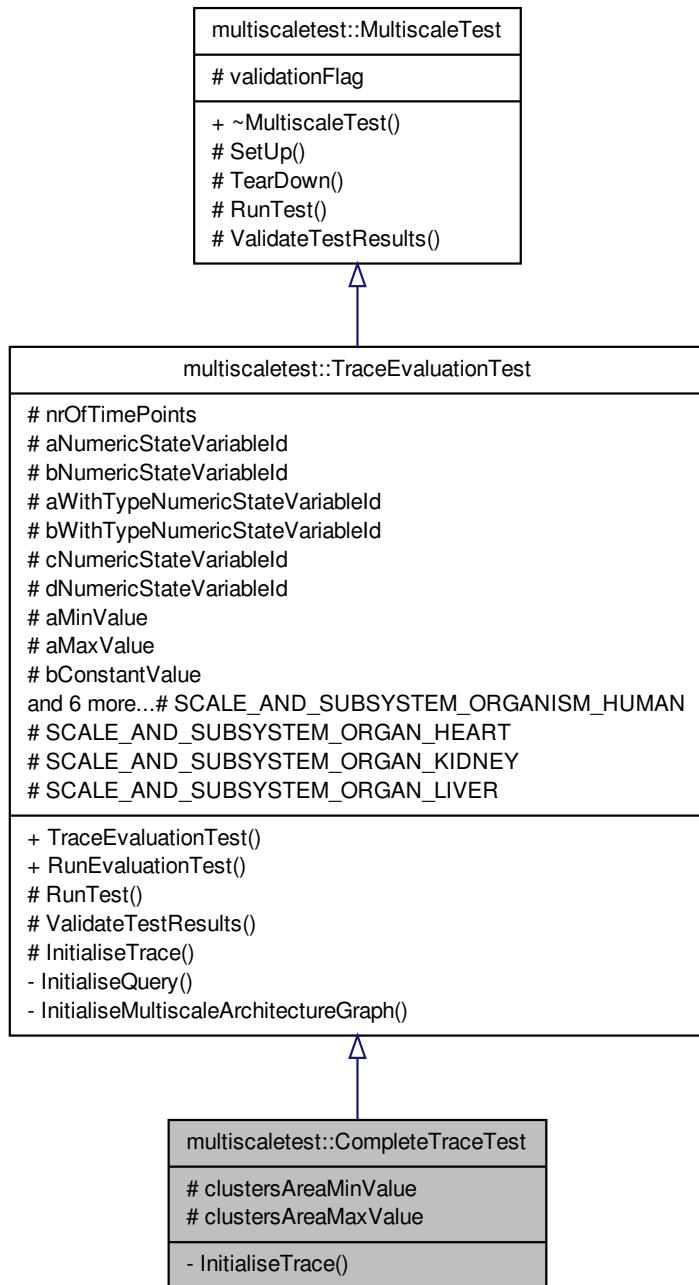
- SymbolTables.hpp

## 6.54 multiscaletest::CompleteTraceTest Class Reference

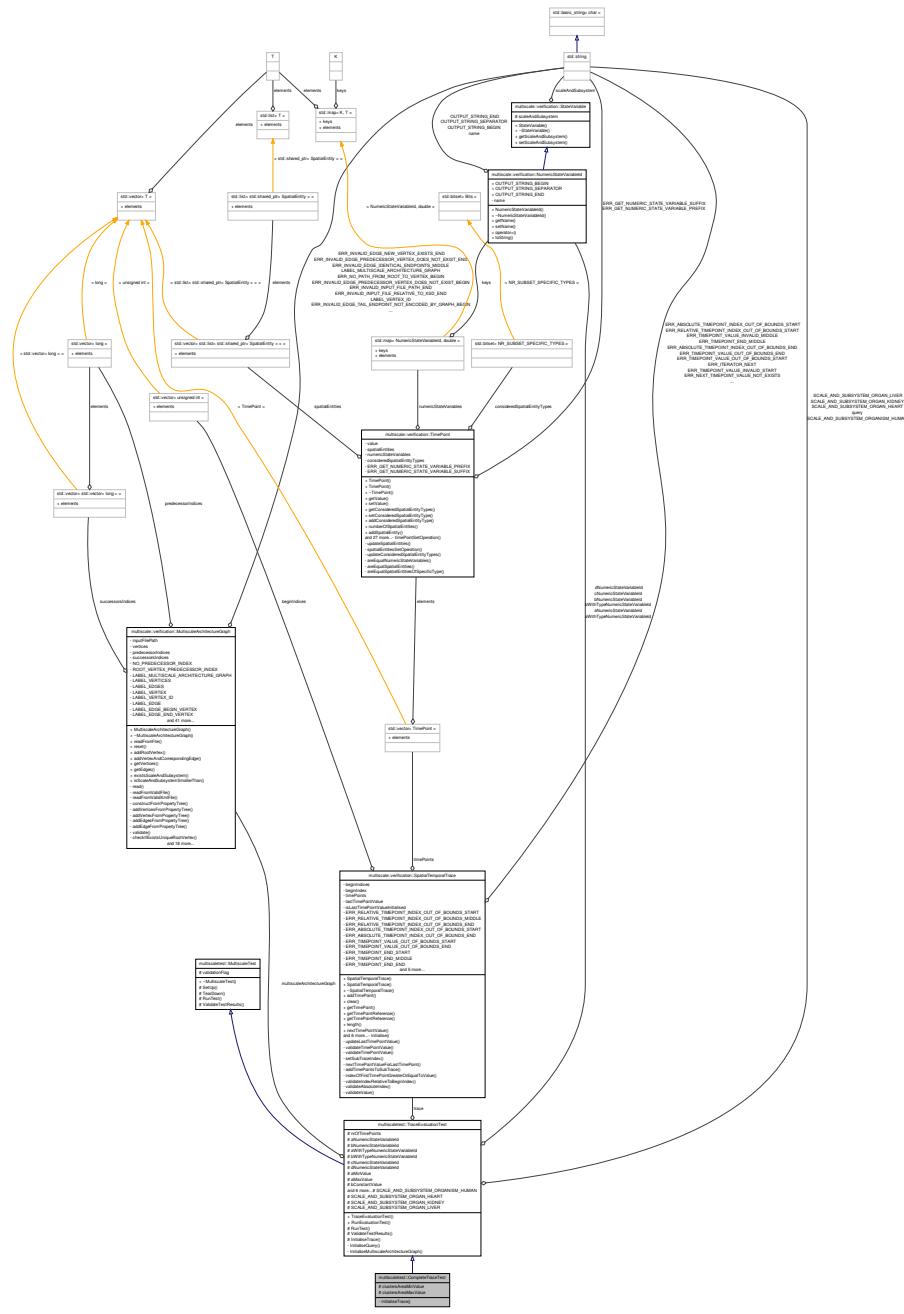
Class for testing evaluation of complete traces containing both numeric state variables and spatial entities.

```
#include <CompleteTraceTest.hpp>
```

Inheritance diagram for multiscaletest::CompleteTraceTest:



## Collaboration diagram for multiscaletest::CompleteTraceTest:



## Protected Attributes

- double clustersAreaMinValue

- double [clustersAreaMaxValue](#)

### Private Member Functions

- virtual void [InitialiseTrace](#) () override

*Initialise the trace.*

#### 6.54.1 Detailed Description

Class for testing evaluation of complete traces containing both numeric state variables and spatial entities.

Definition at line 27 of file CompleteTraceTest.hpp.

#### 6.54.2 Member Function Documentation

##### 6.54.2.1 void [multiscaletest::CompleteTraceTest::InitialiseTrace](#) ( ) [override, private, virtual]

Initialise the trace.

Implements [multiscaletest::TraceEvaluationTest](#).

Definition at line 41 of file CompleteTraceTest.hpp.

#### 6.54.3 Member Data Documentation

##### 6.54.3.1 double [multiscaletest::CompleteTraceTest::clustersAreaMaxValue](#) [protected]

The maximum area value for the cluster spatial entity type

Definition at line 32 of file CompleteTraceTest.hpp.

##### 6.54.3.2 double [multiscaletest::CompleteTraceTest::clustersAreaMinValue](#) [protected]

The minimum area value for the cluster spatial entity type

Definition at line 31 of file CompleteTraceTest.hpp.

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

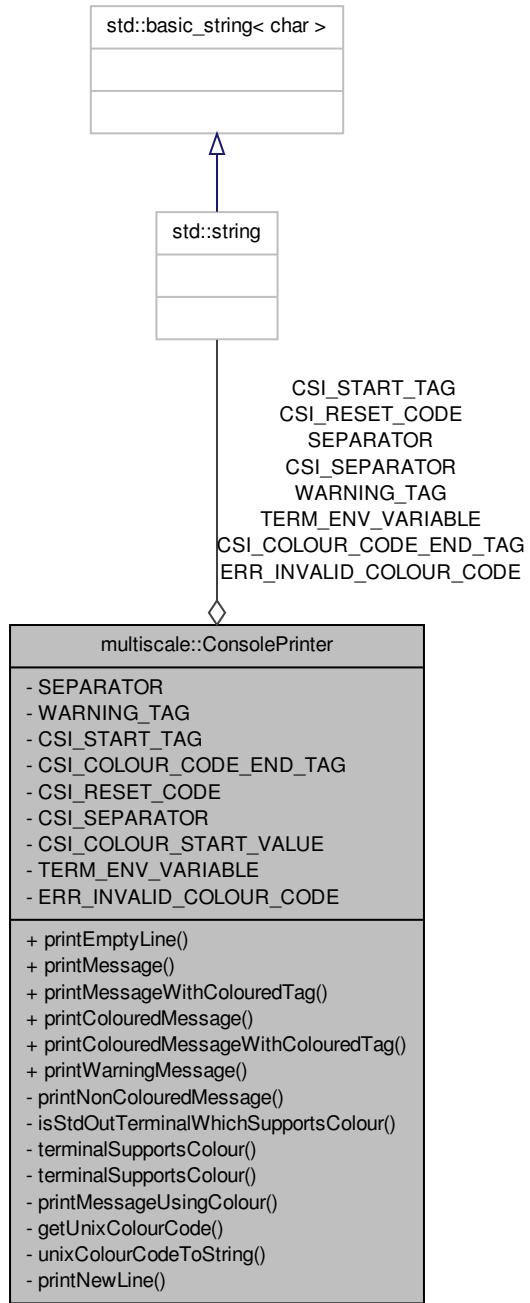
- CompleteTraceTest.hpp

## 6.55 multiscale::ConsolePrinter Class Reference

Class used to print (coloured) messages to the console.

```
#include <ConsolePrinter.hpp>
```

Collaboration diagram for multiscale::ConsolePrinter:



## Static Public Member Functions

- static void [printEmptyLine \(\)](#)  
*Print a new empty line.*
- static void [printMessage \(const std::string &message\)](#)  
*Print a message to the standard output.*
- static void [printMessageWithColouredTag \(const std::string &message, const std::string &tag, const ColourCode &tagColour\)](#)  
*Print a message with a coloured tag to the standard output.*
- static void [printColouredMessage \(const std::string &message, const ColourCode &colourCode\)](#)  
*Print a coloured message to the standard output.*
- static void [printColouredMessageWithColouredTag \(const std::string &message, const ColourCode &messageColour, const std::string &tag, const ColourCode &tagColour\)](#)  
*Print a coloured message with a coloured tag to the standard output.*
- static void [printWarningMessage \(const std::string &message\)](#)  
*Print a warning containing the given message string to the standard output.*

## Static Private Member Functions

- static void [printNonColouredMessage \(const std::string &message, bool appendNewLineAtEnd=true\)](#)  
*Print a (non-coloured) message to the standard output.*
- static bool [isStdOutTerminalWhichSupportsColour \(\)](#)  
*Check if the standard output is a terminal which supports colour.*
- static bool [terminalSupportsColour \(bool isTerminal\)](#)  
*Check if the terminal supports colour.*
- static bool [terminalSupportsColour \(\)](#)  
*Check if the terminal supports colour.*
- static void [printMessageUsingColour \(const std::string &message, const ColourCode &colourCode, bool appendNewLineAtEnd=true\)](#)  
*Print a coloured message to the standard output.*
- static std::string [getUnixColourCode \(const UnixColourCode &unixColourCode\)](#)  
*Get the CSI string representation corresponding to the given UNIX colour code.*
- static std::string [unixColourCodeToString \(const UnixColourCode &unixColourCode\)](#)  
*Get the string representation corresponding to the given UNIX colour code.*
- static void [printNewLine \(bool shouldPrint=true\)](#)  
*Get the CSI string representation for resetting all attributes (including colour)*

## Static Private Attributes

- static const std::string **SEPARATOR** = " "
- static const std::string **WARNING\_TAG** = "[ WARNING ]"
- static const std::string **CSI\_START\_TAG** = "\033["
- static const std::string **CSI\_COLOUR\_CODE\_END\_TAG** = "m"
- static const std::string **CSI\_RESET\_CODE** = "0"
- static const std::string **CSI\_SEPARATOR** = ";"
- static const int **CSI\_COLOUR\_START\_VALUE** = 30
- static const std::string **TERM\_ENV\_VARIABLE** = "TERM"
- static const std::string **ERR\_INVALID\_COLOUR\_CODE** = "The provided colour code is invalid. Please provide a valid colour code instead (see documentation for more details)."

### 6.55.1 Detailed Description

Class used to print (coloured) messages to the console.

Definition at line 57 of file ConsolePrinter.hpp.

### 6.55.2 Member Function Documentation

#### 6.55.2.1 std::string ConsolePrinter::getUnixColourCode ( const UnixColourCode & unixColourCode ) [static, private]

Get the CSI string representation corresponding to the given UNIX colour code.

##### Parameters

<code>unixColour- Code</code>	The given UNIX colour code
-----------------------------------	----------------------------

Definition at line 205 of file ConsolePrinter.cpp.

References `CSI_COLOUR_CODE_END_TAG`, `CSI_RESET_CODE`, `CSI_SEPARAT-OR`, `CSI_START_TAG`, and `unixColourCodeToString()`.

Referenced by `printMessageUsingColour()`.

#### 6.55.2.2 bool ConsolePrinter::isStdOutTerminalWhichSupportsColour ( ) [static, private]

Check if the standard output is a terminal which supports colour.

Definition at line 76 of file ConsolePrinter.cpp.

References `terminalSupportsColour()`.

Referenced by `printColouredMessage()`, `printColouredMessageWithColouredTag()`, and `printMessageWithColouredTag()`.

### 6.55.2.3 void ConsolePrinter::printColouredMessage ( const std::string & *message*, const ColourCode & *colourCode* ) [static]

Print a coloured message to the standard output.

The message will be printed in colour if and only if the standard output is a terminal. Otherwise it will be printed without changing colour.

#### Parameters

<i>message</i>	The given message
<i>colourCode</i>	The colour code used for printing the message

Definition at line 42 of file ConsolePrinter.cpp.

References isStdOutTerminalWhichSupportsColour(), printMessageUsingColour(), and printNonColouredMessage().

Referenced by multiscale::verification::ModelCheckingOutputWriter::printDetailedEvaluationResultsForLogicProperties(), multiscale::verification::ModelCheckingOutputWriter::printDetailedEvaluationResultsIntroductionMessage(), multiscale::verification::ModelCheckingOutputWriter::printEvaluationResultsSummary(), multiscale::verification::ModelCheckingOutputWriter::printFailedMessage(), multiscale::verification::ModelCheckingOutputWriter::printInitialisationMessage(), multiscale::verification::ModelCheckingOutputWriter::printIntroductionMessage(), multiscale::verification::ModelCheckingOutputWriter::printModelCheckingResultsIntroductionMessage(), multiscale::verification::ModelCheckingOutputWriter::printParsingLogicPropertiesBeginMessage(), multiscale::verification::ModelCheckingOutputWriter::printResultTag(), multiscale::verification::ModelCheckingOutputWriter::printSeparatorTag(), multiscale::verification::ModelCheckingOutputWriter::printStartModelCheckingExecutionMessage(), and multiscale::verification::ModelCheckingOutputWriter::printSuccessMessage().

### 6.55.2.4 void ConsolePrinter::printColouredMessageWithColouredTag ( const std::string & *message*, const ColourCode & *messageColour*, const std::string & *tag*, const ColourCode & *tagColour* ) [static]

Print a coloured message with a coloured tag to the standard output.

#### Parameters

<i>message</i>	The given message
<i>message-Colour</i>	The colour of the given message
<i>tag</i>	The given tag
<i>tagColour</i>	The colour of the given tag

Definition at line 51 of file ConsolePrinter.cpp.

References isStdOutTerminalWhichSupportsColour(), printMessageUsingColour(), printNonColouredMessage(), and SEPARATOR.

---

**6.55.2.5 void ConsolePrinter::printEmptyLine( ) [static]**

Print a new empty line.

Definition at line 22 of file ConsolePrinter.cpp.

References print.NewLine().

Referenced by multiscale::verification::ModelCheckingOutputWriter::printDetailedEvaluationResultsIntroductionMessage(), multiscale::verification::ModelCheckingOutputWriter::printInitialisationMessage(), multiscale::verification::ModelCheckingOutputWriter::printIntroductionMessage(), multiscale::verification::ModelCheckingOutputWriter::printModelCheckingResultsIntroductionMessage(), and multiscale::verification::ModelCheckingOutputWriter::printParsingLogicPropertiesEndMessage().

**6.55.2.6 void ConsolePrinter::printMessage ( const std::string & *message* ) [static]**

Print a message to the standard output.

**Parameters**

<i>message</i>	The given message
----------------	-------------------

Definition at line 26 of file ConsolePrinter.cpp.

References printNonColouredMessage().

**6.55.2.7 void ConsolePrinter::printMessageUsingColour ( const std::string & *message*, const ColourCode & *colourCode*, bool *appendNewLineAtEnd* = true ) [static, private]**

Print a coloured message to the standard output.

The message will be printed in colour if and only if the standard output is a terminal. Otherwise it will be printed using default colour.

**Parameters**

<i>message</i>	The given message
<i>colourCode</i>	The given colour code
<i>appendNewLineAtEnd</i>	Flag indicating if a new line character should be printed in the end

Definition at line 111 of file ConsolePrinter.cpp.

References getUnixColourCode(), and print.NewLine().

Referenced by printColouredMessage(), printColouredMessageWithColouredTag(), and printMessageWithColouredTag().

**6.55.2.8 void ConsolePrinter::printMessageWithColouredTag ( const std::string & message, const std::string & tag, const ColourCode & tagColour ) [static]**

Print a message with a coloured tag to the standard output.

**Parameters**

<i>message</i>	The given message
<i>tag</i>	The given tag
<i>tagColour</i>	The colour of the tag

Definition at line 30 of file ConsolePrinter.cpp.

References `isStdOutTerminalWhichSupportsColour()`, `printMessageUsingColour()`, `printNonColouredMessage()`, and `SEPARATOR`.

Referenced by `multiscale::verification::ModelCheckingOutputWriter::printDetailedEvaluationResultsIntroductionMessage()`, `multiscale::verification::ModelCheckingOutputWriter::printEvaluationResultsSummary()`, `multiscale::verification::ModelCheckingOutputWriter::printExecuteExtraEvaluationProgramMessage()`, `multiscale::verification::ModelCheckingOutputWriter::printInitialisationMessage()`, `multiscale::verification::ModelCheckingOutputWriter::printIntroductionMessage()`, `multiscale::verification::ModelCheckingOutputWriter::printLogicPropertyWithTag()`, `multiscale::verification::ModelCheckingOutputWriter::printModelCheckingResultsIntroductionMessage()`, `multiscale::verification::ModelCheckingOutputWriter::printParsingLogicPropertiesBeginMessage()`, `multiscale::verification::ModelCheckingOutputWriter::printParsingLogicPropertyMessage()`, `multiscale::verification::ModelCheckingOutputWriter::printStartModelCheckingExecutionMessage()`, `multiscale::verification::ModelCheckingOutputWriter::printStartTraceEvaluationMessage()`, `multiscale::verification::ModelCheckingOutputWriter::printTimeoutMessage()`, `multiscale::verification::ModelCheckingOutputWriter::printTruthValueDependentMessage()`, and `printWarningMessage()`.

**6.55.2.9 void ConsolePrinter::print.NewLine ( bool *shouldPrint* = true ) [static, private]**

Get the CSI string representation for resetting all attributes (including colour)

Print new line character if `shouldPrint` flag is true

**Parameters**

<i>shouldPrint</i>	Flag indicating if a new line character should be printed to the console
--------------------	--

Definition at line 224 of file ConsolePrinter.cpp.

Referenced by `printEmptyLine()`, `printMessageUsingColour()`, and `printNonColouredMessage()`.

---

**6.55.2.10 void ConsolePrinter::printNonColouredMessage ( const std::string & message, bool appendNewLineAtEnd = true ) [static, private]**

Print a (non-coloured) message to the standard output.

**Parameters**

<i>message</i>	The given message
<i>appendNewLineAtEnd</i>	Flag indicating if a new line character should be printed in the end

Definition at line 69 of file ConsolePrinter.cpp.

References print.NewLine().

Referenced by printColouredMessage(), printColouredMessageWithColouredTag(), printMessage(), and printMessageWithColouredTag().

**6.55.2.11 void ConsolePrinter::printWarningMessage ( const std::string & message ) [static]**

Print a warning containing the given message string to the standard output.

**Parameters**

<i>message</i>	The given message
----------------	-------------------

Definition at line 65 of file ConsolePrinter.cpp.

References printMessageWithColouredTag(), and WARNING\_TAG.

Referenced by multiscale::OperatingSystem::executeProgram(), multiscale::Numeric::numberInverse(), multiscale::verification::LogicPropertyVisitor::printExceptionMessage(), and multiscale::Numeric::printNoValuesWarningMessage().

**6.55.2.12 bool ConsolePrinter::terminalSupportsColour ( bool isTerminal ) [static, private]**

Check if the terminal supports colour.

**Parameters**

<i>isTerminal</i>	Flag indicating if the standard output is a terminal
-------------------	--

Definition at line 89 of file ConsolePrinter.cpp.

References terminalSupportsColour().

6.55.2.13 `bool ConsolePrinter::terminalSupportsColour( ) [static, private]`

Check if the terminal supports colour.

Assumption: Standard output is a terminal

Definition at line 97 of file ConsolePrinter.cpp.

References multiscale::OperatingSystem::getEnvironmentVariable(), and TERM\_ENV\_VARIABLE.

Referenced by isStdOutTerminalWhichSupportsColour(), and terminalSupportsColour().

6.55.2.14 `std::string ConsolePrinter::unixColourCodeToString( const UnixColourCode & unixColourCode ) [static, private]`

Get the string representation corresponding to the given UNIX colour code.

#### Parameters

<code>unixColour- Code</code>	The given UNIX colour code
-----------------------------------	----------------------------

Definition at line 210 of file ConsolePrinter.cpp.

References CSI\_COLOUR\_START\_VALUE, and multiscale::StringManipulator::toString().

Referenced by getUnixColourCode().

### 6.55.3 Member Data Documentation

6.55.3.1 `const std::string ConsolePrinter::CSI_COLOUR_CODE_END_TAG = "m" [static, private]`

Definition at line 187 of file ConsolePrinter.hpp.

Referenced by getUnixColourCode().

6.55.3.2 `const int ConsolePrinter::CSI_COLOUR_START_VALUE = 30 [static, private]`

Definition at line 191 of file ConsolePrinter.hpp.

Referenced by unixColourCodeToString().

6.55.3.3 `const std::string ConsolePrinter::CSI_RESET_CODE = "0" [static, private]`

Definition at line 188 of file ConsolePrinter.hpp.

Referenced by `getUnixColourCode()`.

6.55.3.4 `const std::string ConsolePrinter::CSI_SEPARATOR = ";" [static, private]`

Definition at line 189 of file ConsolePrinter.hpp.

Referenced by `getUnixColourCode()`.

6.55.3.5 `const std::string ConsolePrinter::CSI_START_TAG = "\033[" [static, private]`

Definition at line 186 of file ConsolePrinter.hpp.

Referenced by `getUnixColourCode()`.

6.55.3.6 `const std::string ConsolePrinter::ERR_INVALID_COLOUR_CODE = "The provided colour code is invalid. Please provide a valid colour code instead (see documentation for more details)." [static, private]`

Definition at line 195 of file ConsolePrinter.hpp.

6.55.3.7 `const std::string ConsolePrinter::SEPARATOR = "" [static, private]`

Definition at line 182 of file ConsolePrinter.hpp.

Referenced by `printColouredMessageWithColouredTag()`, and `printMessageWithColouredTag()`.

6.55.3.8 `const std::string ConsolePrinter::TERM_ENV_VARIABLE = "TERM" [static, private]`

Definition at line 193 of file ConsolePrinter.hpp.

Referenced by `terminalSupportsColour()`.

6.55.3.9 `const std::string ConsolePrinter::WARNING_TAG = "[ WARNING ]" [static, private]`

Definition at line 184 of file ConsolePrinter.hpp.

Referenced by `printWarningMessage()`.

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

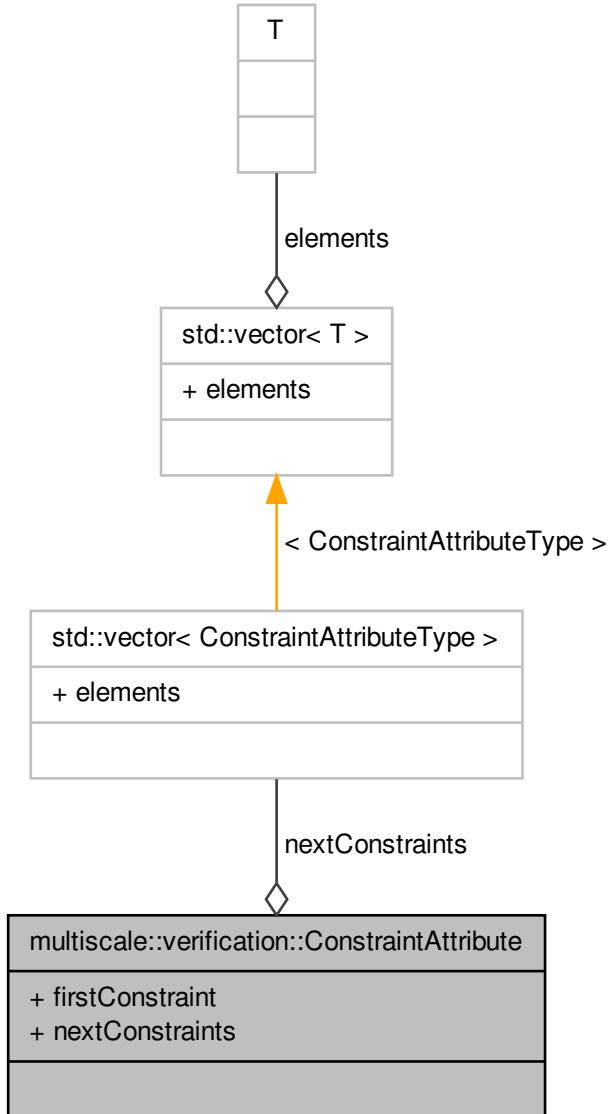
- ConsolePrinter.hpp
- ConsolePrinter.cpp

## 6.56 multiscale::verification::ConstraintAttribute Class Reference

Class for representing a constraint attribute.

```
#include <ConstraintAttribute.hpp>
```

Collaboration diagram for multiscale::verification::ConstraintAttribute:



### Public Attributes

- `ConstraintAttributeType firstConstraint`

- std::vector < ConstraintAttributeType > nextConstraints

### 6.56.1 Detailed Description

Class for representing a constraint attribute.

Definition at line 36 of file ConstraintAttribute.hpp.

### 6.56.2 Member Data Documentation

#### 6.56.2.1 ConstraintAttributeType multiscale::verification::ConstraintAttribute- ::firstConstraint

The first constraint

Definition at line 40 of file ConstraintAttribute.hpp.

Referenced by multiscale::verification::ConstraintVisitor::operator()().

#### 6.56.2.2 std::vector<ConstraintAttributeType> multiscale::verification::- ConstraintAttribute::nextConstraints

The next constraints

Definition at line 41 of file ConstraintAttribute.hpp.

Referenced by multiscale::verification::ConstraintVisitor::evaluateNextConstraints().

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

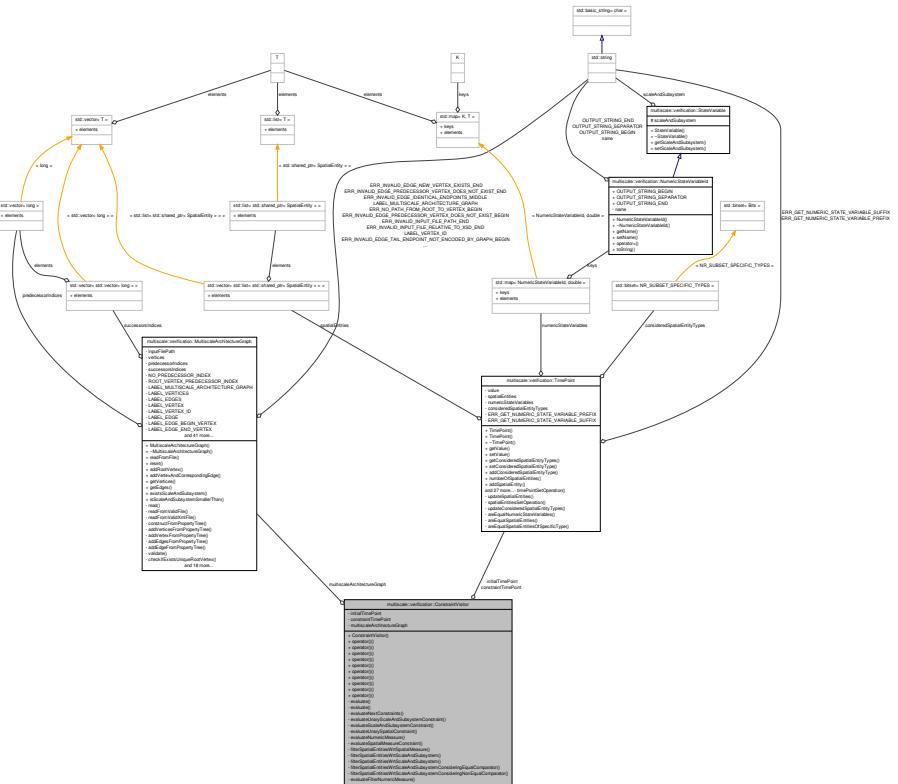
- ConstraintAttribute.hpp

## 6.57 multiscale::verification::ConstraintVisitor Class Reference

Class used to evaluate constraints.

```
#include <ConstraintVisitor.hpp>
```

## Collaboration diagram for multiscale::verification::ConstraintVisitor:



## Public Member Functions

- **ConstraintVisitor** (*TimePoint &initialTimePoint, TimePoint &constraintTimePoint, const MultiscaleArchitectureGraph &multiscaleArchitectureGraph*)
  - **TimePoint operator()** (*const Nil &constraint*) const  
*Overloading the "()" operator for the **Nil** alternative.*
  - **TimePoint operator()** (*const ConstraintAttribute &constraint*) const  
*Overloading the "()" operator for the **ConstraintAttribute** alternative.*
  - **TimePoint operator()** (*const OrConstraintAttribute &constraint*) const  
*Overloading the "()" operator for the **OrConstraintAttribute** alternative.*
  - **TimePoint operator()** (*const AndConstraintAttribute &constraint*) const  
*Overloading the "()" operator for the **AndConstraintAttribute** alternative.*
  - **TimePoint operator()** (*const ImplicationConstraintAttribute &constraint*) const  
*Overloading the "()" operator for the **ImplicationConstraintAttribute** alternative.*
  - **TimePoint operator()** (*const EquivalenceConstraintAttribute &constraint*) const  
*Overloading the "()" operator for the **EquivalenceConstraintAttribute** alternative.*

- `TimePoint operator() (const PrimaryConstraintAttribute &primaryConstraint) const`  
*Overloading the "(" operator for the `PrimaryConstraintAttribute` alternative.*
- `TimePoint operator() (const NotConstraintAttribute &primaryConstraint) const`  
*Overloading the "(" operator for the `NotConstraintAttribute` alternative.*
- `TimePoint operator() (const UnaryScaleAndSubsystemConstraintAttribute &primaryConstraint) const`  
*Overloading the "(" operator for the `UnaryScaleAndSubsystemConstraintAttribute` alternative.*
- `TimePoint operator() (const UnarySpatialConstraintAttribute &primaryConstraint) const`  
*Overloading the "(" operator for the `UnarySpatialConstraintAttribute` alternative.*

## Private Member Functions

- `TimePoint evaluate (const ConstraintAttributeType &constraint, TimePoint &timePoint) const`  
*Evaluate the constraint considering the given timepoint.*
- `TimePoint evaluate (const PrimaryConstraintAttributeType &primaryConstraint, - TimePoint &timePoint) const`  
*Evaluate the primary constraint considering the given timepoints.*
- `TimePoint evaluateNextConstraints (const ConstraintAttribute &constraint, const TimePoint &timePoint) const`  
*Evaluate the next constraints.*
- `TimePoint evaluateUnaryScaleAndSubsystemConstraint (const ComparatorType &comparator, const ScaleAndSubsystemAttribute &scaleAndSubsystem, TimePoint &timePoint) const`  
*Evaluate the unary scale and subsystem constraint.*
- `void evaluateScaleAndSubsystemConstraint (TimePoint &timePoint, const - ComparatorType &comparator, const ScaleAndSubsystemAttribute &scaleAndSubsystem) const`  
*Filter the timepoint's spatial entities considering the scale and subsystem of each spatial entity.*
- `TimePoint evaluateUnarySpatialConstraint (const SpatialMeasureType &spatialMeasure, const ComparatorType &comparator, const FilterNumericMeasureAttributeType &filterNumericMeasure, TimePoint &timePoint) const`  
*Evaluate the unary spatial constraint.*
- `double evaluateNumericMeasure (const NumericMeasureType &numericMeasure, TimePoint &timePoint) const`  
*Evaluate the numeric measure considering the given timepoint.*
- `void evaluateSpatialMeasureConstraint (TimePoint &timePoint, const SpatialMeasureType &spatialMeasure, const ComparatorType &comparator, const - FilterNumericMeasureAttributeType &filterNumericMeasure) const`  
*Filter the timepoint's spatial entities considering the given spatial measure constraint.*

- void `filterSpatialEntitiesWrtSpatialMeasure` (`TimePoint &timePoint, const SubsetSpecificType &spatialEntityType, const SpatialMeasureType &spatialMeasure, const ComparatorType &comparator, const FilterNumericMeasureAttributeType &filterNumericMeasure) const`  
*Remove from the timepoint the spatial entities which fail to meet the spatial measure constraint.*
- void `filterSpatialEntitiesWrtScaleAndSubsystem` (`TimePoint &timePoint, const -SubsetSpecificType &subsetSpecificType, const ComparatorType &comparator, const ScaleAndSubsystemAttribute &scaleAndSubsystem) const`  
*Remove from timepoint the spatial entities which fail to meet the scale and subsystem constraint.*
- void `filterSpatialEntitiesWrtScaleAndSubsystem` (`TimePoint &timePoint, const SubsetSpecificType &spatialEntityType, const ComparatorType &comparator, const std::string &scaleAndSubsystem) const`  
*Remove from timepoint the spatial entities which fail to meet the scale and subsystem constraint.*
- void `filterSpatialEntitiesWrtScaleAndSubsystemConsideringEqualComparator` (`-TimePoint &timePoint, const SubsetSpecificType &spatialEntityType, const std::string &rhsScaleAndSubsystem) const`  
*Remove from timepoint the spatial entities which fail to meet the scale and subsystem constraint.*
- void `filterSpatialEntitiesWrtScaleAndSubsystemConsideringNonEqualComparator` (`TimePoint &timePoint, const SubsetSpecificType &spatialEntityType, const -ComparatorType &comparator, const std::string &rhsScaleAndSubsystem) const`  
*Remove from timepoint the spatial entities which fail to meet the scale and subsystem constraint.*
- double `evaluateFilterNumericMeasure` (`const FilterNumericMeasureAttributeType &filterNumericMeasure, TimePoint &timePoint, const SpatialEntity &spatialEntity) const`  
*Evaluate the filter numeric measure considering the provided timepoint and spatial entity.*

## Private Attributes

- `TimePoint & initialTimePoint`
- `TimePoint & constraintTimePoint`
- `const MultiscaleArchitectureGraph & multiscaleArchitectureGraph`

### 6.57.1 Detailed Description

Class used to evaluate constraints.

Definition at line 19 of file ConstraintVisitor.hpp.

### 6.57.2 Constructor & Destructor Documentation

6.57.2.1 **multiscale::verification::ConstraintVisitor::ConstraintVisitor ( TimePoint & *initialTimePoint*, TimePoint & *constraintTimePoint*, const MultiscaleArchitectureGraph & *multiscaleArchitectureGraph* ) [inline]**

Definition at line 32 of file ConstraintVisitor.hpp.

Referenced by evaluate(), and evaluateNextConstraints().

### 6.57.3 Member Function Documentation

6.57.3.1 **TimePoint multiscale::verification::ConstraintVisitor::evaluate ( const ConstraintAttributeType & *constraint*, TimePoint & *timePoint* ) const [inline, private]**

Evaluate the constraint considering the given timepoint.

#### Parameters

<i>constraint</i>	The given constraint
<i>timePoint</i>	The given timepoint

Definition at line 182 of file ConstraintVisitor.hpp.

References ConstraintVisitor(), initialTimePoint, and multiscaleArchitectureGraph.

Referenced by filterSpatialEntitiesWrtSpatialMeasure(), and operator()().

6.57.3.2 **TimePoint multiscale::verification::ConstraintVisitor::evaluate ( const PrimaryConstraintAttributeType & *primaryConstraint*, TimePoint & *timePoint* ) const [inline, private]**

Evaluate the primary constraint considering the given timepoints.

#### Parameters

<i>primary-Constraint</i>	The given primary constraint
<i>timePoint</i>	The given timepoint

Definition at line 200 of file ConstraintVisitor.hpp.

References ConstraintVisitor(), initialTimePoint, and multiscaleArchitectureGraph.

---

**6.57.3.3 double multiscale::verification::ConstraintVisitor::evaluateFilterNumericMeasure ( const FilterNumericMeasureAttributeType & *filterNumericMeasure*, TimePoint & *timePoint*, const SpatialEntity & *spatialEntity* ) const [inline, private]**

Evaluate the filter numeric measure considering the provided timepoint and spatial entity.

**Parameters**

<i>filter-Numeric-Measure</i>	The filter numeric measure
<i>timePoint</i>	The considered timepoint
<i>spatialEntity</i>	The considered spatial entity

Definition at line 531 of file ConstraintVisitor.hpp.

References multiscaleArchitectureGraph.

Referenced by filterSpatialEntitiesWrtSpatialMeasure().

**6.57.3.4 TimePoint multiscale::verification::ConstraintVisitor::evaluateNextConstraints ( const ConstraintAttribute & *constraint*, const TimePoint & *timePoint* ) const [inline, private]**

Evaluate the next constraints.

Evaluate the next constraints considering the given constraint and timepoints

**Parameters**

<i>constraint</i>	The given constraint
<i>timePoint</i>	The resulting timepoint after applying the first constraint to the initial timepoint

Definition at line 222 of file ConstraintVisitor.hpp.

References ConstraintVisitor(), initialTimePoint, multiscaleArchitectureGraph, and multiscale::verification::ConstraintAttribute::nextConstraints.

Referenced by operator()().

**6.57.3.5 double multiscale::verification::ConstraintVisitor::evaluateNumericMeasure ( const NumericMeasureType & *numericMeasure*, TimePoint & *timePoint* ) const [inline, private]**

Evaluate the numeric measure considering the given timepoint.

**Parameters**

<i>numericMeasure</i>	The numeric measure
<i>timePoint</i>	The given timepoint

Definition at line 320 of file ConstraintVisitor.hpp.

References multiscaleArchitectureGraph.

```
6.57.3.6 void multiscale::verification::ConstraintVisitor::evaluateScaleAnd-
    SubsystemConstraint ( TimePoint & timePoint, const ComparatorType &
    comparator, const ScaleAndSubsystemAttribute & scaleAndSubsystem ) const
    [inline, private]
```

Filter the timepoint's spatial entities considering the scale and subsystem of each spatial entity.

**Parameters**

<i>timePoint</i>	The timepoint storing the collection of spatial entities which will be filtered
<i>comparator</i>	The type of the comparator
<i>scaleAndSubsystem</i>	The scale and subsystem

Definition at line 271 of file ConstraintVisitor.hpp.

References multiscale::verification::subsetspecific::computeSubsetSpecificType(), filterSpatialEntitiesWrtScaleAndSubsystem(), multiscale::verification::TimePoint::getConsideredSpatialEntityTypes(), and multiscale::verification::NR\_SUBSET\_SPECIFIC\_TYPES.

Referenced by evaluateUnaryScaleAndSubsystemConstraint().

```
6.57.3.7 void multiscale::verification::ConstraintVisitor::evaluateSpatial-
    MeasureConstraint ( TimePoint & timePoint, const SpatialMeasureType
    & spatialMeasure, const ComparatorType & comparator, const
    FilterNumericMeasureAttributeType & filterNumericMeasure ) const
    [inline, private]
```

Filter the timepoint's spatial entities considering the given spatial measure constraint.

All considered spatial entities which fail to meet the constraints will be removed from the given timepoint.

**Parameters**

<i>timePoint</i>	The timepoint storing the collection of spatial entities which will be filtered
<i>spatialMeasure</i>	The type of the spatial measure

<i>comparator</i>	The type of the comparator
<i>filter-Numeric-Measure</i>	The filter numeric measure

Definition at line 344 of file ConstraintVisitor.hpp.

References multiscale::verification::subsetspecific::computeSubsetSpecificType(), filterSpatialEntitiesWrtSpatialMeasure(), multiscale::verification::TimePoint::getConsideredSpatialEntityTypes(), and multiscale::verification::NR\_SUBSET\_SPECIFIC\_TYPES.

Referenced by evaluateUnarySpatialConstraint().

**6.57.3.8 TimePoint multiscale::verification::ConstraintVisitor::evaluateUnaryScaleAndSubsystemConstraint ( const ComparatorType & *comparator*, const ScaleAndSubsystemAttribute & *scaleAndSubsystem*, TimePoint & *timePoint* ) const [inline, private]**

Evaluate the unary scale and subsystem constraint.

Evaluate the unary scale and subsystem constraint considering the given spatial measure, comparator, scale and subsystem and timepoint

#### Parameters

<i>comparator</i>	The comparator type
<i>scaleAnd-Subsystem</i>	The scale and subsystem
<i>timePoint</i>	The considered timepoint

Definition at line 248 of file ConstraintVisitor.hpp.

References evaluateScaleAndSubsystemConstraint().

Referenced by operator()().

**6.57.3.9 TimePoint multiscale::verification::ConstraintVisitor::evaluateUnarySpatialConstraint ( const SpatialMeasureType & *spatialMeasure*, const ComparatorType & *comparator*, const FilterNumericMeasureAttributeType & *filterNumericMeasure*, TimePoint & *timePoint* ) const [inline, private]**

Evaluate the unary spatial constraint.

Evaluate the unary spatial constraint considering the given spatial measure, comparator, numeric measure and timepoint

#### Parameters

<i>spatial-Measure</i>	The spatial measure type
------------------------	--------------------------

<i>comparator</i>	The comparator type
<i>filter-Numeric-Measure</i>	The filter numeric measure
<i>timePoint</i>	The considered timepoint

Definition at line 299 of file ConstraintVisitor.hpp.

References evaluateSpatialMeasureConstraint().

Referenced by operator()().

```
6.57.3.10 void multiscale::verification::ConstraintVisitor::filterSpatialEntitiesWrt-
ScaleAndSubsystem ( TimePoint & timePoint, const SubsetSpecificType
& subsetSpecificType, const ComparatorType & comparator, const
ScaleAndSubsystemAttribute & scaleAndSubsystem ) const [inline,
private]
```

Remove from timepoint the spatial entities which fail to meet the scale and subsystem constraint.

#### Parameters

<i>timePoint</i>	The timepoint which will be filtered
<i>subset-SpecificType</i>	The considered subset specific type
<i>comparator</i>	The type of the comparator
<i>scaleAnd-Subsystem</i>	The scaleAndSubsystem type

Definition at line 408 of file ConstraintVisitor.hpp.

References multiscaleArchitectureGraph, multiscale::verification::ScaleAndSubsystem-Attribute::scaleAndSubsystem, and multiscale::verification::ScaleAndSubsystem-Evaluator::validateScaleAndSubsystem().

Referenced by evaluateScaleAndSubsystemConstraint().

```
6.57.3.11 void multiscale::verification::ConstraintVisitor::filterSpatialEntitiesWrt-
ScaleAndSubsystem ( TimePoint & timePoint, const SubsetSpecificType &
spatialEntityType, const ComparatorType & comparator, const std::string &
scaleAndSubsystem ) const [inline, private]
```

Remove from timepoint the spatial entities which fail to meet the scale and subsystem constraint.

The assumption for this method is that the provided scale and subsystem exists in the multiscale architecture graph.

**Parameters**

<i>timePoint</i>	The timepoint which will be filtered
<i>spatialEntityType</i>	The considered spatial entity type
<i>comparator</i>	The type of the comparator
<i>scaleAndSubsystem</i>	The scale and subsystem

Definition at line 441 of file ConstraintVisitor.hpp.

References filterSpatialEntitiesWrtScaleAndSubsystemConsideringEqualComparator(), and filterSpatialEntitiesWrtScaleAndSubsystemConsideringNonEqualComparator().

**6.57.3.12 void multiscale::verification::ConstraintVisitor::filterSpatialEntities-WrtScaleAndSubsystemConsideringEqualComparator ( TimePoint & *timePoint*, const SubsetSpecificType & *spatialEntityType*, const std::string & *rhsScaleAndSubsystem* ) const [inline, private]**

Remove from timepoint the spatial entities which fail to meet the scale and subsystem constraint.

The assumption for this method is that the considered comparator is "=".

In this case the multiscale architecture graph is NOT used.

**Parameters**

<i>timePoint</i>	The timepoint which will be filtered
<i>spatialEntityType</i>	The considered spatial entity type
<i>rhsScale-And-Subsystem</i>	The scale and subsystem on the right hand side of the comparator

Definition at line 470 of file ConstraintVisitor.hpp.

References multiscale::verification::TimePoint::getSpatialEntitiesBeginIterator(), multiscale::verification::TimePoint::getSpatialEntitiesEndIterator(), and multiscale::verification::TimePoint::removeSpatialEntity().

Referenced by filterSpatialEntitiesWrtScaleAndSubsystem().

**6.57.3.13 void multiscale::verification::ConstraintVisitor::filterSpatialEntities-WrtScaleAndSubsystemConsideringNonEqualComparator ( TimePoint & *timePoint*, const SubsetSpecificType & *spatialEntityType*, const ComparatorType & *comparator*, const std::string & *rhsScaleAndSubsystem* ) const [inline, private]**

Remove from timepoint the spatial entities which fail to meet the scale and subsystem constraint.

The assumption for this method is that the considered comparator is different from "=".

In this case the multiscale architecture graph is used.

#### Parameters

<i>timePoint</i>	The timepoint which will be filtered
<i>spatialEntity-Type</i>	The considered spatial entity type
<i>comparator</i>	The type of the comparator
<i>rhsScale-And-Subsystem</i>	The scale and subsystem on the right hand side of the comparator

Definition at line 499 of file ConstraintVisitor.hpp.

References multiscale::verification::ComparatorEvaluator::evaluate(), multiscale::verification::TimePoint::getSpatialEntitiesBeginIterator(), multiscale::verification::TimePoint::getSpatialEntitiesEndIterator(), multiscaleArchitectureGraph, and multiscale::verification::TimePoint::removeSpatialEntity().

Referenced by filterSpatialEntitiesWrtScaleAndSubsystem().

```
6.57.3.14 void multiscale::verification::ConstraintVisitor::filterSpatialEntities-
WrtSpatialMeasure ( TimePoint & timePoint, const SubsetSpecificType
& spatialEntityType, const SpatialMeasureType & spatialMeasure, const
ComparatorType & comparator, const FilterNumericMeasureAttributeType
& filterNumericMeasure ) const [inline, private]
```

Remove from the timepoint the spatial entities which fail to meet the spatial measure constraint.

#### Parameters

<i>timePoint</i>	The timepoint which will be filtered
<i>spatialEntity-Type</i>	The considered spatial entity type
<i>spatial-Measure</i>	The type of the spatial measure
<i>comparator</i>	The type of the comparator
<i>filter-Numeric-Measure</i>	The filter numeric measure

Definition at line 375 of file ConstraintVisitor.hpp.

References multiscale::verification::ComparatorEvaluator::evaluate(), evaluate(), evaluateFilterNumericMeasure(), multiscale::verification::TimePoint::getSpatialEntitiesBeginIterator(), multiscale::verification::TimePoint::getSpatialEntitiesEndIterator(), and multiscale::verification::TimePoint::removeSpatialEntity().

Referenced by evaluateSpatialMeasureConstraint().

6.57.3.15 **TimePoint multiscale::verification::ConstraintVisitor::operator()** ( **const Nil & constraint** ) **const** [inline]

Overloading the "(" operator for the [Nil](#) alternative.

**Parameters**

<b>constraint</b>	The constraint
-------------------	----------------

Definition at line 41 of file ConstraintVisitor.hpp.

References initialTimePoint.

6.57.3.16 **TimePoint multiscale::verification::ConstraintVisitor::operator()** ( **const ConstraintAttribute & constraint** ) **const** [inline]

Overloading the "(" operator for the [ConstraintAttribute](#) alternative.

**Parameters**

<b>constraint</b>	The constraint
-------------------	----------------

Definition at line 49 of file ConstraintVisitor.hpp.

References evaluate(), evaluateNextConstraints(), multiscale::verification::ConstraintAttribute::firstConstraint, and initialTimePoint.

6.57.3.17 **TimePoint multiscale::verification::ConstraintVisitor::operator()** ( **const OrConstraintAttribute & constraint** ) **const** [inline]

Overloading the "(" operator for the [OrConstraintAttribute](#) alternative.

**Parameters**

<b>constraint</b>	The constraint
-------------------	----------------

Definition at line 63 of file ConstraintVisitor.hpp.

References multiscale::verification::OrConstraintAttribute::constraint, constraintTimePoint, evaluate(), initialTimePoint, and multiscale::verification::TimePoint::timePointUnion().

6.57.3.18 **TimePoint multiscale::verification::ConstraintVisitor::operator()** ( **const AndConstraintAttribute & constraint** ) **const** [inline]

Overloading the "(" operator for the [AndConstraintAttribute](#) alternative.

**Parameters**

<i>constraint</i>	The constraint
-------------------	----------------

Definition at line 75 of file ConstraintVisitor.hpp.

References multiscale::verification::AndConstraintAttribute::constraint, constraintTimePoint, evaluate(), initialTimePoint, and multiscale::verification::TimePoint::timePointIntersection().

**6.57.3.19 TimePoint multiscale::verification::ConstraintVisitor::operator() ( const ImplicationConstraintAttribute & *constraint* ) const [inline]**

Overloading the "(" operator for the [ImplicationConstraintAttribute](#) alternative.

**Parameters**

<i>constraint</i>	The constraint
-------------------	----------------

Definition at line 87 of file ConstraintVisitor.hpp.

References multiscale::verification::ImplicationConstraintAttribute::constraint, constraintTimePoint, evaluate(), initialTimePoint, multiscale::verification::TimePoint::timePointDifference(), and multiscale::verification::TimePoint::timePointUnion().

**6.57.3.20 TimePoint multiscale::verification::ConstraintVisitor::operator() ( const EquivalenceConstraintAttribute & *constraint* ) const [inline]**

Overloading the "(" operator for the [EquivalenceConstraintAttribute](#) alternative.

**Parameters**

<i>constraint</i>	The constraint
-------------------	----------------

Definition at line 102 of file ConstraintVisitor.hpp.

References multiscale::verification::EquivalenceConstraintAttribute::constraint, constraintTimePoint, evaluate(), initialTimePoint, multiscale::verification::TimePoint::timePointDifference(), multiscale::verification::TimePoint::timePointIntersection(), and multiscale::verification::TimePoint::timePointUnion().

**6.57.3.21 TimePoint multiscale::verification::ConstraintVisitor::operator() ( const PrimaryConstraintAttribute & *primaryConstraint* ) const [inline]**

Overloading the "(" operator for the [PrimaryConstraintAttribute](#) alternative.

**Parameters**

<i>primary- Constraint</i>	The primary constraint
--------------------------------	------------------------

Definition at line 125 of file ConstraintVisitor.hpp.

References constraintTimePoint, evaluate(), and multiscale::verification::PrimaryConstraintAttribute::primaryConstraint.

**6.57.3.22 TimePoint multiscale::verification::ConstraintVisitor::operator() ( const NotConstraintAttribute & primaryConstraint ) const [inline]**

Overloading the "(") operator for the [NotConstraintAttribute](#) alternative.

**Parameters**

<i>primary- Constraint</i>	The primary constraint
--------------------------------	------------------------

Definition at line 133 of file ConstraintVisitor.hpp.

References multiscale::verification::NotConstraintAttribute::constraint, constraintTimePoint, evaluate(), initialTimePoint, and multiscale::verification::TimePoint::timePointDifference().

**6.57.3.23 TimePoint multiscale::verification::ConstraintVisitor::operator() ( const UnaryScaleAndSubsystemConstraintAttribute & primaryConstraint ) const [inline]**

Overloading the "(") operator for the [UnaryScaleAndSubsystemConstraintAttribute](#) alternative.

**Parameters**

<i>primary- Constraint</i>	The primary constraint
--------------------------------	------------------------

Definition at line 146 of file ConstraintVisitor.hpp.

References multiscale::verification::UnaryScaleAndSubsystemConstraintAttribute::comparator, multiscale::verification::ComparatorAttribute::comparatorType, constraintTimePoint, evaluateUnaryScaleAndSubsystemConstraint(), and multiscale::verification::UnaryScaleAndSubsystemConstraintAttribute::scaleAndSubsystem.

**6.57.3.24 TimePoint multiscale::verification::ConstraintVisitor::operator() ( const UnarySpatialConstraintAttribute & primaryConstraint ) const [inline]**

Overloading the "(") operator for the [UnarySpatialConstraintAttribute](#) alternative.

**Parameters**

<i>primary- Constraint</i>	The primary constraint
--------------------------------	------------------------

Definition at line 162 of file ConstraintVisitor.hpp.

References      multiscale::verification::UnarySpatialConstraintAttribute::comparator, multiscale::verification::ComparatorAttribute::comparatorType, constraintTimePoint, evaluateUnarySpatialConstraint(), multiscale::verification::UnarySpatialConstraintAttribute::filterNumericMeasure, multiscale::verification::UnarySpatialConstraintAttribute::spatialMeasure, and multiscale::verification::SpatialMeasureAttribute::spatialMeasureType.

#### **6.57.4 Member Data Documentation**

##### **6.57.4.1 TimePoint& multiscale::verification::ConstraintVisitor::constraintTimePoint [private]**

The currently obtained constraint timepoint

Definition at line 26 of file ConstraintVisitor.hpp.

Referenced by operator()().

##### **6.57.4.2 TimePoint& multiscale::verification::ConstraintVisitor::initialTimePoint [private]**

A copy of the initial timepoint

Definition at line 24 of file ConstraintVisitor.hpp.

Referenced by evaluate(), evaluateNextConstraints(), and operator()().

##### **6.57.4.3 const MultiscaleArchitectureGraph& multiscale::verification::ConstraintVisitor::multiscaleArchitectureGraph [private]**

The considered multiscale architecture graph

Definition at line 28 of file ConstraintVisitor.hpp.

Referenced by evaluate(), evaluateFilterNumericMeasure(), evaluateNextConstraints(), evaluateNumericMeasure(), filterSpatialEntitiesWrtScaleAndSubsystem(), and filterSpatialEntitiesWrtScaleAndSubsystemConsideringNonEqualComparator().

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

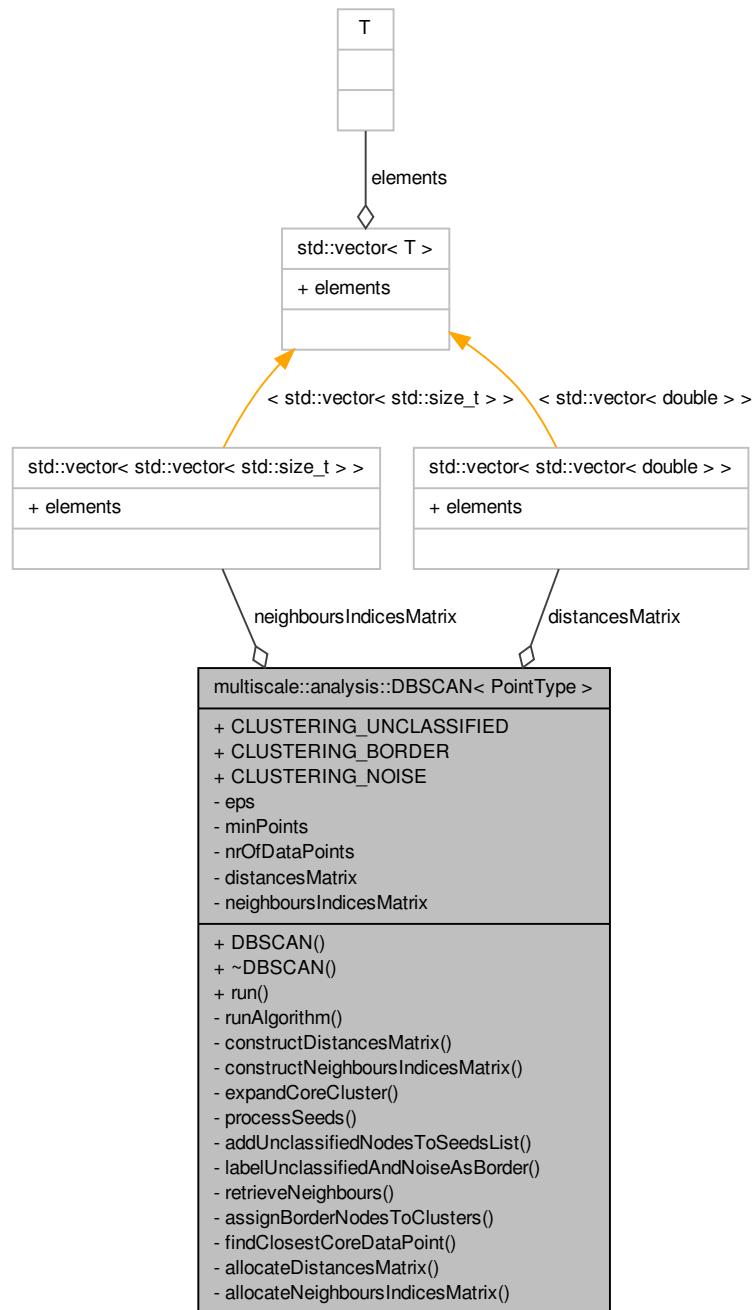
- ConstraintVisitor.hpp

## **6.58 multiscale::analysis::DBSCAN< PointType > Class Template Reference**

Class which implements an improved version of the [DBSCAN](#) algorithm.

```
#include <DBSCAN.hpp>
```

Collaboration diagram for multiscale::analysis::DBSCAN< PointType >:



## Public Member Functions

- `DBSCAN ()`
- `~DBSCAN ()`
- void `run` (const std::vector< PointType > &dataPoints, std::vector< int > &clusterIndexes, std::size\_t &nrOfClusters, double `eps`, int `minPoints`)

*Run the improved DBSCAN algorithm on the provided set of points.*

## Static Public Attributes

- static const int `CLUSTERING_UNCLASSIFIED` = -2
- static const int `CLUSTERING_BORDER` = -1
- static const int `CLUSTERING_NOISE` = 0

## Private Member Functions

- void `runAlgorithm` (const std::vector< PointType > &dataPoints, std::vector< int > &clusterIndexes, std::size\_t &nrOfClusters)
 

*Run the improved DBSCAN algorithm on the provided set of points.*
- void `constructDistancesMatrix` (const std::vector< PointType > &dataPoints)
 

*Construct the distances matrix.*
- void `constructNeighboursIndicesMatrix` (const std::vector< PointType > &dataPoints)
 

*Construct the neighbours indices matrix.*
- bool `expandCoreCluster` (std::vector< int > &clusterIndexes, std::size\_t coreDataPointIndex, std::size\_t clusterId)
 

*Expand the cluster around the given core data point.*
- void `processSeeds` (std::vector< int > &clusterIndexes, std::size\_t clusterId, std::vector< std::size\_t > &seeds, std::vector< bool > &consideredSeeds)
 

*Process the collection of provided seeds (see DBSCAN algorithm for details)*
- void `addUnclassifiedNodesToSeedsList` (const std::vector< std::size\_t > &neighbours, const std::vector< int > &clusterIndexes, std::vector< std::size\_t > &seeds, std::vector< bool > &consideredSeeds)
 

*Add all unclassified previously not considered neighbour nodes to the seeds list.*
- void `labelUnclassifiedAndNoiseAsBorder` (const std::vector< std::size\_t > &neighbours, std::vector< int > &clusterIndexes)
 

*Label all unclassified and noise neighbour nodes as border nodes.*
- std::vector< std::size\_t > `retrieveNeighbours` (std::size\_t dataPointIndex)
 

*Retrieve the list of neighbour indexes for the given data point.*
- void `assignBorderNodesToClusters` (std::vector< int > &clusterIndexes)
 

*Assign the border nodes to the clusters to which the closest core objects belong.*
- std::size\_t `findClosestCoreDataPoint` (std::size\_t borderDataPointIndex, const std::vector< int > &clusterIndexes)
 

*Find the closest core data point from the given set of neighbours to the given border data point.*

- void `allocateDistancesMatrix ()`  
*Allocate the distances matrix.*
- void `allocateNeighboursIndicesMatrix ()`  
*Allocate the neighbours indices matrix.*

### Private Attributes

- double `eps`
- std::size\_t `minPoints`
- std::size\_t `nrOfDataPoints`
- std::vector< std::vector < double > > `distancesMatrix`
- std::vector< std::vector < std::size\_t > > `neighboursIndicesMatrix`

### 6.58.1 Detailed Description

`template<typename PointType> class multiscale::analysis::DBSCAN< PointType >`

Class which implements an improved version of the [DBSCAN](#) algorithm.

Definition at line 24 of file DBSCAN.hpp.

### 6.58.2 Constructor & Destructor Documentation

**6.58.2.1 template<typename PointType > multiscale::analysis::DBSCAN< PointType >::DBSCAN( ) [inline]**

Definition at line 44 of file DBSCAN.hpp.

**6.58.2.2 template<typename PointType > multiscale::analysis::DBSCAN< PointType >::~DBSCAN( ) [inline]**

Definition at line 46 of file DBSCAN.hpp.

References `multiscale::analysis::DBSCAN< PointType >::distancesMatrix`, and `multiscale::analysis::DBSCAN< PointType >::neighboursIndicesMatrix`.

### 6.58.3 Member Function Documentation

**6.58.3.1 template<typename PointType > void multiscale::analysis::DBSCAN< PointType >::addUnclassifiedNodesToSeedsList ( const std::vector< std::size\_t > & *neighbours*, const std::vector< int > & *clusterIndexes*, std::vector< std::size\_t > & *seeds*, std::vector< bool > & *consideredSeeds* ) [inline, private]**

Add all unclassified previously not considered neighbour nodes to the seeds list.

A node is added to the seed list only if it was not considered already

#### Parameters

<i>neighbours</i>	Neighbour nodes
<i>cluster-Indexes</i>	Indexes to which cluster each data point belongs
<i>seeds</i>	List of seeds (see <a href="#">DBSCAN</a> algorithm)
<i>considered-Seeds</i>	List of previously considered seeds

Definition at line 203 of file DBSCAN.hpp.

References multiscale::analysis::DBSCAN< PointType >::CLUSTERING\_UNCLASSIFIED.

Referenced by multiscale::analysis::DBSCAN< PointType >::processSeeds().

**6.58.3.2 template<typename PointType> void multiscale::analysis::DBSCAN< PointType >::allocateDistancesMatrix( ) [inline, private]**

Allocate the distances matrix.

Definition at line 285 of file DBSCAN.hpp.

References multiscale::analysis::DBSCAN< PointType >::distancesMatrix, and multiscale::analysis::DBSCAN< PointType >::nrOfDataPoints.

Referenced by multiscale::analysis::DBSCAN< PointType >::constructDistancesMatrix().

**6.58.3.3 template<typename PointType> void multiscale::analysis::DBSCAN< PointType >::allocateNeighboursIndicesMatrix( ) [inline, private]**

Allocate the neighbours indices matrix.

Definition at line 296 of file DBSCAN.hpp.

References multiscale::analysis::DBSCAN< PointType >::neighboursIndicesMatrix, and multiscale::analysis::DBSCAN< PointType >::nrOfDataPoints.

Referenced by multiscale::analysis::DBSCAN< PointType >::constructNeighboursIndicesMatrix().

**6.58.3.4 template<typename PointType> void multiscale::analysis::DBSCAN< PointType >::assignBorderNodesToClusters( std::vector< int > & clusterIndexes ) [inline, private]**

Assign the border nodes to the clusters to which the closest core objects belong.

**Parameters**

<code>cluster-Indexes</code>	Indexes to which cluster each data point belongs
------------------------------	--

Definition at line 249 of file DBSCAN.hpp.

References multiscale::analysis::DBSCAN< PointType >::findClosestCoreDataPoint(), and multiscale::analysis::DBSCAN< PointType >::nrOfDataPoints.

Referenced by multiscale::analysis::DBSCAN< PointType >::runAlgorithm().

**6.58.3.5 template<typename PointType> void multiscale::analysis::DBSCAN< PointType >::constructDistancesMatrix ( const std::vector< PointType > & `dataPoints` ) [inline, private]**

Construct the distances matrix.

**Parameters**

<code>dataPoints</code>	The considered data points
-------------------------	----------------------------

Definition at line 111 of file DBSCAN.hpp.

References multiscale::analysis::DBSCAN< PointType >::allocateDistancesMatrix(), multiscale::analysis::DBSCAN< PointType >::distancesMatrix, and multiscale::analysis::DBSCAN< PointType >::nrOfDataPoints.

Referenced by multiscale::analysis::DBSCAN< PointType >::run().

**6.58.3.6 template<typename PointType> void multiscale::analysis::DBSCAN< PointType >::constructNeighboursIndicesMatrix ( const std::vector< PointType > & `dataPoints` ) [inline, private]**

Construct the neighbours indices matrix.

**Parameters**

<code>dataPoints</code>	The considered data points
-------------------------	----------------------------

Definition at line 126 of file DBSCAN.hpp.

References multiscale::analysis::DBSCAN< PointType >::allocateNeighboursIndicesMatrix(), multiscale::analysis::DBSCAN< PointType >::distancesMatrix, multiscale::analysis::DBSCAN< PointType >::eps, multiscale::analysis::DBSCAN< PointType >::neighboursIndicesMatrix, and multiscale::analysis::DBSCAN< PointType >::nrOfDataPoints.

Referenced by multiscale::analysis::DBSCAN< PointType >::run().

---

6.58.3.7 template<typename PointType > bool multiscale::analysis::DBSCAN< PointType >::expandCoreCluster( std::vector< int > & clusterIndexes, std::size\_t coreDataPointIndex, std::size\_t clusterId ) [inline, private]

Expand the cluster around the given core data point.

**Parameters**

<i>cluster- Indexes</i>	Indexes to which cluster each data point belongs
<i>coreData- PointIndex</i>	Core data point index
<i>clusterId</i>	Id of the cluster to which the core data point belongs

Definition at line 145 of file DBSCAN.hpp.

References multiscale::analysis::DBSCAN< PointType >::CLUSTERING\_NOISE, multiscale::analysis::DBSCAN< PointType >::minPoints, multiscale::analysis::DBSCAN< PointType >::nrOfDataPoints, multiscale::analysis::DBSCAN< PointType >::processSeeds(), and multiscale::analysis::DBSCAN< PointType >::retrieveNeighbours().

Referenced by multiscale::analysis::DBSCAN< PointType >::runAlgorithm().

6.58.3.8 template<typename PointType > std::size\_t multiscale::analysis::DBSCAN< PointType >::findClosestCoreDataPoint( std::size\_t borderDataPointIndex, const std::vector< int > & clusterIndexes ) [inline, private]

Find the closest core data point from the given set of neighbours to the given border data point.

**Parameters**

<i>borderData- PointIndex</i>	Index of the border data point
<i>cluster- Indexes</i>	Indexes to which cluster each data point belongs

Definition at line 264 of file DBSCAN.hpp.

References multiscale::analysis::DBSCAN< PointType >::distancesMatrix, and multiscale::analysis::DBSCAN< PointType >::retrieveNeighbours().

Referenced by multiscale::analysis::DBSCAN< PointType >::assignBorderNodesToClusters().

```
6.58.3.9 template<typename PointType> void multiscale::analysis::DBSCAN<
    PointType>::labelUnclassifiedAndNoiseAsBorder ( const std::vector<
        std::size_t > & neighbours, std::vector< int > & clusterIndexes ) [inline,
    private]
```

Label all unclassified and noise neighbour nodes as border nodes.

#### Parameters

<i>neighbours</i>	Neighbour nodes
<i>cluster- Indexes</i>	Indexes to which cluster each data point belongs

Definition at line 223 of file DBSCAN.hpp.

References multiscale::analysis::DBSCAN< PointType >::CLUSTERING\_BORDER, multiscale::analysis::DBSCAN< PointType >::CLUSTERING\_NOISE, and multiscale::analysis::DBSCAN< PointType >::CLUSTERING\_UNCLASSIFIED.

Referenced by multiscale::analysis::DBSCAN< PointType >::processSeeds().

```
6.58.3.10 template<typename PointType> void multiscale::analysis::DBSCAN<
    PointType>::processSeeds ( std::vector< int > & clusterIndexes, std::size_t
    clusterId, std::vector< std::size_t > & seeds, std::vector< bool > &
    consideredSeeds ) [inline, private]
```

Process the collection of provided seeds (see [DBSCAN](#) algorithm for details)

#### Parameters

<i>cluster- Indexes</i>	Indexes to which cluster each data point belongs
<i>clusterId</i>	The index of the data point to which the seeds correspond
<i>seeds</i>	The collection of seeds
<i>considered- Seeds</i>	The collection of previously considered seeds

Definition at line 173 of file DBSCAN.hpp.

References multiscale::analysis::DBSCAN< PointType >::addUnclassifiedNodesToSeedsList(), multiscale::analysis::DBSCAN< PointType >::labelUnclassifiedAndNoiseAsBorder(), multiscale::analysis::DBSCAN< PointType >::minPoints, and multiscale::analysis::DBSCAN< PointType >::retrieveNeighbours().

Referenced by multiscale::analysis::DBSCAN< PointType >::expandCoreCluster().

---

6.58.3.11 template<typename PointType > std::vector<std::size\_t>  
**multiscale::analysis::DBSCAN< PointType >::retrieveNeighbours (**  
**std::size\_t dataPointIndex )** [inline, private]

Retrieve the list of neighbour indexes for the given data point.

Retrieve the list of neighbour indexes which are at a distance < eps far from the given data point

**Parameters**

<i>dataPoint- Index</i>	Index of the data point for which the neighbours will be retrieved
-----------------------------	--

Definition at line 239 of file DBSCAN.hpp.

References multiscale::analysis::DBSCAN< PointType >::neighboursIndicesMatrix.

Referenced by multiscale::analysis::DBSCAN< PointType >::expandCoreCluster(), multiscale::analysis::DBSCAN< PointType >::findClosestCoreDataPoint(), and multiscale::analysis::DBSCAN< PointType >::processSeeds().

6.58.3.12 template<typename PointType > void **multiscale::analysis::DBSCAN<**  
**PointType >::run ( const std::vector< PointType > & *dataPoints*, std::vector<**  
**int > & *clusterIndexes*, std::size\_t & *nrOfClusters*, double *eps*, int *minPoints* )**  
[inline]

Run the improved [DBSCAN](#) algorithm on the provided set of points.

The implementation of the improved [DBSCAN](#) algorithm is based on the paper: T. N. Tran, K. Drab, and M. Daszykowski, ‘Revised [DBSCAN](#) algorithm to cluster data with dense adjacent clusters’, Chemometrics and Intelligent Laboratory Systems, vol. 120, pp. 92–96, Jan. 2013.

Clusters start from index 1, because cluster 0 contains only noise data/points.

**Parameters**

<i>dataPoints</i>	Collection of data points
<i>cluster- Indexes</i>	Indexes to which cluster each data point belongs
<i>nrOfClusters</i>	Total number of clusters
<i>eps</i>	Maximum distance between two neighbours
<i>minPoints</i>	Minimum number of points in one cluster

Definition at line 65 of file DBSCAN.hpp.

References multiscale::analysis::DBSCAN< PointType >::constructDistancesMatrix(), multiscale::analysis::DBSCAN< PointType >::constructNeighboursIndicesMatrix(), multiscale::analysis::DBSCAN< PointType >::eps, multiscale::analysis::DBSCAN< PointType >::minPoints, multiscale::analysis::DBSCAN< PointType >::nrOfDataPoints, and multiscale::analysis::DBSCAN< PointType >::runAlgorithm().

**6.58.3.13 template<typename PointType > void multiscale::analysis::DBSCAN< PointType >::runAlgorithm ( const std::vector< PointType > & *dataPoints*, std::vector< int > & *clusterIndexes*, std::size\_t & *nrOfClusters* ) [inline, private]**

Run the improved [DBSCAN](#) algorithm on the provided set of points.

**Parameters**

<i>dataPoints</i>	Collection of data points
<i>clusterIndexes</i>	Indexes to which cluster each data point belongs
<i>nrOfClusters</i>	Total number of clusters

Definition at line 87 of file DBSCAN.hpp.

References multiscale::analysis::DBSCAN< PointType >::assignBorderNodesToClusters(), multiscale::analysis::DBSCAN< PointType >::CLUSTERING\_UNCLASSIFIED, multiscale::analysis::DBSCAN< PointType >::expandCoreCluster(), and multiscale::analysis::DBSCAN< PointType >::nrOfDataPoints.

Referenced by multiscale::analysis::DBSCAN< PointType >::run().

## 6.58.4 Member Data Documentation

**6.58.4.1 template<typename PointType > const int multiscale::analysis::DBSCAN< PointType >::CLUSTERING\_BORDER = -1 [static]**

Definition at line 310 of file DBSCAN.hpp.

Referenced by multiscale::analysis::DBSCAN< PointType >::labelUnclassifiedAndNoiseAsBorder().

**6.58.4.2 template<typename PointType > const int multiscale::analysis::DBSCAN< PointType >::CLUSTERING\_NOISE = 0 [static]**

Definition at line 311 of file DBSCAN.hpp.

Referenced by multiscale::analysis::DBSCAN< PointType >::expandCoreCluster(), and multiscale::analysis::DBSCAN< PointType >::labelUnclassifiedAndNoiseAsBorder().

**6.58.4.3 template<typename PointType > const int multiscale::analysis::DBSCAN< PointType >::CLUSTERING\_UNCLASSIFIED = -2 [static]**

Definition at line 309 of file DBSCAN.hpp.

Referenced by multiscale::analysis::DBSCAN< PointType >::addUnclassifiedNodesToSeedsList(), multiscale::analysis::DBSCAN< PointType >::labelUnclassifiedAndNoiseAsBorder(), and multiscale::analysis::DBSCAN< PointType >::runAlgorithm().

---

```
6.58.4.4 template<typename PointType > std::vector<std::vector<double> >
    multiscale::analysis::DBSCAN< PointType >::distancesMatrix
    [private]
```

The matrix containing the distances between each possible pair of datapoints

Definition at line 36 of file DBSCAN.hpp.

Referenced by multiscale::analysis::DBSCAN< PointType >::allocateDistancesMatrix(), multiscale::analysis::DBSCAN< PointType >::constructDistancesMatrix(), multiscale::analysis::DBSCAN< PointType >::constructNeighboursIndicesMatrix(), multiscale::analysis::DBSCAN< PointType >::findClosestCoreDataPoint(), and multiscale::analysis::DBSCAN< PointType >::~DBSCAN().

```
6.58.4.5 template<typename PointType > double multiscale::analysis::DBSCAN<
    PointType >::eps [private]
```

**DBSCAN** algorithm parameter for specifying the maximum radius of the neighbourhood

Definition at line 28 of file DBSCAN.hpp.

Referenced by multiscale::analysis::DBSCAN< PointType >::constructNeighboursIndicesMatrix(), and multiscale::analysis::DBSCAN< PointType >::run().

```
6.58.4.6 template<typename PointType > std::size_t multiscale::analysis::DBSCAN<
    PointType >::minPoints [private]
```

**DBSCAN** algorithm parameter for specifying the minimum number of points in an eps-neighbourhood of that point

Definition at line 30 of file DBSCAN.hpp.

Referenced by multiscale::analysis::DBSCAN< PointType >::expandCoreCluster(), multiscale::analysis::DBSCAN< PointType >::processSeeds(), and multiscale::analysis::DBSCAN< PointType >::run().

```
6.58.4.7 template<typename PointType > std::vector<std::vector<std::size_t> >
    multiscale::analysis::DBSCAN< PointType >::neighboursIndicesMatrix
    [private]
```

The matrix containing for each data point a vector of indices corresponding to the neighbouring data points

Definition at line 39 of file DBSCAN.hpp.

Referenced by multiscale::analysis::DBSCAN< PointType >::allocateNeighboursIndicesMatrix(), multiscale::analysis::DBSCAN< PointType >::constructNeighboursIndicesMatrix(), multiscale::analysis::DBSCAN< PointType >::retrieveNeighbours(), and multiscale::analysis::DBSCAN< PointType >::~DBSCAN().

6.58.4.8 template<typename PointType> std::size\_t multiscale::analysis::DBSCAN< PointType >::nrOfDataPoints [private]

Number of data points in the data set

Definition at line 33 of file DBSCAN.hpp.

Referenced by multiscale::analysis::DBSCAN< PointType >::allocateDistancesMatrix(), multiscale::analysis::DBSCAN< PointType >::allocateNeighboursIndicesMatrix(), multiscale::analysis::DBSCAN< PointType >::assignBorderNodesToClusters(), multiscale::analysis::DBSCAN< PointType >::constructDistancesMatrix(), multiscale::analysis::DBSCAN< PointType >::constructNeighboursIndicesMatrix(), multiscale::analysis::DBSCAN< PointType >::expandCoreCluster(), multiscale::analysis::DBSCAN< PointType >::run(), and multiscale::analysis::DBSCAN< PointType >::runAlgorithm().

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

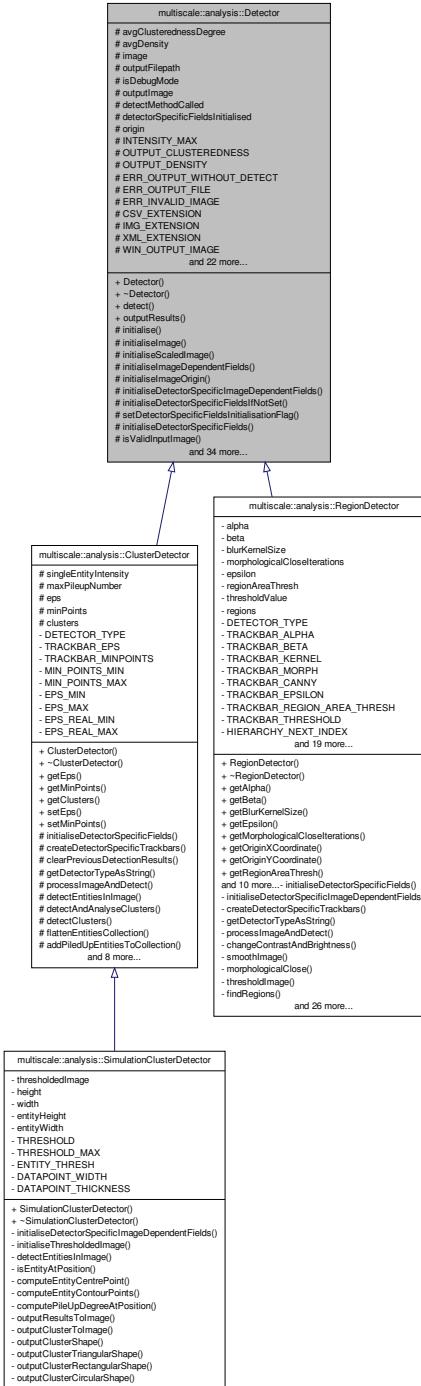
- DBSCAN.hpp

## 6.59 multiscale::analysis::Detector Class Reference

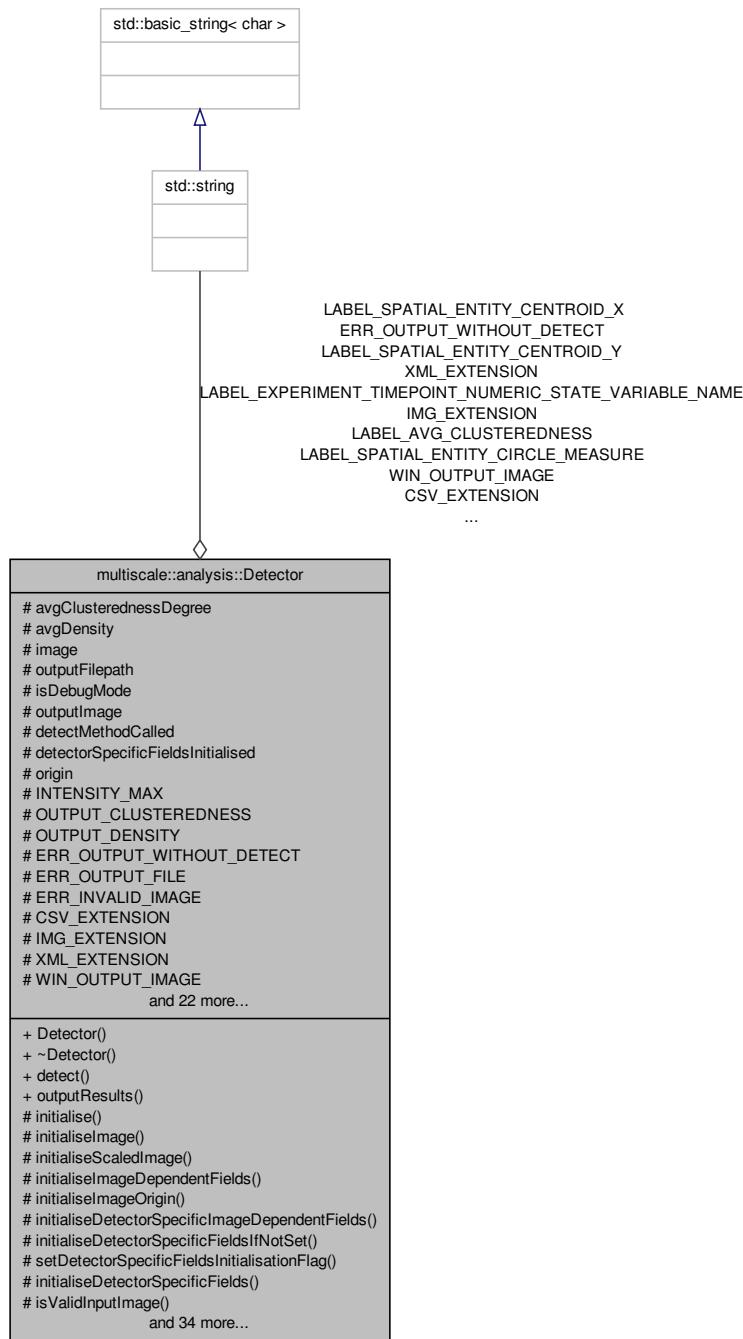
Abstract class for detecting entities of interest in images.

```
#include <Detector.hpp>
```

## Inheritance diagram for multiscale::analysis::Detector:



Collaboration diagram for multiscale::analysis::Detector:



## Public Member Functions

- `Detector (bool isDebugMode=false)`
- `virtual ~Detector ()`
- `void detect (const cv::Mat &inputImage)`  
*Run the detection procedure on the given image.*
- `void outputResults (const std::string &outputFilepath)`  
*Output the results to the given file.*

## Protected Member Functions

- `void initialise (const cv::Mat &inputImage)`  
*Initialisation function for the class.*
- `void initialiseImage (const cv::Mat &inputImage)`  
*Initialise the image field of the class.*
- `void initialiseScaledImage (const cv::Mat &inputImage)`  
*Initialise the image field of the class as a scaled copy of the given image.*
- `void initialiseImageDependentFields ()`  
*Initialisation of the image dependent values.*
- `void initialiseImageOrigin ()`
- `virtual void initialiseDetectorSpecificImageDependentFields ()=0`  
*Initialisation of the detector specific image dependent values.*
- `void initialiseDetectorSpecificFieldsIfNotSet ()`  
*Initialisation of the detector specific values in case they were not set.*
- `void setDetectorSpecificFieldsInitialisationFlag (bool flag=true)`  
*Set the detector specific fields initialisation flag to true.*
- `virtual void initialiseDetectorSpecificFields ()=0`  
*Initialisation of the detector specific values.*
- `bool isValidInputImage (const cv::Mat &inputImage)`  
*Check if the image is valid.*
- `virtual std::string getDetectorTypeAsString ()=0`  
*Get the type of the employed detector as a string.*
- `void detect ()`  
*Run the detection procedure.*
- `void detectInDebugMode ()`  
*Run the detection procedure when in debug mode.*
- `void detectInReleaseMode ()`  
*Run the detection procedure when in release mode (i.e. non-debug mode)*
- `double computeDistanceFromOrigin (const std::vector< cv::Point > &polygon)`  
*Compute the distance of the given polygon from the origin.*
- `double computeDistanceFromOrigin (const std::vector< cv::Point2f > &polygon)`  
*Compute the distance of the given polygon from the origin.*

- double `computePolygonAngle` (const std::vector< cv::Point > &polygon)  
*Compute the angle determined by the polygon tangents from a given point.*
- double `computePolygonAngle` (const std::vector< cv::Point2f > &polygon)  
*Compute the angle determined by the polygon tangents from a given point.*
- double `computePolygonAngle` (const std::vector< cv::Point2f > &polygon->ConvexHull, const cv::Point2f &tangentsPoint)  
*Compute the angle of the tangents from the provided point to the polygon.*
- void `displayResultsInWindow` ()  
*Display the results in a window.*
- void `outputResultsToFile` ()  
*Output the results to file(s)*
- virtual void `outputResultsToImage` ()=0  
*Output the results to an image.*
- void `storeOutputImageOnDisk` ()  
*Store the image with the output results on disk.*
- void `outputResultsToCsvFile` ()  
*Output the results to a csv file.*
- void `outputResultsToCsvFile` (std::ofstream &fout)  
*Output the results to a file using the provided output file stream.*
- void `outputSpatialEntitiesToCsvFile` (std::ofstream &fout)  
*Output the pseudo 3D spatial entities to a csv file.*
- void `outputAveragedMeasuresToCsvFile` (std::ofstream &fout)  
*Output the averaged measures to a csv file.*
- void `outputResultsToXMLFile` ()  
*Output the results to an xml file.*
- void `outputResultsToXMLFile` (const std::string &filepath)  
*Output the clusters and averaged measures to an xml file.*
- void `addSpatialEntitiesToPropertyTree` (pt::ptree &propertyTree)  
*Add the pseudo 3D spatial entities to the property tree.*
- void `addAverageMeasuresToPropertyTree` (pt::ptree &propertyTree)  
*Add the average clusteredness and average density to the property tree.*
- void `addNumericStateVariableToPropertyTree` (pt::ptree &propertyTree, const std::string &name, double value)  
*Add a numeric state variable to the property tree.*
- pt::ptree `constructSpatialEntityPropertyTree` (SpatialEntityPseudo3D &spatialEntity)  
*Construct the property tree corresponding to the given spatial entity.*
- void `addSpatialEntityPropertiesToTree` (SpatialEntityPseudo3D &spatialEntity, pt::ptree &propertyTree)  
*Add the properties of the spatial entity to the property tree.*
- void `addSpatialEntityTypeToPropertyTree` (SpatialEntityPseudo3D &spatialEntity, pt::ptree &propertyTree)  
*Add the type of the spatial entity to the property tree.*

- virtual std::vector < std::shared\_ptr < SpatialEntityPseudo3D > > `getCollectionOfSpatialEntityPseudo3D ()=0`  
*Get the collection of pseudo 3D entities detected in the image.*
- virtual void `processImageAndDetect ()=0`  
*Process the input image and detect objects/entities of interest.*
- virtual void `clearPreviousDetectionResults ()=0`  
*Clear the results from the previous detection.*
- void `createTrackbars ()`  
*Create the trackbars which allow the user to change the values of the parameters.*
- void `createTrackbarsWindow ()`  
*Create the window in which the trackbars are placed.*
- virtual void `createDetectorSpecificTrackbars ()=0`  
*Create the trackbars specific to the used detector.*
- void `processPressedKeyRequest (char &pressedKey)`  
*Process the request of the user by pressing the key.*
- void `displayImage (const cv::Mat &image, const std::string &windowName)`  
*Display an image in a particular window.*
- void `printOutputErrorMessage ()`  
*Print error message, because the detect method was not called before calling the output method.*
- void `offsetPolygons (std::vector< std::vector< cv::Point > > &polygons, const cv::Point &offset)`  
*Offset the given collection of polygons by the given point.*

### Protected Attributes

- double `avgClusterednessDegree`
- double `avgDensity`
- cv::Mat `image`
- std::string `outputFilepath`
- bool `isDebugMode`
- cv::Mat `outputImage`
- bool `detectMethodCalled`
- bool `detectorSpecificFieldsInitialised`
- cv::Point2f `origin`

### Static Protected Attributes

- static const int `INTENSITY_MAX` = 255
- static const std::string `OUTPUT_CLUSTEREDNESS` = "Average clusteredness degree: "
- static const std::string `OUTPUT_DENSITY` = "Average density: "
- static const std::string `ERR_OUTPUT_WITHOUT_DETECT` = "Unable to output results if the `detect` method was not called previously."

- static const std::string `ERR_OUTPUT_FILE` = "Unable to create output file."
- static const std::string `ERR_INVALID_IMAGE` = "The input `image` is invalid."
- static const std::string `CSV_EXTENSION` = ".out"
- static const std::string `IMG_EXTENSION` = ".png"
- static const std::string `XML_EXTENSION` = ".xml"
- static const std::string `WIN_OUTPUT_IMAGE` = "Output `image`"
- static const int `KEY_ESC` = 27
- static const int `KEY_SAVE` = 115
- static const std::string `LABEL_ATTRIBUTE` = "<xmllattr>"
- static const std::string `LABEL_COMMENT` = "<xmlcomment>"
- static const std::string `LABEL_COMMENT_CONTENTS` = "Warning! This xml file was automatically generated by a C++ program using the Boost PropertyTree library."
- static const std::string `LABEL_EXPERIMENT_TIMEPOINT_NUMERIC_STATE_VARIABLE` = "experiment.timepoint.numericStateVariable"
- static const std::string `LABEL_EXPERIMENT_TIMEPOINT_SPATIAL_ENTITY` = "experiment.timepoint.spatialEntity"
- static const std::string `LABEL_EXPERIMENT_TIMEPOINT_NUMERIC_STATE_VARIABLE_NAME` = "name"
- static const std::string `LABEL_EXPERIMENT_TIMEPOINT_NUMERIC_STATE_VARIABLE_VALUE` = "value"
- static const std::string `LABEL_SPATIAL_ENTITY_SPATIAL_TYPE` = "spatial-Type"
- static const std::string `LABEL_SPATIAL_ENTITY_CLUSTEREDNESS` = "clusteredness"
- static const std::string `LABEL_SPATIAL_ENTITY_DENSITY` = "density"
- static const std::string `LABEL_SPATIAL_ENTITY_AREA` = "area"
- static const std::string `LABEL_SPATIAL_ENTITY_PERIMETER` = "perimeter"
- static const std::string `LABEL_SPATIAL_ENTITY_DISTANCE_FROM_ORIGIN` = "distanceFromOrigin"
- static const std::string `LABEL_SPATIAL_ENTITY_ANGLE` = "angle"
- static const std::string `LABEL_SPATIAL_ENTITY_TRIANGLE_MEASURE` = "triangleMeasure"
- static const std::string `LABEL_SPATIAL_ENTITY_RECTANGLE_MEASURE` = "rectangleMeasure"
- static const std::string `LABEL_SPATIAL_ENTITY_CIRCLE_MEASURE` = "circleMeasure"
- static const std::string `LABEL_SPATIAL_ENTITY_CENTROID_X` = "centroidX"
- static const std::string `LABEL_SPATIAL_ENTITY_CENTROID_Y` = "centroidY"
- static const std::string `LABEL_AVG_CLUSTEREDNESS` = "avgClusteredness"
- static const std::string `LABEL_AVG_DENSITY` = "avgDensity"

### 6.59.1 Detailed Description

Abstract class for detecting entities of interest in images.

Definition at line 22 of file Detector.hpp.

### 6.59.2 Constructor & Destructor Documentation

#### 6.59.2.1 Detector::Detector ( bool *isDebugEnabled* = false )

Definition at line 11 of file Detector.cpp.

References avgClusterednessDegree, avgDensity, detectMethodCalled, and detector-SpecificFieldsInitialised.

#### 6.59.2.2 Detector::~Detector ( ) [virtual]

Definition at line 20 of file Detector.cpp.

References image, and outputImage.

### 6.59.3 Member Function Documentation

#### 6.59.3.1 void Detector::addAverageMeasuresToPropertyTree ( pt::ptree & *propertyTree* ) [protected]

Add the average clusteredness and average density to the property tree.

##### Parameters

<i>propertyTree</i>	The property tree
---------------------	-------------------

Definition at line 276 of file Detector.cpp.

References addNumericStateVariableToPropertyTree(), avgClusterednessDegree, avg-Density, getDetectorTypeAsString(), LABEL\_AVG\_CLUSTEREDNESS, and LABEL\_A-VG\_DENSITY.

Referenced by outputResultsToXMLFile().

#### 6.59.3.2 void Detector::addNumericStateVariableToPropertyTree ( pt::ptree & *propertyTree*, const std::string & *name*, double *value* ) [protected]

Add a numeric state variable to the property tree.

##### Parameters

<i>propertyTree</i>	The property tree
<i>name</i>	The name of the numeric state variable
<i>value</i>	The value of the numeric state variable

Definition at line 285 of file Detector.cpp.

References LABEL\_EXPERIMENT\_TIMEPOINT\_NUMERIC\_STATE\_VARIABLE, LA-BEL\_EXPERIMENT\_TIMEPOINT\_NUMERIC\_STATE\_VARIABLE\_NAME, and LABE-L\_EXPERIMENT\_TIMEPOINT\_NUMERIC\_STATE\_VARIABLE\_VALUE.

Referenced by addAverageMeasuresToPropertyTree().

**6.59.3.3 void Detector::addSpatialEntitiesToPropertyTree ( pt::ptree & *propertyTree* ) [protected]**

Add the pseudo 3D spatial entities to the property tree.

**Parameters**

<i>propertyTree</i>	The property tree
---------------------	-------------------

Definition at line 266 of file Detector.cpp.

References constructSpatialEntityPropertyTree(), getCollectionOfSpatialEntityPseudo3D(), and LABEL\_EXPERIMENT\_TIMEPOINT\_SPATIAL\_ENTITY.

Referenced by outputResultsToXMLFile().

**6.59.3.4 void Detector::addSpatialEntityPropertiesToTree ( SpatialEntityPseudo3D & *spatialEntity*, pt::ptree & *propertyTree* ) [protected]**

Add the properties of the spatial entity to the property tree.

**Parameters**

<i>spatialEntity</i>	Spatial entity
<i>propertyTree</i>	Property tree

Definition at line 304 of file Detector.cpp.

References multiscale::analysis::SpatialEntityPseudo3D::getAngle(), multiscale::analysis::SpatialEntityPseudo3D::getArea(), multiscale::analysis::SpatialEntityPseudo3D::getCentre(), multiscale::analysis::SpatialEntityPseudo3D::getCircularMeasure(), multiscale::analysis::SpatialEntityPseudo3D::getClusterednessDegree(), multiscale::analysis::SpatialEntityPseudo3D::getDensity(), multiscale::analysis::SpatialEntityPseudo3D::getDistanceFromOrigin(), multiscale::analysis::SpatialEntityPseudo3D::getPerimeter(), multiscale::analysis::SpatialEntityPseudo3D::getRectangularMeasure(), multiscale::analysis::SpatialEntityPseudo3D::getTriangularMeasure(), LABEL\_SPATIAL\_ENTITY\_ANGLE, LABEL\_SPATIAL\_ENTITY\_AREA, LABEL\_SPATIAL\_ENTITY\_CENTROID\_X, LABEL\_SPATIAL\_ENTITY\_CENTROID\_Y, LABEL\_SPATIAL\_ENTITY\_CIRCLE\_MEASURE, LABEL\_SPATIAL\_ENTITY\_CLUSTREDNESS, LABEL\_SPATIAL\_ENTITY\_DENSITY, LABEL\_SPATIAL\_ENTITY\_DISTANCE\_FROM\_ORIGIN, LABEL\_SPATIAL\_ENTITY\_PERIMETER, LABEL\_SPATIAL\_ENTITY\_RECTANGLE\_MEASURE, and LABEL\_SPATIAL\_ENTITY\_TRIANGLE\_MEASURE.

Referenced by constructSpatialEntityPropertyTree().

---

**6.59.3.5 void Detector::addSpatialEntityTypeToPropertyTree (**  
**SpatialEntityPseudo3D & *spatialEntity*, pt::ptree & *propertyTree* )**  
**[protected]**

Add the type of the spatial entity to the property tree.

**Parameters**

<i>spatialEntity</i>	Spatial entity
<i>propertyTree</i>	Property tree

Definition at line 321 of file Detector.cpp.

References `LABEL_ATTRIBUTE`, `LABEL_SPATIAL_ENTITY_SPATIAL_TYPE`, and `multiscale::analysis::SpatialEntityPseudo3D::typeAsString()`.

Referenced by `constructSpatialEntityPropertyTree()`.

**6.59.3.6 virtual void multiscale::analysis::Detector::clearPreviousDetectionResults ( ) [protected, pure virtual]**

Clear the results from the previous detection.

Implemented in [multiscale::analysis::RegionDetector](#), and [multiscale::analysis::ClusterDetector](#).

Referenced by `detectInDebugMode()`, and `detectInReleaseMode()`.

**6.59.3.7 double Detector::computeDistanceFromOrigin ( const std::vector< cv::Point > & *polygon* ) [protected]**

Compute the distance of the given polygon from the origin.

**Parameters**

<i>polygon</i>	The considered polygon
----------------	------------------------

Definition at line 120 of file Detector.cpp.

Referenced by `multiscale::analysis::RegionDetector::createRegionFromPolygon()`, and `multiscale::analysis::ClusterDetector::updateClusterOriginDependentValues()`.

**6.59.3.8 double Detector::computeDistanceFromOrigin ( const std::vector< cv::Point2f > & *polygon* ) [protected]**

Compute the distance of the given polygon from the origin.

**Parameters**

<i>polygon</i>	The considered polygon
----------------	------------------------

Definition at line 126 of file Detector.cpp.

References multiscale::Geometry2D::distanceBtwPoints(), multiscale::Geometry2D::isPointInsidePolygon(), multiscale::Geometry2D::minimumDistancePointIndex(), and origin.

**6.59.3.9 double Detector::computePolygonAngle ( const std::vector< cv::Point > & *polygon* ) [protected]**

Compute the angle determined by the polygon tangents from a given point.

The considered point through which the tangents pass is the image/matrix centre point (origin).

**Parameters**

<i>polygon</i>	Polygon for which the angle is computed
----------------	---

Definition at line 142 of file Detector.cpp.

Referenced by computePolygonAngle(), multiscale::analysis::RegionDetector::createRegionFromPolygon(), and multiscale::analysis::ClusterDetector::updateClusterOriginDependentValues().

**6.59.3.10 double Detector::computePolygonAngle ( const std::vector< cv::Point2f > & *polygon* ) [protected]**

Compute the angle determined by the polygon tangents from a given point.

The considered point through which the tangents pass is the image/matrix centre point (origin).

**Parameters**

<i>polygon</i>	Polygon for which the angle is computed
----------------	---

Definition at line 150 of file Detector.cpp.

References multiscale::Geometry2D::computeConvexHull(), computePolygonAngle(), and origin.

**6.59.3.11 double Detector::computePolygonAngle ( const std::vector< cv::Point2f > & *polygonConvexHull*, const cv::Point2f & *tangentsPoint* ) [protected]**

Compute the angle of the tangents from the provided point to the polygon.

**Parameters**

<i>polygon-ConvexHull</i>	Convex hull of polygon
<i>tangents-Point</i>	Point through which the tangents pass

Definition at line 160 of file Detector.cpp.

References multiscale::Geometry2D::angleBtwPoints(), multiscale::Geometry2D::isPointInsidePolygon(), and multiscale::Geometry2D::tangentsFromPointToPolygon().

**6.59.3.12 `pt::ptree Detector::constructSpatialEntityPropertyTree ( SpatialEntityPseudo3D & spatialEntity )` [protected]**

Construct the property tree corresponding to the given spatial entity.

**Parameters**

<code>spatialEntity</code>	The spatial entity to be converted
----------------------------	------------------------------------

Definition at line 295 of file Detector.cpp.

References addSpatialEntityPropertiesToTree(), and addSpatialEntityTypeToPropertyTree().

Referenced by addSpatialEntitiesToPropertyTree().

**6.59.3.13 `virtual void multiscale::analysis::Detector::createDetectorSpecificTrackbars ( )` [protected, pure virtual]**

Create the trackbars specific to the used detector.

Implemented in [multiscale::analysis::RegionDetector](#), and [multiscale::analysis::ClusterDetector](#).

Referenced by createTrackbars().

**6.59.3.14 `void Detector::createTrackbars ( )` [protected]**

Create the trackbars which allow the user to change the values of the parameters.

Definition at line 330 of file Detector.cpp.

References createDetectorSpecificTrackbars(), and createTrackbarsWindow().

Referenced by detectInDebugMode().

**6.59.3.15 `void Detector::createTrackbarsWindow ( )` [protected]**

Create the window in which the trackbars are placed.

Definition at line 335 of file Detector.cpp.

References WIN\_OUTPUT\_IMAGE.

Referenced by createTrackbars().

**6.59.3.16 void Detector::detect ( const cv::Mat & *inputImage* )**

Run the detection procedure on the given image.

**Parameters**

<i>inputImage</i>	The input image
-------------------	-----------------

Definition at line 25 of file Detector.cpp.

References detect(), ERR\_INVALID\_IMAGE, initialise(), and isValidInputImage().

**6.59.3.17 void Detector::detect ( ) [protected]**

Run the detection procedure.

Definition at line 90 of file Detector.cpp.

References detectInDebugMode(), detectInReleaseMode(), detectMethodCalled, and isDebugEnabled.

Referenced by detect().

**6.59.3.18 void Detector::detectInDebugMode ( ) [protected]**

Run the detection procedure when in debug mode.

Definition at line 100 of file Detector.cpp.

References clearPreviousDetectionResults(), createTrackbars(), displayResultsInWindow(), KEY\_ESC, processImageAndDetect(), and processPressedKeyRequest().

Referenced by detect().

**6.59.3.19 void Detector::detectInReleaseMode ( ) [protected]**

Run the detection procedure when in release mode (i.e. non-debug mode)

Definition at line 115 of file Detector.cpp.

References clearPreviousDetectionResults(), and processImageAndDetect().

Referenced by detect().

**6.59.3.20 void Detector::displayImage ( const cv::Mat & *image*, const std::string & *windowName* ) [protected]**

Display an image in a particular window.

**Parameters**

<i>image</i>	The image
<i>window- Name</i>	The name of the window

Definition at line 346 of file Detector.cpp.

Referenced by `displayResultsInWindow()`.

#### 6.59.3.21 void **Detector::displayResultsInWindow( )** [protected]

Display the results in a window.

Definition at line 189 of file Detector.cpp.

References `displayImage()`, `outputImage`, `outputResultsToImage()`, and `WIN_OUTPUT_IMAGE`.

Referenced by `detectInDebugMode()`.

#### 6.59.3.22 virtual std::vector<std::shared\_ptr<**SpatialEntityPseudo3D**> > **multiscale::analysis::Detector::getCollectionOfSpatialEntityPseudo3D( )** [protected, pure virtual]

Get the collection of pseudo 3D entities detected in the image.

Implemented in [multiscale::analysis::RegionDetector](#), and [multiscale::analysis::ClusterDetector](#).

Referenced by `addSpatialEntitiesToPropertyTree()`, and `outputSpatialEntitiesToCsvFile()`.

#### 6.59.3.23 virtual std::string **multiscale::analysis::Detector::getDetectorTypeAsString( )** [protected, pure virtual]

Get the type of the employed detector as a string.

Implemented in [multiscale::analysis::RegionDetector](#), and [multiscale::analysis::ClusterDetector](#).

Referenced by `addAverageMeasuresToPropertyTree()`.

#### 6.59.3.24 void **Detector::initialise( const cv::Mat & *inputImage* )** [protected]

Initialisation function for the class.

**Parameters**

<i>inputImage</i>	The provided input image
-------------------	--------------------------

Definition at line 47 of file Detector.cpp.

References `initialiseDetectorSpecificFieldsIfNotSet()`, `initialiseImage()`, and `initialiseImageDependentFields()`.

Referenced by `detect()`.

**6.59.3.25 virtual void multiscale::analysis::Detector::initialiseDetectorSpecificFields( ) [protected, pure virtual]**

Initialisation of the detector specific values.

Implemented in [multiscale::analysis::RegionDetector](#), and [multiscale::analysis::ClusterDetector](#).

Referenced by `initialiseDetectorSpecificFieldsIfNotSet()`.

**6.59.3.26 void Detector::initialiseDetectorSpecificFieldsIfNotSet( ) [protected]**

Initialisation of the detector specific values in case they were not set.

Definition at line 69 of file `Detector.cpp`.

References `detectorSpecificFieldsInitialised`, and `initialiseDetectorSpecificFields()`.

Referenced by `initialise()`.

**6.59.3.27 virtual void multiscale::analysis::Detector::initialiseDetectorSpecificImageDependentFields( ) [protected, pure virtual]**

Initialisation of the detector specific image dependent values.

Implemented in [multiscale::analysis::RegionDetector](#), and [multiscale::analysis::SimulationClusterDetector](#).

Referenced by `initialiseImageDependentFields()`.

**6.59.3.28 void Detector::initialiseImage( const cv::Mat & *inputImage* ) [protected]**

Initialise the image field of the class.

#### Parameters

<i>inputImage</i>	The provided input image
-------------------	--------------------------

Definition at line 53 of file `Detector.cpp`.

References `image`.

Referenced by `initialise()`.

**6.59.3.29 void Detector::initialiseImageDependentFields( ) [protected]**

Initialisation of the image dependent values.

Definition at line 57 of file Detector.cpp.

References initialiseDetectorSpecificImageDependentFields(), and initialiseImageOrigin().

Referenced by initialise().

**6.59.3.30 void Detector::initialiseImageOrigin( ) [protected]**

Definition at line 62 of file Detector.cpp.

References image, and origin.

Referenced by initialiseImageDependentFields().

**6.59.3.31 void multiscale::analysis::Detector::initialiseScaledImage( const cv::Mat & inputImage ) [protected]**

Initialise the image field of the class as a scaled copy of the given image.

**Parameters**

<i>inputImage</i>	The provided input image
-------------------	--------------------------

**6.59.3.32 bool Detector::isValidInputImage( const cv::Mat & inputImage ) [protected]**

Check if the image is valid.

Check if the number of dimensions = 2, if the number of rows and number of columns is greater than one and if the image is of type CV\_8UC1

**Parameters**

<i>inputImage</i>	The input image
-------------------	-----------------

Definition at line 81 of file Detector.cpp.

Referenced by detect().

**6.59.3.33 void Detector::offsetPolygons( std::vector< std::vector< cv::Point >> & polygons, const cv::Point & offset ) [protected]**

Offset the given collection of polygons by the given point.

The polygon points are offset by adding to their coordinates the coordinates of the given point.

**Parameters**

<i>polygons</i>	The collection of given polygons
<i>offset</i>	The offset point

Definition at line 355 of file Detector.cpp.

Referenced by multiscale::analysis::RegionDetector::findPolygonsInImage().

**6.59.3.34 void Detector::outputAveragedMeasuresToCsvFile ( std::ofstream & *fout* ) [protected]**

Output the averaged measures to a csv file.

**Parameters**

<i>fout</i>	Output file stream
-------------	--------------------

Definition at line 243 of file Detector.cpp.

References avgClusterednessDegree, avgDensity, OUTPUT\_CLUSTEREDNESS, and OUTPUT\_DENSITY.

Referenced by outputResultsToCsvFile().

**6.59.3.35 void Detector::outputResults ( const std::string & *outputFilepath* )**

Output the results to the given file.

**Parameters**

<i>output-Filepath</i>	Path to the output file
------------------------	-------------------------

Definition at line 37 of file Detector.cpp.

References detectMethodCalled, outputFilepath, outputResultsToFile(), and printOutputErrorMessage().

**6.59.3.36 void Detector::outputResultsToCsvFile ( ) [protected]**

Output the results to a csv file.

Definition at line 211 of file Detector.cpp.

References CSV\_EXTENSION, ERR\_OUTPUT\_FILE, and outputFilepath.

**6.59.3.37 void Detector::outputResultsToCsvFile ( std::ofstream & *fout* ) [protected]**

Output the results to a file using the provided output file stream.

**Parameters**

<i>fout</i>	Output file stream
-------------	--------------------

Definition at line 223 of file Detector.cpp.

References multiscale::analysis::SpatialEntityPseudo3D::fieldNamesToString(), output-AveragedMeasuresToCsvFile(), and outputSpatialEntitiesToCsvFile().

**6.59.3.38 void Detector::outputResultsToFile( ) [protected]**

Output the results to file(s)

Definition at line 194 of file Detector.cpp.

References outputResultsToImage(), and outputResultsToXMLFile().

Referenced by outputResults().

**6.59.3.39 virtual void multiscale::analysis::Detector::outputResultsToImage( ) [protected, pure virtual]**

Output the results to an image.

Implemented in [multiscale::analysis::RegionDetector](#), and [multiscale::analysis::SimulationClusterDetector](#).

Referenced by displayResultsInWindow(), and outputResultsToFile().

**6.59.3.40 void Detector::outputResultsToXMLFile( ) [protected]**

Output the results to an xml file.

Definition at line 248 of file Detector.cpp.

References outputFilepath, and XML\_EXTENSION.

Referenced by outputResultsToFile().

**6.59.3.41 void Detector::outputResultsToXMLFile( const std::string & *filepath* ) [protected]**

Output the clusters and averaged measures to an xml file.

**Parameters**

<i>filepath</i>	Output file path
-----------------	------------------

Definition at line 252 of file Detector.cpp.

References addAverageMeasuresToPropertyTree(), addSpatialEntitiesToPropertyTree(), LABEL\_COMMENT, and LABEL\_COMMENT\_CONTENTS.

6.59.3.42 void **Detector::outputSpatialEntitiesToCsvFile ( std::ofstream & *fout* )**  
[protected]

Output the pseudo 3D spatial entities to a csv file.

Parameters

<i>fout</i>	Output file stream
-------------	--------------------

Definition at line 235 of file Detector.cpp.

References getCollectionOfSpatialEntityPseudo3D().

Referenced by outputResultsToCsvFile().

6.59.3.43 void **Detector::printOutputErrorMessage ( )** [protected]

Print error message, because the detect method was not called before calling the output method.

Definition at line 351 of file Detector.cpp.

References ERR\_OUTPUT\_WITHOUT\_DETECT.

Referenced by outputResults().

6.59.3.44 virtual void **multiscale::analysis::Detector::processImageAndDetect ( )**  
[protected, pure virtual]

Process the input image and detect objects/entities of interest.

Implemented in [multiscale::analysis::RegionDetector](#), and [multiscale::analysis::ClusterDetector](#).

Referenced by detectInDebugMode(), and detectInReleaseMode().

6.59.3.45 void **Detector::processPressedKeyRequest ( char & *pressedKey* )**  
[protected]

Process the request of the user by pressing the key.

Parameters

<i>pressedKey</i>	Key pressed by the user, if a key was pressed, or "-1", otherwise
-------------------	---

Definition at line 340 of file Detector.cpp.

Referenced by detectInDebugMode().

**6.59.3.46 void Detector::setDetectorSpecificFieldsInitialisationFlag ( bool *flag* = true ) [protected]**

Set the detector specific fields initialisation flag to true.

Definition at line 77 of file Detector.cpp.

References detectorSpecificFieldsInitialised.

Referenced by multiscale::analysis::RegionDetector::setAlpha(), multiscale::analysis::RegionDetector::setBeta(), multiscale::analysis::RegionDetector::setBlurKernelSize(), multiscale::analysis::ClusterDetector::setEps(), multiscale::analysis::RegionDetector::setEpsilon(), multiscale::analysis::ClusterDetector::setMinPoints(), multiscale::analysis::RegionDetector::setMorphologicalCloselterations(), multiscale::analysis::RegionDetector::setOriginXCoordinate(), multiscale::analysis::RegionDetector::setOriginYCoordinate(), multiscale::analysis::RegionDetector::setRegionAreaThresh(), and multiscale::analysis::RegionDetector::setThresholdValue().

**6.59.3.47 void Detector::storeOutputImageOnDisk ( ) [protected]**

Store the image with the output results on disk.

Definition at line 205 of file Detector.cpp.

References IMG\_EXTENSION, outputFilepath, and outputImage.

## 6.59.4 Member Data Documentation

**6.59.4.1 double multiscale::analysis::Detector::avgClusterednessDegree [protected]**

For regions: Average degree of clusteredness of all regions

For clusters: Index of clusteredness for all clusters

Definition at line 26 of file Detector.hpp.

Referenced by addAverageMeasuresToPropertyTree(), multiscale::analysis::ClusterDetector::analyseClusters(), multiscale::analysis::ClusterDetector::ClusterDetector(), multiscale::analysis::RegionDetector::computeAverageClusterednessDegree(), - Detector(), outputAveragedMeasuresToCsvFile(), and multiscale::analysis::RegionDetector::RegionDetector().

**6.59.4.2 double multiscale::analysis::Detector::avgDensity [protected]**

For regions: Average density of all regions

For clusters: Average pile up degree of all clusters

Definition at line 31 of file Detector.hpp.

Referenced by addAverageMeasuresToPropertyTree(), multiscale::analysis::ClusterDetector::analyseClusters(), multiscale::analysis::ClusterDetector::ClusterDetector(),

multiscale::analysis::RegionDetector::computeAverageDensity(), Detector(), outputAveragedMeasuresToCsvFile(), and multiscale::analysis::RegionDetector::RegionDetector().

**6.59.4.3 const std::string Detector::CSV\_EXTENSION = ".out" [static, protected]**

Definition at line 303 of file Detector.hpp.

Referenced by outputResultsToCsvFile().

**6.59.4.4 bool multiscale::analysis::Detector::detectMethodCalled [protected]**

Flag for indicating if the detect method was called

Definition at line 44 of file Detector.hpp.

Referenced by detect(), Detector(), and outputResults().

**6.59.4.5 bool multiscale::analysis::Detector::detectorSpecificFieldsInitialised [protected]**

Flag for indicating if the parameters were

Definition at line 45 of file Detector.hpp.

Referenced by Detector(), initialiseDetectorSpecificFieldsIfNotSet(), and setDetectorSpecificFieldsInitialisationFlag().

**6.59.4.6 const std::string Detector::ERR\_INVALID\_IMAGE = "The input image is invalid." [static, protected]**

Definition at line 301 of file Detector.hpp.

Referenced by detect().

**6.59.4.7 const std::string Detector::ERR\_OUTPUT\_FILE = "Unable to create output file." [static, protected]**

Definition at line 300 of file Detector.hpp.

Referenced by outputResultsToCsvFile().

**6.59.4.8 const std::string Detector::ERR\_OUTPUT\_WITHOUT\_DETECT = "Unable to output results if the detect method was not called previously." [static, protected]**

Definition at line 299 of file Detector.hpp.

Referenced by printOutputErrorMessage().

**6.59.4.9 cv::Mat multiscale::analysis::Detector::image [protected]**

Input image

Definition at line 37 of file Detector.hpp.

Referenced by multiscale::analysis::RegionDetector::changeContrastAndBrightness(), multiscale::analysis::SimulationClusterDetector::computePileUpDegreeAtPosition(), multiscale::analysis::RegionDetector::computeRegionDensity(), multiscale::analysis::RegionDetector::createMaskForPolygon(), multiscale::analysis::SimulationClusterDetector::initialiseDetectorSpecificImageDependentFields(), initialiseImage(), initialiseImageOrigin(), multiscale::analysis::SimulationClusterDetector::initialiseThresholdedImage(), multiscale::analysis::SimulationClusterDetector::outputResultsToImage(), multiscale::analysis::RegionDetector::outputResultsToImage(), and ~Detector().

**6.59.4.10 const std::string Detector::IMG\_EXTENSION = ".png" [static, protected]**

Definition at line 304 of file Detector.hpp.

Referenced by storeOutputImageOnDisk().

**6.59.4.11 const int Detector::INTENSITY\_MAX = 255 [static, protected]**

Definition at line 294 of file Detector.hpp.

Referenced by multiscale::analysis::ClusterDetector::ClusterDetector(), multiscale::analysis::RegionDetector::computeAverageIntensity(), multiscale::analysis::RegionDetector::computeRegionDensity(), multiscale::analysis::RegionDetector::createMaskForPolygon(), multiscale::analysis::RegionDetector::outputRegionInnerBordersToImage(), and multiscale::analysis::RegionDetector::outputRegionOuterBorderToImage().

**6.59.4.12 bool multiscale::analysis::Detector::isDebugEnabled [protected]**

Flag for indicating if debug mode is set

Definition at line 40 of file Detector.hpp.

Referenced by detect().

**6.59.4.13 const int Detector::KEY\_ESC = 27 [static, protected]**

Definition at line 309 of file Detector.hpp.

Referenced by detectInDebugMode().

6.59.4.14 `const int Detector::KEY_SAVE = 115` [static, protected]

Definition at line 310 of file Detector.hpp.

6.59.4.15 `const std::string Detector::LABEL_ATTRIBUTE = "<xmlelement>"` [static, protected]

Definition at line 312 of file Detector.hpp.

Referenced by addSpatialEntityTypeToPropertyTree().

6.59.4.16 `const std::string Detector::LABEL_AVG_CLUSTEREDNESS = "avgClusteredness"` [static, protected]

Definition at line 336 of file Detector.hpp.

Referenced by addAverageMeasuresToPropertyTree().

6.59.4.17 `const std::string Detector::LABEL_AVG_DENSITY = "avgDensity"` [static, protected]

Definition at line 337 of file Detector.hpp.

Referenced by addAverageMeasuresToPropertyTree().

6.59.4.18 `const std::string Detector::LABEL_COMMENT = "<xmlelement>"` [static, protected]

Definition at line 313 of file Detector.hpp.

Referenced by outputResultsToXMLFile().

6.59.4.19 `const std::string Detector::LABEL_COMMENT_CONTENTS = "Warning! This  
xml file was automatically generated by a C++ program using the Boost PropertyTree  
library."` [static, protected]

Definition at line 315 of file Detector.hpp.

Referenced by outputResultsToXMLFile().

6.59.4.20 `const std::string Detector::LABEL_EXPERIMENT_TIMEPOINT_NUMERIC_-  
STATE_VARIABLE = "experiment.timepoint.numericStateVariable"` [static, protected]

Definition at line 317 of file Detector.hpp.

Referenced by addNumericStateVariableToPropertyTree().

---

```
6.59.4.21 const std::string Detector::LABEL_EXPERIMENT_TIMEPOINT_-  
    NUMERIC_STATE_VARIABLE_NAME = "name" [static,  
    protected]
```

Definition at line 320 of file Detector.hpp.

Referenced by addNumericStateVariableToPropertyTree().

```
6.59.4.22 const std::string Detector::LABEL_EXPERIMENT_TIMEPOINT_-  
    NUMERIC_STATE_VARIABLE_VALUE = "value" [static,  
    protected]
```

Definition at line 321 of file Detector.hpp.

Referenced by addNumericStateVariableToPropertyTree().

```
6.59.4.23 const std::string Detector::LABEL_EXPERIMENT_TIMEPOINT_S-  
    PATIAL_ENTITY = "experiment.timepoint.spatialEntity" [static,  
    protected]
```

Definition at line 318 of file Detector.hpp.

Referenced by addSpatialEntitiesToPropertyTree().

```
6.59.4.24 const std::string Detector::LABEL_SPATIAL_ENTITY_ANGLE = "angle"  
    [static, protected]
```

Definition at line 329 of file Detector.hpp.

Referenced by addSpatialEntityPropertiesToTree().

```
6.59.4.25 const std::string Detector::LABEL_SPATIAL_ENTITY_AREA = "area"  
    [static, protected]
```

Definition at line 326 of file Detector.hpp.

Referenced by addSpatialEntityPropertiesToTree().

```
6.59.4.26 const std::string Detector::LABEL_SPATIAL_ENTITY_CENTROID_X =  
    "centroidX" [static, protected]
```

Definition at line 333 of file Detector.hpp.

Referenced by addSpatialEntityPropertiesToTree().

```
6.59.4.27 const std::string Detector::LABEL_SPATIAL_ENTITY_CENTROID_Y =
    "centroidY" [static, protected]
```

Definition at line 334 of file Detector.hpp.

Referenced by addSpatialEntityPropertiesToTree().

```
6.59.4.28 const std::string Detector::LABEL_SPATIAL_ENTITY_CIRCLE_MEASURE
    = "circleMeasure" [static, protected]
```

Definition at line 332 of file Detector.hpp.

Referenced by addSpatialEntityPropertiesToTree().

```
6.59.4.29 const std::string Detector::LABEL_SPATIAL_ENTITY_CLUSTEREDNESS =
    "clusteredness" [static, protected]
```

Definition at line 324 of file Detector.hpp.

Referenced by addSpatialEntityPropertiesToTree().

```
6.59.4.30 const std::string Detector::LABEL_SPATIAL_ENTITY_DENSITY = "density"
[static, protected]
```

Definition at line 325 of file Detector.hpp.

Referenced by addSpatialEntityPropertiesToTree().

```
6.59.4.31 const std::string Detector::LABEL_SPATIAL_ENTITY_DISTANCE_FROM_ORIGIN =
    "distanceFromOrigin" [static,
protected]
```

Definition at line 328 of file Detector.hpp.

Referenced by addSpatialEntityPropertiesToTree().

```
6.59.4.32 const std::string Detector::LABEL_SPATIAL_ENTITY_PERIMETER =
    "perimeter" [static, protected]
```

Definition at line 327 of file Detector.hpp.

Referenced by addSpatialEntityPropertiesToTree().

```
6.59.4.33 const std::string Detector::LABEL_SPATIAL_ENTITY_RECTANGLE_MEASURE =
    "rectangleMeasure" [static,
protected]
```

Definition at line 331 of file Detector.hpp.

Referenced by addSpatialEntityPropertiesToTree().

**6.59.4.34 const std::string Detector::LABEL\_SPATIAL\_ENTITY\_SPATIAL\_TYPE = "spatialType" [static, protected]**

Definition at line 323 of file Detector.hpp.

Referenced by addSpatialEntityTypeToPropertyTree().

**6.59.4.35 const std::string Detector::LABEL\_SPATIAL\_ENTITY\_-\_TRIANGLE\_MEASURE = "triangleMeasure" [static, protected]**

Definition at line 330 of file Detector.hpp.

Referenced by addSpatialEntityPropertiesToTree().

**6.59.4.36 cv::Point2f multiscale::analysis::Detector::origin [protected]**

The point representing the origin

Definition at line 47 of file Detector.hpp.

Referenced by computeDistanceFromOrigin(), computePolygonAngle(), multiscale::analysis::RegionDetector::getOriginXCoordinate(), multiscale::analysis::RegionDetector::getOriginYCoordinate(), initialisImageOrigin(), multiscale::analysis::RegionDetector::setOriginXCoordinate(), and multiscale::analysis::RegionDetector::setOriginYCoordinate().

**6.59.4.37 const std::string Detector::OUTPUT\_CLUSTEREDNESS = "Average clusteredness degree: " [static, protected]**

Definition at line 296 of file Detector.hpp.

Referenced by outputAveragedMeasuresToCsvFile().

**6.59.4.38 const std::string Detector::OUTPUT\_DENSITY = "Average density: " [static, protected]**

Definition at line 297 of file Detector.hpp.

Referenced by outputAveragedMeasuresToCsvFile().

**6.59.4.39 std::string multiscale::analysis::Detector::outputFilepath [protected]**

Path of the output file

Definition at line 38 of file Detector.hpp.

Referenced by outputResults(), outputResultsToCsvFile(), outputResultsToXMLFile(), and storeOutputImageOnDisk().

#### 6.59.4.40 cv::Mat multiscale::analysis::Detector::outputImage [protected]

Image for displaying the results

Definition at line 42 of file Detector.hpp.

Referenced by displayResultsInWindow(), multiscale::analysis::RegionDetector::outputRegionToImage(), multiscale::analysis::SimulationClusterDetector::outputResultsToImage(), multiscale::analysis::RegionDetector::outputResultsToImage(), storeOutputImageOnDisk(), and ~Detector().

#### 6.59.4.41 const std::string Detector::WIN\_OUTPUT\_IMAGE = "Output image" [static, protected]

Definition at line 307 of file Detector.hpp.

Referenced by multiscale::analysis::ClusterDetector::createDetectorSpecificTrackbars(), multiscale::analysis::RegionDetector::createDetectorSpecificTrackbars(), createTrackbarsWindow(), and displayResultsInWindow().

#### 6.59.4.42 const std::string Detector::XML\_EXTENSION = ".xml" [static, protected]

Definition at line 305 of file Detector.hpp.

Referenced by outputResultsToXMLFile().

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

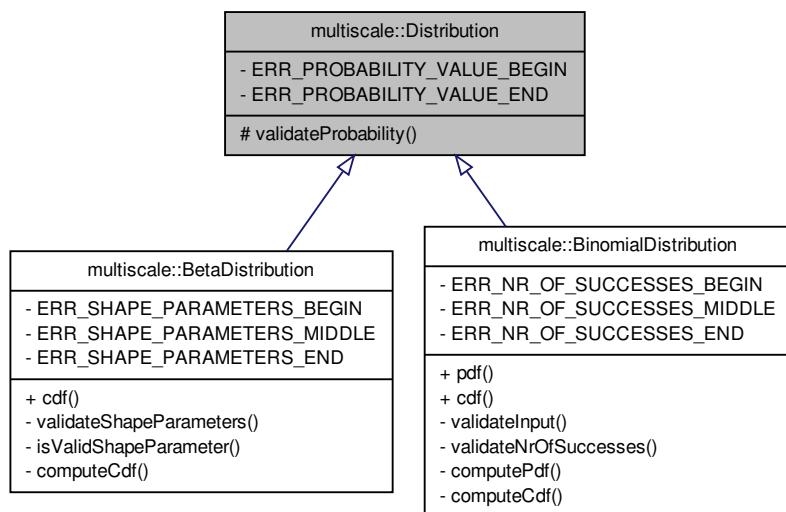
- Detector.hpp
- Detector.cpp

## 6.60 multiscale::Distribution Class Reference

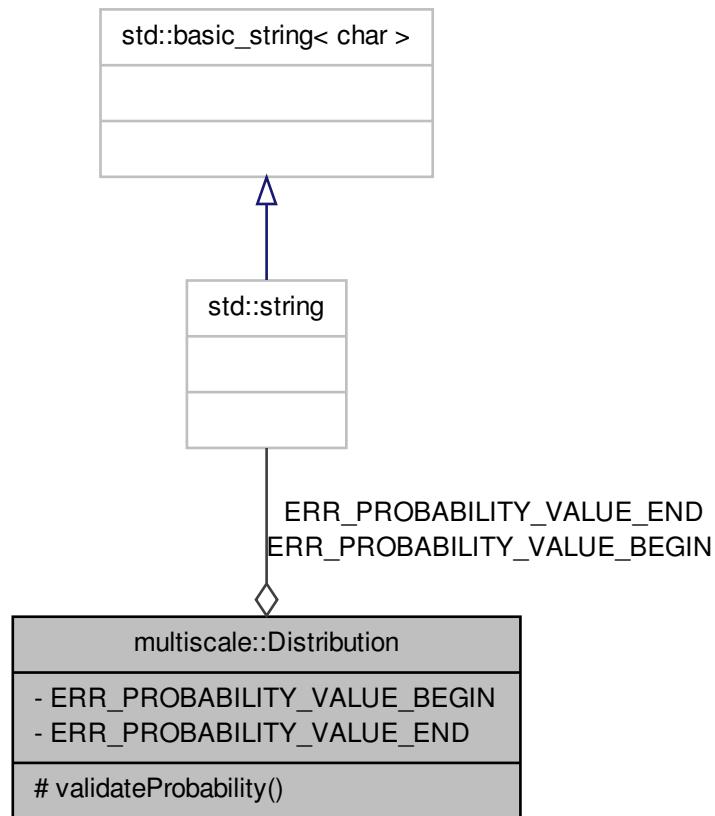
---

```
#include <Distribution.hpp>
```

Inheritance diagram for multiscale::Distribution:



Collaboration diagram for multiscale::Distribution:



### Static Protected Member Functions

- static void [validateProbability](#) (double probability)

*Check if the value of the probability is valid.*

### Static Private Attributes

- static const std::string [ERR\\_PROBABILITY\\_VALUE\\_BEGIN](#) = "The given probability value ("
- static const std::string [ERR\\_PROBABILITY\\_VALUE\\_END](#) = ") should be between 0 and 1."

### 6.60.1 Detailed Description

Definition at line 10 of file Distribution.hpp.

### 6.60.2 Member Function Documentation

**6.60.2.1 void Distribution::validateProbability ( double *probability* ) [static, protected]**

Check if the value of the probability is valid.

#### Parameters

<i>probability</i>	The value of the probability
--------------------	------------------------------

Definition at line 8 of file Distribution.cpp.

References ERR\_PROBABILITY\_VALUE\_BEGIN, ERR\_PROBABILITY\_VALUE\_END, and multiscale::StringManipulator::toString().

Referenced by multiscale::BetaDistribution::cdf(), and multiscale::BinomialDistribution::validateInput().

### 6.60.3 Member Data Documentation

**6.60.3.1 const std::string Distribution::ERR\_PROBABILITY\_VALUE\_BEGIN = "The given probability value (" [static, private]**

Definition at line 23 of file Distribution.hpp.

Referenced by validateProbability().

**6.60.3.2 const std::string Distribution::ERR\_PROBABILITY\_VALUE\_END = ") should be between 0 and 1." [static, private]**

Definition at line 24 of file Distribution.hpp.

Referenced by validateProbability().

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

- Distribution.hpp
- Distribution.cpp

## 6.61 multiscale::DivisionOperation Class Reference

Functor representing a division operation.

```
#include <Numeric.hpp>
```

## Public Member Functions

- template<typename Operand >  
Operand **operator()** (Operand operand1, Operand operand2) const  
*Divide the two operands.*

### 6.61.1 Detailed Description

Functor representing a division operation.

Definition at line 553 of file Numeric.hpp.

### 6.61.2 Member Function Documentation

- 6.61.2.1 template<typename Operand > Operand multiscale::DivisionOperation::operator() (  
Operand *operand1*, Operand *operand2* ) const [inline]

Divide the two operands.

#### Parameters

<i>operand1</i>	The first operand
<i>operand2</i>	The second operand

Definition at line 563 of file Numeric.hpp.

References multiscale::Numeric::almostEqual().

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

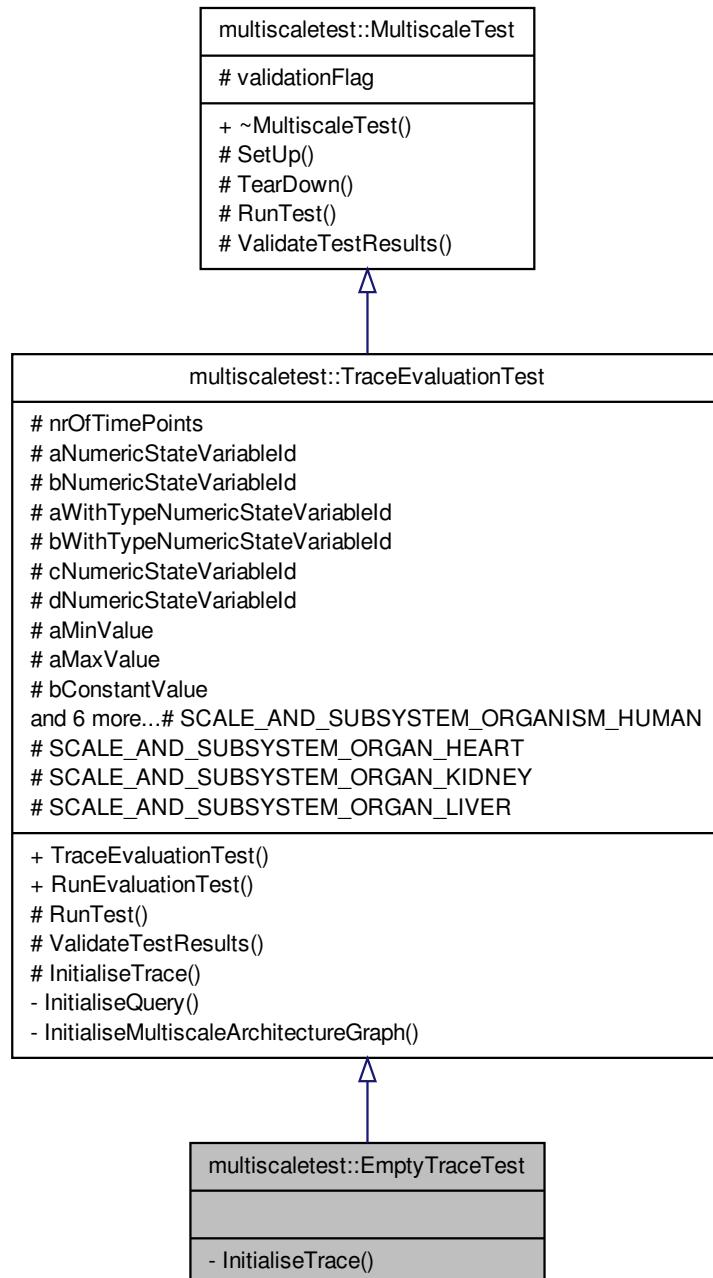
- Numeric.hpp

## 6.62 multiscaletest::EmptyTraceTest Class Reference

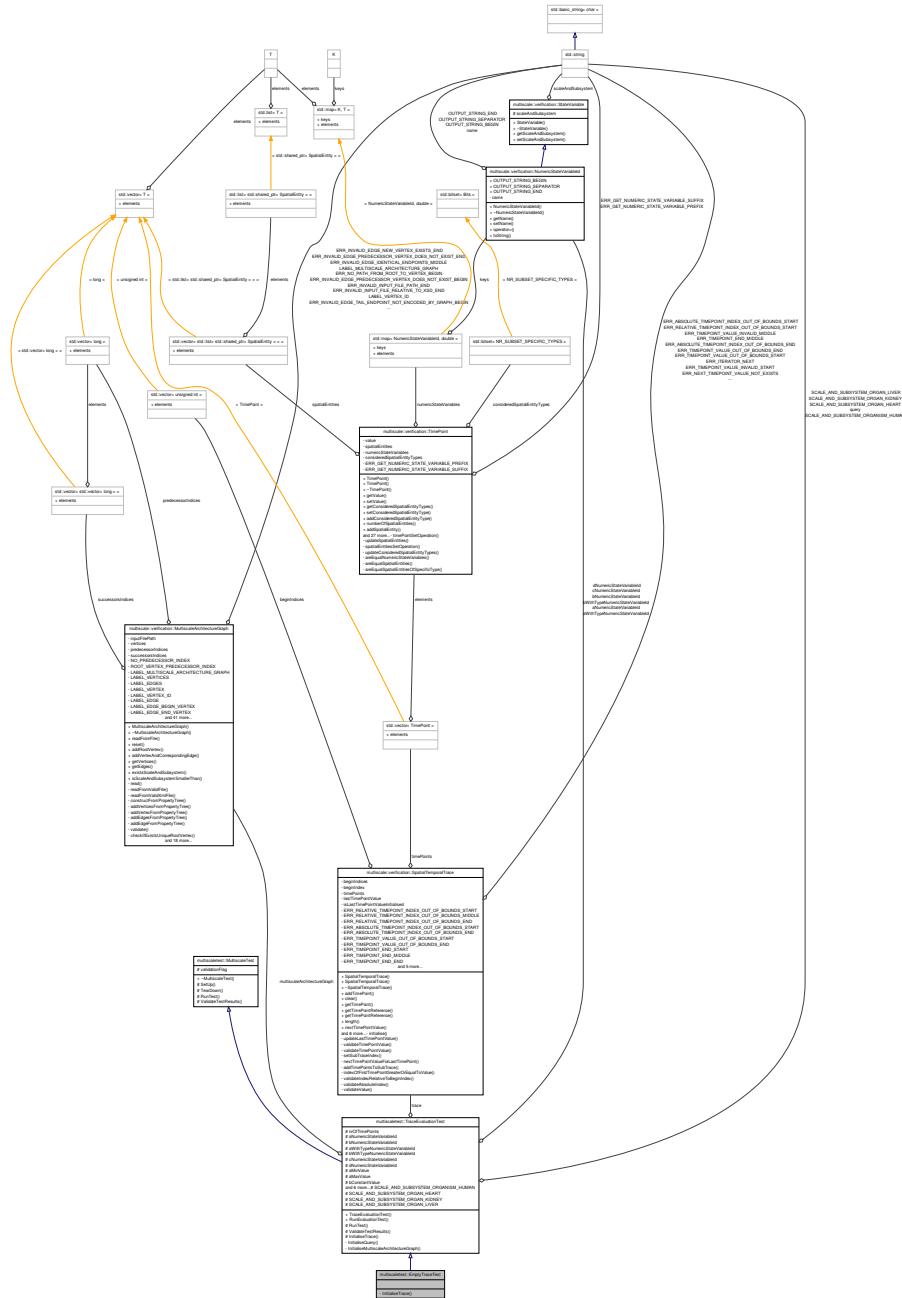
Class for testing evaluation of empty traces.

```
#include <EmptyTraceTest.hpp>
```

Inheritance diagram for multiscaletest::EmptyTraceTest:



## Collaboration diagram for multiscaletest::EmptyTraceTest:



## Private Member Functions

- virtual void InitialiseTrace () override

*Initialise the trace.*

### 6.62.1 Detailed Description

Class for testing evaluation of empty traces.

Definition at line 22 of file `EmptyTraceTest.hpp`.

### 6.62.2 Member Function Documentation

#### 6.62.2.1 `void multiscaletest::EmptyTraceTest::InitialiseTrace( ) [override, private, virtual]`

Initialise the trace.

Implements [multiscaletest::TraceEvaluationTest](#).

Definition at line 31 of file `EmptyTraceTest.hpp`.

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

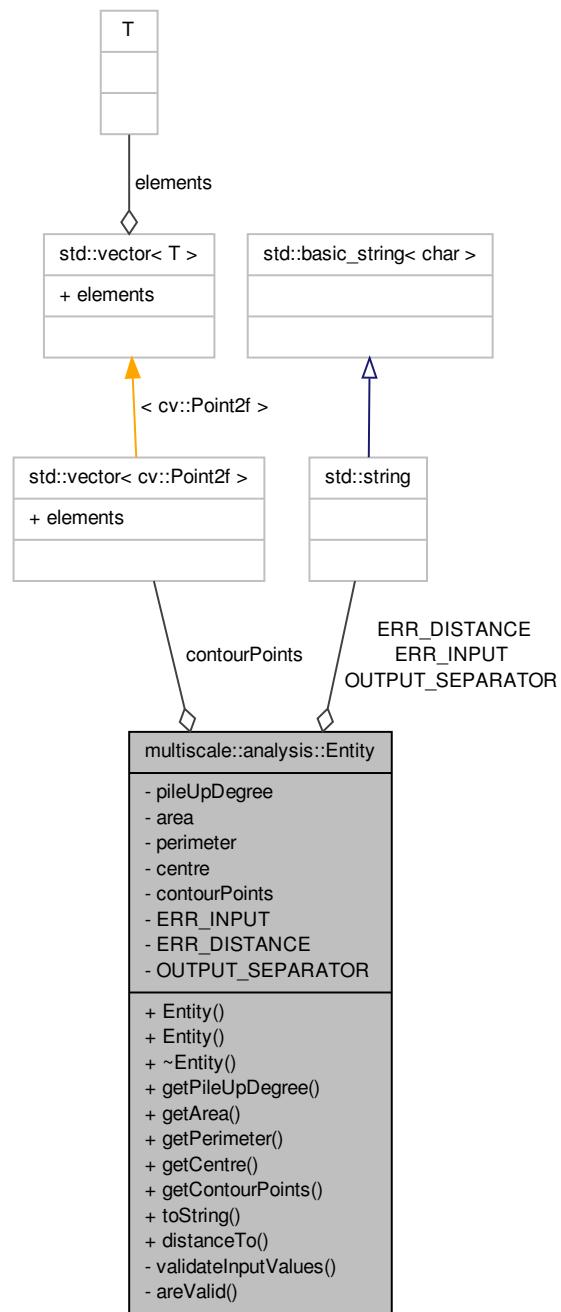
- `EmptyTraceTest.hpp`

## 6.63 `multiscale::analysis::Entity` Class Reference

Class for representing an entity in an image (e.g. cell, organism etc.)

```
#include <Entity.hpp>
```

Collaboration diagram for multiscale::analysis::Entity:



## Public Member Functions

- `Entity` (double `pileUpDegree`, double `area`, double `perimeter`, const `cv::Point2f &centre`, const `std::vector<cv::Point2f> &contourPoints`)
- `Entity` (const `Entity` &`entity`)
- `~Entity` ()
- double `getPileUpDegree` () const  
`Get the degree of pile up.`
- double `getArea` () const  
`Get the area.`
- double `getPerimeter` () const  
`Get the perimeter.`
- `cv::Point2f getCentre` () const  
`Get the point defining the centre of the entity.`
- `std::vector<cv::Point2f> getContourPoints` () const  
`Get the set of points defining the contour of the entity.`
- `std::string toString` ()  
`Get a string representation of all the field values.`
- double `distanceTo` (const `Entity` &`entity`) const  
`Get the distance between this entity and another one.`

## Private Member Functions

- void `validateInputValues` (double `pileUpDegree`, double `area`, double `perimeter`, const `cv::Point2f &centre`, const `std::vector<cv::Point2f> &contourPoints`)
- bool `isValid` (double `pileUpDegree`, double `area`, double `perimeter`, const `cv::Point2f &centre`, const `std::vector<cv::Point2f> &contourPoints`)  
`Check if the provided degree of pile up, area, centre and contour points are valid.`

## Private Attributes

- double `pileUpDegree`
- double `area`
- double `perimeter`
- `cv::Point2f centre`
- `std::vector<cv::Point2f> contourPoints`

## Static Private Attributes

- static const `std::string ERR_INPUT` = "Invalid input parameters were provided to the constructor."
- static const `std::string ERR_DISTANCE` = "The distance to an object of a different type cannot be computed."
- static const `std::string OUTPUT_SEPARATOR` = ","

### 6.63.1 Detailed Description

Class for representing an entity in an image (e.g. cell, organism etc.)

Definition at line 14 of file Entity.hpp.

### 6.63.2 Constructor & Destructor Documentation

**6.63.2.1 Entity::Entity ( double *pileUpDegree*, double *area*, double *perimeter*, const cv::Point2f & *centre*, const std::vector<cv::Point2f> & *contourPoints* )**

Definition at line 9 of file Entity.cpp.

References area, centre, contourPoints, perimeter, pileUpDegree, and validateInputValues().

**6.63.2.2 Entity::Entity ( const Entity & *entity* )**

Definition at line 20 of file Entity.cpp.

References area, centre, contourPoints, perimeter, pileUpDegree, and validateInputValues().

**6.63.2.3 Entity::~Entity ( )**

Definition at line 30 of file Entity.cpp.

### 6.63.3 Member Function Documentation

**6.63.3.1 bool Entity::isValid ( double *pileUpDegree*, double *area*, double *perimeter*, const cv::Point2f & *centre*, const std::vector<cv::Point2f> & *contourPoints* ) [private]**

Check if the provided degree of pile up, area, centre and contour points are valid.

#### Parameters

<i>pileUpDegree</i>	Degree of pile up
<i>area</i>	Area
<i>perimeter</i>	Perimeter
<i>centre</i>	Centre of the entity
<i>contourPoints</i>	Points defining the contour of the entity

Definition at line 78 of file Entity.cpp.

References multiscale::Numeric::greaterOrEqual(), and multiscale::Numeric::lessOrEqual().

Referenced by validateInputValues().

#### 6.63.3.2 double Entity::distanceTo ( const Entity & entity ) const

Get the distance between this entity and another one.

Definition at line 62 of file Entity.cpp.

References centre, and multiscale::Geometry2D::distanceBtwPoints().

#### 6.63.3.3 double Entity::getArea ( ) const

Get the area.

Definition at line 36 of file Entity.cpp.

References area.

#### 6.63.3.4 cv::Point2f Entity::getCentre ( ) const

Get the point defining the centre of the entity.

Definition at line 44 of file Entity.cpp.

References centre.

#### 6.63.3.5 std::vector< cv::Point2f > Entity::getContourPoints ( ) const

Get the set of points defining the contour of the entity.

Definition at line 48 of file Entity.cpp.

References contourPoints.

#### 6.63.3.6 double Entity::getPerimeter ( ) const

Get the perimeter.

Definition at line 40 of file Entity.cpp.

References perimeter.

#### 6.63.3.7 double Entity::getPileUpDegree ( ) const

Get the degree of pile up.

Definition at line 32 of file Entity.cpp.

References pileUpDegree.

## 6.63.3.8 std::string Entity::toString( )

Get a string representation of all the field values.

Definition at line 52 of file Entity.cpp.

References centre, OUTPUT\_SEPARATOR, and pileUpDegree.

## 6.63.3.9 void Entity::validateInputValues ( double pileUpDegree, double area, double perimeter, const cv::Point2f &amp; centre, const std::vector&lt; cv::Point2f &gt; &amp; contourPoints ) [private]

## Parameters

<i>pileUpDegree</i>	Degree of pile up
<i>area</i>	Area
<i>perimeter</i>	Perimeter
<i>centre</i>	Centre of the entity
<i>contourPoints</i>	Points defining the contour of the entity

Definition at line 71 of file Entity.cpp.

References isValid(), and ERR\_INPUT.

Referenced by Entity().

## 6.63.4 Member Data Documentation

## 6.63.4.1 double multiscale::analysis::Entity::area [private]

Area of the entity

Definition at line 21 of file Entity.hpp.

Referenced by Entity(), and getArea().

## 6.63.4.2 cv::Point2f multiscale::analysis::Entity::centre [private]

Point defining the centre of the entity

Definition at line 26 of file Entity.hpp.

Referenced by distanceTo(), Entity(), getCentre(), and toString().

## 6.63.4.3 std::vector&lt;cv::Point2f&gt; multiscale::analysis::Entity::contourPoints [private]

Set of points defining the contour of the entity

Definition at line 28 of file Entity.hpp.

Referenced by Entity(), and getContourPoints().

**6.63.4.4 const std::string Entity::ERR\_DISTANCE = "The distance to an object of a different type cannot be computed." [static, private]**

Definition at line 86 of file Entity.hpp.

**6.63.4.5 const std::string Entity::ERR\_INPUT = "Invalid input parameters were provided to the constructor." [static, private]**

Definition at line 85 of file Entity.hpp.

Referenced by validateInputValues().

**6.63.4.6 const std::string Entity::OUTPUT\_SEPARATOR = "" [static, private]**

Definition at line 88 of file Entity.hpp.

Referenced by toString().

**6.63.4.7 double multiscale::analysis::Entity::perimeter [private]**

Perimeter of the entity

Definition at line 23 of file Entity.hpp.

Referenced by Entity(), and getPerimeter().

**6.63.4.8 double multiscale::analysis::Entity::pileUpDegree [private]**

Degree of pile up (relevant only if entities can pile up onto each other)

Definition at line 19 of file Entity.hpp.

Referenced by Entity(), getPileUpDegree(), and toString().

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

- Entity.hpp
- Entity.cpp

## **6.64 multiscale::verification::EquivalenceConstraintAttribute - Class Reference**

Class for representing an "equivalence" constraint attribute.

```
#include <EquivalenceConstraintAttribute.hpp>
```

**Public Attributes**

- [ConstraintAttributeType constraint](#)

**6.64.1 Detailed Description**

Class for representing an "equivalence" constraint attribute.

Definition at line 14 of file EquivalenceConstraintAttribute.hpp.

**6.64.2 Member Data Documentation****6.64.2.1 ConstraintAttributeType multiscale::verification::EquivalenceConstraintAttribute::constraint**

The constraint following the "equivalence" operator

Definition at line 18 of file EquivalenceConstraintAttribute.hpp.

Referenced by multiscale::verification::ConstraintVisitor::operator()().

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

- [EquivalenceConstraintAttribute.hpp](#)

**6.65 multiscale::verification::EquivalenceLogicPropertyAttribute Class Reference**

Class for representing an "equivalence" logic property attribute.

```
#include <EquivalenceLogicPropertyAttribute.hpp>
```

**Public Attributes**

- [LogicPropertyAttributeType logicProperty](#)

**6.65.1 Detailed Description**

Class for representing an "equivalence" logic property attribute.

Definition at line 14 of file EquivalenceLogicPropertyAttribute.hpp.

### 6.65.2 Member Data Documentation

#### 6.65.2.1 LogicPropertyAttributeType multiscale::verification::EquivalenceLogicPropertyAttribute::logicProperty

The logic property following the "equivalence" operator

Definition at line 18 of file EquivalenceLogicPropertyAttribute.hpp.

Referenced by multiscale::verification::LogicPropertyVisitor::operator()().

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

- EquivalenceLogicPropertyAttribute.hpp

## 6.66 EuclideanDataPoint Class Reference

### Public Member Functions

- [EuclideanDataPoint \(double x, double y\)](#)
- [EuclideanDataPoint \(const EuclideanDataPoint &point\)](#)
- [~EuclideanDataPoint \(\)](#)
- double [distanceTo \(const EuclideanDataPoint &point\) const](#)

### Private Attributes

- double [x](#)
- double [y](#)

#### 6.66.1 Detailed Description

Definition at line 12 of file DBSCANTest.cpp.

### 6.66.2 Constructor & Destructor Documentation

#### 6.66.2.1 EuclideanDataPoint::EuclideanDataPoint ( double x, double y ) [inline]

Definition at line 19 of file DBSCANTest.cpp.

#### 6.66.2.2 EuclideanDataPoint::EuclideanDataPoint ( const EuclideanDataPoint & point ) [inline]

Definition at line 20 of file DBSCANTest.cpp.

**6.66.2.3 EuclideanDataPoint::~EuclideanDataPoint( ) [inline]**

Definition at line 21 of file DBSCANTest.cpp.

**6.66.3 Member Function Documentation****6.66.3.1 double EuclideanDataPoint::distanceTo ( const EuclideanDataPoint & point ) const [inline]**

Definition at line 23 of file DBSCANTest.cpp.

References multiscale::Geometry2D::distanceBtwPoints(), x, and y.

**6.66.4 Member Data Documentation****6.66.4.1 double EuclideanDataPoint::x [private]**

Definition at line 15 of file DBSCANTest.cpp.

Referenced by distanceTo().

**6.66.4.2 double EuclideanDataPoint::y [private]**

Definition at line 16 of file DBSCANTest.cpp.

Referenced by distanceTo().

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

- DBSCANTest.cpp

**6.67 multiscale::ExceptionHandler Class Reference**

Exception handler class.

```
#include <ExceptionHandler.hpp>
```

**Static Public Member Functions**

- static void **printDetailedErrorMessage** (const std::exception &ex)  
*Print the detailed error message.*
- static void **printRawErrorMessage** (const **MultiscaleException** &ex)  
*Print the raw error message.*
- static void **printHelpMessage** ()  
*Print the help message.*

### 6.67.1 Detailed Description

Exception handler class.

Definition at line 14 of file ExceptionHandler.hpp.

### 6.67.2 Member Function Documentation

**6.67.2.1 static void multiscale::ExceptionHandler::printDetailedErrorMessage ( const std::exception & ex ) [inline, static]**

Print the detailed error message.

The error message is printed using the ex.what() method

#### Parameters

<code>ex</code>	Exception
-----------------	-----------

Definition at line 23 of file ExceptionHandler.hpp.

References multiscale::ERR\_MSG.

Referenced by multiscale::OperatingSystem::executeProgram(), and multiscale::verification::ModelCheckingManager::parseLogicProperty().

**6.67.2.2 static void multiscale::ExceptionHandler::printHelpMessage ( ) [inline, static]**

Print the help message.

A message is printed informing the user that (s)he should run the program with the "-h" command line argument for useful execution information.

Definition at line 45 of file ExceptionHandler.hpp.

**6.67.2.3 static void multiscale::ExceptionHandler::printRawErrorMessage ( const MultiscaleException & ex ) [inline, static]**

Print the raw error message.

The error message is printed using the ex.rawMessage() method

#### Parameters

<code>ex</code>	Exception
-----------------	-----------

Definition at line 34 of file ExceptionHandler.hpp.

References multiscale::ERR\_MSG, and multiscale::MultiscaleException::rawMessage().

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

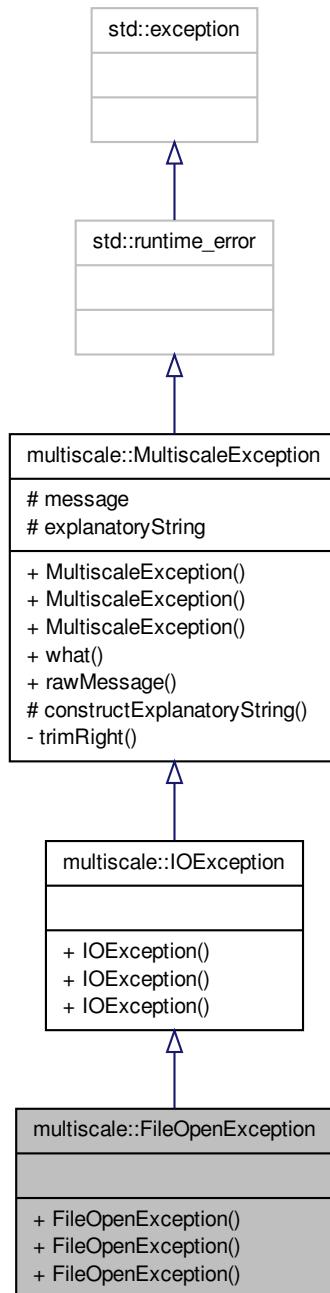
- `ExceptionHandler.hpp`

## 6.68 multiscale::FileOpenException Class Reference

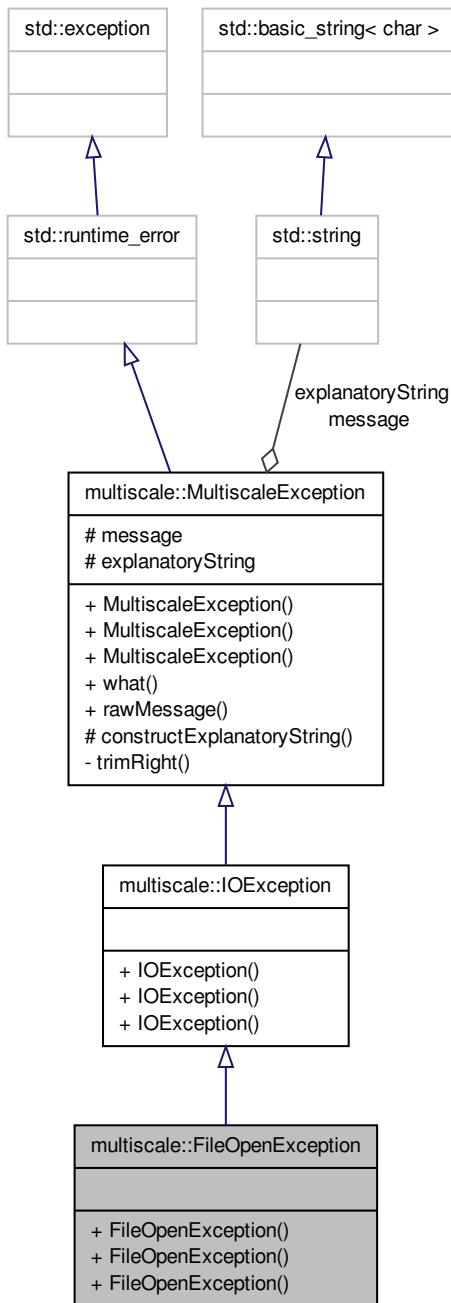
Class for representing exceptions when opening a file.

```
#include <FileOpenException.hpp>
```

Inheritance diagram for multiscale::FileOpenException:



Collaboration diagram for multiscale::FileOpenException:



## Public Member Functions

- [FileOpenException \(\)](#)
- [FileOpenException \(const std::string &file, int line, const std::string &msg\)](#)
- [FileOpenException \(const std::string &file, int line, const char \\*msg\)](#)

### 6.68.1 Detailed Description

Class for representing exceptions when opening a file.

Definition at line 12 of file FileOpenException.hpp.

### 6.68.2 Constructor & Destructor Documentation

#### 6.68.2.1 multiscale::FileOpenException::FileOpenException( ) [inline]

Definition at line 16 of file FileOpenException.hpp.

#### 6.68.2.2 multiscale::FileOpenException::FileOpenException( const std::string & file, int line, const std::string & msg ) [inline, explicit]

Definition at line 18 of file FileOpenException.hpp.

#### 6.68.2.3 multiscale::FileOpenException::FileOpenException( const std::string & file, int line, const char \* msg ) [inline, explicit]

Definition at line 23 of file FileOpenException.hpp.

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

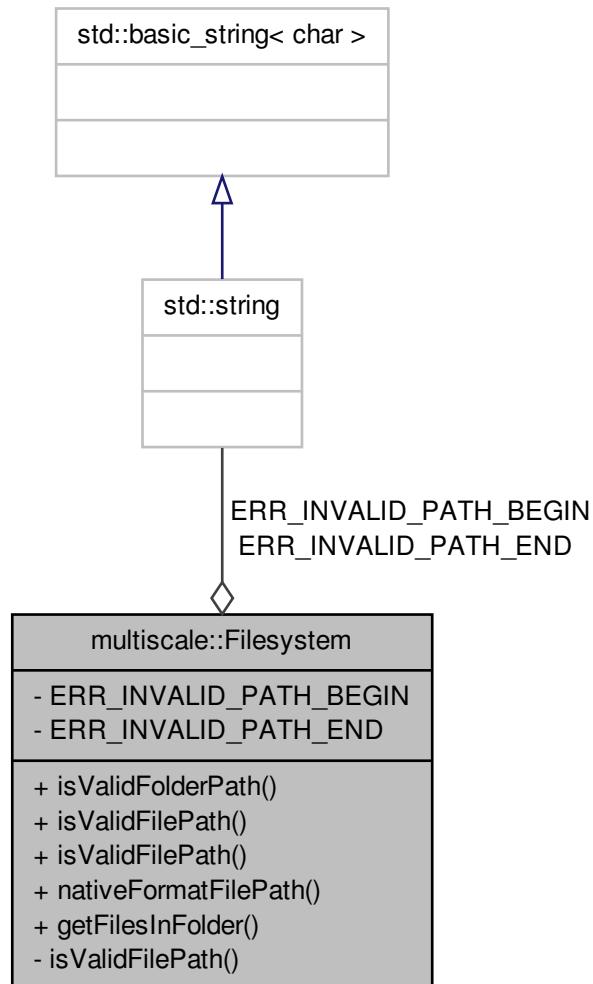
- [FileOpenException.hpp](#)

## 6.69 multiscale::Filesystem Class Reference

Class containing methods for interacting with the filesystem.

```
#include <Filesystem.hpp>
```

Collaboration diagram for multiscale::Filesystem:



### Static Public Member Functions

- static bool `isValidFolderPath` (const std::string &path)  
*Check if the given path is a valid folder path.*
- static bool `isValidFilePath` (const std::string &path)  
*Check if the given path is a valid file path.*
- static bool `isValidFilePath` (const std::string &path, const std::string &extension)

*Check if the given path is a valid file path and has the given extension.*

- static std::string [nativeFormatFilePath](#) (const std::string &path)

*Return the given path as an absolute path in native format.*

- static std::vector< std::string > [getFilesInFolder](#) (const std::string &folderPath, const std::string &extension)

*Get the list of files with the given extension in the provided folder.*

### Static Private Member Functions

- static bool [isValidFilePath](#) (const fs::path &path)

*Check if the given path is a valid file path.*

### Static Private Attributes

- static const std::string [ERR\\_INVALID\\_PATH\\_BEGIN](#) = "The given input file path ("
- static const std::string [ERR\\_INVALID\\_PATH\\_END](#) = ") does not exist or does not point to a regular file. Please change."

#### 6.69.1 Detailed Description

Class containing methods for interacting with the filesystem.

This class is using the Boost::Filesystem library.

Definition at line 18 of file Filesystem.hpp.

#### 6.69.2 Member Function Documentation

##### 6.69.2.1 std::vector< std::string > [Filesystem::getFilesInFolder](#) ( const std::string & folderPath, const std::string & extension ) [static]

Get the list of files with the given extension in the provided folder.

Precondition: The provided folder path points to a directory.

#### Parameters

<code>FolderPath</code>	The path to the folder
<code>extension</code>	The given extension

Definition at line 46 of file Filesystem.cpp.

References [isValidFolderPath\(\)](#).

**6.69.2.2 bool Filesystem::isValidFilePath ( const std::string & *path* ) [static]**

Check if the given path is a valid file path.

A file path is valid if it points to a regular file.

**Parameters**

<i>path</i>	The given path
-------------	----------------

Definition at line 17 of file Filesystem.cpp.

Referenced by multiscale::OperatingSystem::executeProgramAndVerifyPath(), isValidFilePath(), multiscale::verification::MultiscaleArchitectureGraph::read(), multiscale::verification::LogicPropertyDataReader::readLogicPropertiesFromFile(), multiscale::verification::TemporalDataReader::readTimeseriesFromFile(), multiscale::XmlValidator::validateXmlFilepath(), and multiscale::XmlValidator::validateXmlSchemaPath().

**6.69.2.3 bool Filesystem::isValidFilePath ( const std::string & *path*, const std::string & *extension* ) [static]**

Check if the given path is a valid file path and has the given extension.

A file path is valid if it points to a regular file with the given extension.

**Parameters**

<i>path</i>	The given path
<i>extension</i>	The given extension

Definition at line 23 of file Filesystem.cpp.

References isValidFilePath().

**6.69.2.4 bool Filesystem::isValidFilePath ( const fs::path & *path* ) [static, private]**

Check if the given path is a valid file path.

**Parameters**

<i>path</i>	The given path
-------------	----------------

Definition at line 64 of file Filesystem.cpp.

**6.69.2.5 bool Filesystem::isValidFolderPath ( const std::string & *path* ) [static]**

Check if the given path is a valid folder path.

A folder path is valid if it points to a directory.

Parameters

<i>path</i>	The given path
-------------	----------------

Definition at line 7 of file Filesystem.cpp.

Referenced by `getFilesInFolder()`, and `multiscale::verification::SpatialTemporalDataReader::validateFolderPath()`.

**6.69.2.6 std::string Filesystem::nativeFormatFilePath ( const std::string & *path* ) [static]**

Return the given path as an absolute path in native format.

Precondition: The given path points to a regular file.

Parameters

<i>path</i>	The given path
-------------	----------------

Definition at line 33 of file Filesystem.cpp.

References `ERR_INVALID_PATH_BEGIN`, and `ERR_INVALID_PATH_END`.

### 6.69.3 Member Data Documentation

**6.69.3.1 const std::string Filesystem::ERR\_INVALID\_PATH\_BEGIN = "The given input file path (" [static, private]**

Definition at line 75 of file Filesystem.hpp.

Referenced by `nativeFormatFilePath()`.

**6.69.3.2 const std::string Filesystem::ERR\_INVALID\_PATH\_END = ") does not exist or does not point to a regular file. Please change." [static, private]**

Definition at line 76 of file Filesystem.hpp.

Referenced by `nativeFormatFilePath()`.

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

- `Filesystem.hpp`
- `Filesystem.cpp`

## **6.70 multiscale::verification::FilterNumericMeasureAttribute Class Reference**

Class for representing a filter numeric measure.

```
#include <FilterNumericMeasureAttribute.hpp>
```

### **Public Attributes**

- `FilterNumericMeasureAttributeType filterNumericMeasure`

#### **6.70.1 Detailed Description**

Class for representing a filter numeric measure.

Definition at line 32 of file FilterNumericMeasureAttribute.hpp.

#### **6.70.2 Member Data Documentation**

##### **6.70.2.1 FilterNumericMeasureAttributeType multiscale::verification::Filter- NumericMeasureAttribute::filterNumericMeasure**

The filter numeric measure

Definition at line 36 of file FilterNumericMeasureAttribute.hpp.

Referenced by `multiscale::verification::FilterNumericVisitor::operator()()`.

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

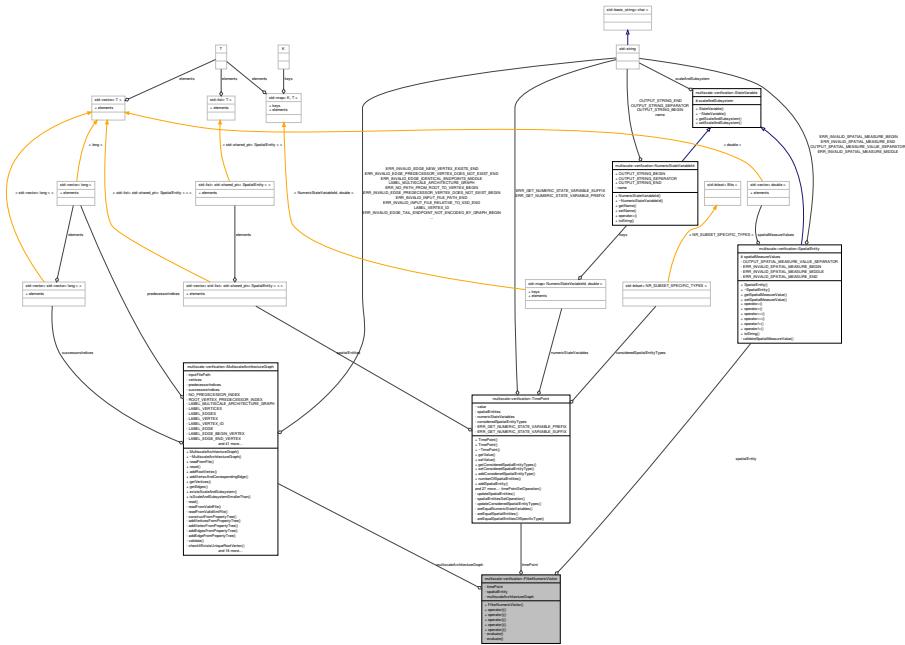
- `FilterNumericMeasureAttribute.hpp`

## **6.71 multiscale::verification::FilterNumericVisitor Class Reference**

Class for evaluating filter numeric measures.

```
#include <FilterNumericVisitor.hpp>
```

Collaboration diagram for multiscale::verification::FilterNumericVisitor:



## Public Member Functions

- `FilterNumericVisitor (TimePoint &timePoint, const SpatialEntity &spatialEntity, const MultiscaleArchitectureGraph &multiscaleArchitectureGraph)`
- double `operator() (const FilterNumericMeasureAttribute &filterNumericMeasure) const`

*Overloading the "()" operator for the `FilterNumericMeasureAttribute` alternative.*
- double `operator() (const PrimaryNumericMeasureAttribute &primaryNumericMeasure) const`

*Overloading the "()" operator for the `PrimaryNumericMeasureAttribute` alternative.*
- double `operator() (const SpatialMeasureAttribute &spatialMeasure) const`

*Overloading the "()" operator for the `SpatialMeasureAttribute` alternative.*
- double `operator() (const UnaryNumericFilterAttribute &unaryNumericFilter) const`

*Overloading the "()" operator for the `UnaryNumericFilterAttribute` alternative.*
- double `operator() (const BinaryNumericFilterAttribute &binaryNumericFilter) const`

*Overloading the "()" operator for the `BinaryNumericFilterAttribute` alternative.*

### Private Member Functions

- double `evaluate` (const `FilterNumericMeasureAttributeType` &`filterNumericMeasure`) const  
*Evaluate the given filter numeric measure considering the timePoint and spatialEntity fields.*
- double `evaluate` (const `PrimaryNumericMeasureAttributeType` &`primaryNumericMeasure`) const  
*Evaluate the given primary numeric measure considering the timePoint field.*

### Private Attributes

- `TimePoint` & `timePoint`
- const `SpatialEntity` & `spatialEntity`
- const `MultiscaleArchitectureGraph` & `multiscaleArchitectureGraph`

#### 6.71.1 Detailed Description

Class for evaluating filter numeric measures.

Definition at line 16 of file FilterNumericVisitor.hpp.

#### 6.71.2 Constructor & Destructor Documentation

**6.71.2.1 multiscale::verification::FilterNumericVisitor::FilterNumericVisitor**  
`( TimePoint & timePoint, const SpatialEntity & spatialEntity, const MultiscaleArchitectureGraph & multiscaleArchitectureGraph ) [inline]`

Definition at line 29 of file FilterNumericVisitor.hpp.

Referenced by `evaluate()`.

#### 6.71.3 Member Function Documentation

**6.71.3.1 double multiscale::verification::FilterNumericVisitor::evaluate ( const FilterNumericMeasureAttributeType & filterNumericMeasure ) const [inline, private]**

Evaluate the given filter numeric measure considering the timePoint and spatialEntity fields.

##### Parameters

<code>filter-Numeric-Measure</code>	The given filter numeric measure
-------------------------------------	----------------------------------

Definition at line 101 of file FilterNumericVisitor.hpp.

References FilterNumericVisitor(), multiscaleArchitectureGraph, spatialEntity, and timePoint.

Referenced by operator()().

```
6.71.3.2 double multiscale::verification::FilterNumericVisitor::evaluate ( const
    PrimaryNumericMeasureAttributeType & primaryNumericMeasure ) const
    [inline, private]
```

Evaluate the given primary numeric measure considering the timePoint field.

#### Parameters

<i>primary-</i> <i>Numeric-</i> <i>Measure</i>	The given primary numeric measure
--	-----------------------------------

Definition at line 118 of file FilterNumericVisitor.hpp.

References multiscaleArchitectureGraph, and timePoint.

```
6.71.3.3 double multiscale::verification::FilterNumericVisitor::operator() ( const
    FilterNumericMeasureAttribute & filterNumericMeasure ) const [inline]
```

Overloading the "(") operator for the [FilterNumericMeasureAttribute](#) alternative.

#### Parameters

<i>filter-</i> <i>Numeric-</i> <i>Measure</i>	The filter numeric measure
---	----------------------------

Definition at line 38 of file FilterNumericVisitor.hpp.

References evaluate(), and multiscale::verification::FilterNumericMeasureAttribute-  
::filterNumericMeasure.

```
6.71.3.4 double multiscale::verification::FilterNumericVisitor::operator() ( const
    PrimaryNumericMeasureAttribute & primaryNumericMeasure ) const
    [inline]
```

Overloading the "(") operator for the [PrimaryNumericMeasureAttribute](#) alternative.

#### Parameters

<i>primary-</i> <i>Numeric-</i> <i>Measure</i>	The primary numeric measure
--	-----------------------------

Definition at line 46 of file FilterNumericVisitor.hpp.

References `evaluate()`, and `multiscale::verification::PrimaryNumericMeasureAttribute::primaryNumericMeasure`.

**6.71.3.5** `double multiscale::verification::FilterNumericVisitor::operator() ( const SpatialMeasureAttribute & spatialMeasure ) const [inline]`

Overloading the "`()`" operator for the [SpatialMeasureAttribute](#) alternative.

#### Parameters

<code><i>spatialMeasure</i></code>	The spatial measure
------------------------------------	---------------------

Definition at line 54 of file FilterNumericVisitor.hpp.

References `multiscale::verification::SpatialMeasureEvaluator::evaluate()`, `spatialEntity`, and `multiscale::verification::SpatialMeasureAttribute::spatialMeasureType`.

**6.71.3.6** `double multiscale::verification::FilterNumericVisitor::operator() ( const UnaryNumericFilterAttribute & unaryNumericFilter ) const [inline]`

Overloading the "`()`" operator for the [UnaryNumericFilterAttribute](#) alternative.

#### Parameters

<code><i>unaryNumericFilter</i></code>	The unary numeric filter
--	--------------------------

Definition at line 67 of file FilterNumericVisitor.hpp.

References `multiscale::verification::NumericEvaluator::evaluate()`, `evaluate()`, `multiscale::verification::UnaryNumericFilterAttribute::filterNumericMeasure`, `multiscale::verification::UnaryNumericFilterAttribute::unaryNumericMeasure`, and `multiscale::verification::UnaryNumericMeasureAttribute::unaryNumericMeasureType`.

**6.71.3.7** `double multiscale::verification::FilterNumericVisitor::operator() ( const BinaryNumericFilterAttribute & binaryNumericFilter ) const [inline]`

Overloading the "`()`" operator for the [BinaryNumericFilterAttribute](#) alternative.

#### Parameters

<code><i>binaryNumericFilter</i></code>	The binary numeric filter
---	---------------------------

Definition at line 82 of file FilterNumericVisitor.hpp.

References      multiscale::verification::BinaryNumericFilterAttribute::binaryNumericMeasure,    multiscale::verification::BinaryNumericMeasureAttribute::binaryNumericMeasureType,    multiscale::verification::NumericEvaluator::evaluate(),    evaluate(),    multiscale::verification::BinaryNumericFilterAttribute::firstFilterNumericMeasure,    and    multiscale::verification::BinaryNumericFilterAttribute::secondFilterNumericMeasure.

#### 6.71.4 Member Data Documentation

**6.71.4.1 const MultiscaleArchitectureGraph& multiscale::verification::FilterNumericVisitor::multiscaleArchitectureGraph [private]**

The considered multiscale architecture graph

Definition at line 25 of file FilterNumericVisitor.hpp.

Referenced by evaluate().

**6.71.4.2 const SpatialEntity& multiscale::verification::FilterNumericVisitor::spatialEntity [private]**

The considered spatial entity

Definition at line 23 of file FilterNumericVisitor.hpp.

Referenced by evaluate(), and operator()().

**6.71.4.3 TimePoint& multiscale::verification::FilterNumericVisitor::timePoint [private]**

The considered timepoint

Definition at line 21 of file FilterNumericVisitor.hpp.

Referenced by evaluate().

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

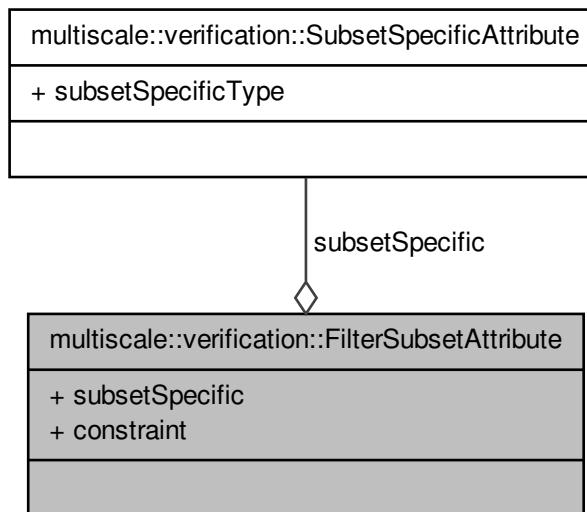
- FilterNumericVisitor.hpp

### 6.72 multiscale::verification::FilterSubsetAttribute Class Reference

Class for representing a filter subset attribute.

```
#include <FilterSubsetAttribute.hpp>
```

Collaboration diagram for multiscale::verification::FilterSubsetAttribute:



## Public Attributes

- `SubsetSpecificAttribute subsetSpecific`
- `ConstraintAttributeType constraint`

### 6.72.1 Detailed Description

Class for representing a filter subset attribute.

Definition at line 15 of file FilterSubsetAttribute.hpp.

### 6.72.2 Member Data Documentation

#### 6.72.2.1 ConstraintAttributeType multiscale::verification::FilterSubsetAttribute- ::constraint

The constraint

Definition at line 20 of file FilterSubsetAttribute.hpp.

Referenced by `multiscale::verification::SubsetVisitor::operator()()`.

### 6.72.2.2 **SubsetSpecificAttribute multiscale::verification::FilterSubsetAttribute- ::subsetSpecific**

The specific subset to consider

Definition at line 19 of file FilterSubsetAttribute.hpp.

Referenced by multiscale::verification::SubsetVisitor::operator()().

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

- FilterSubsetAttribute.hpp

## 6.73 multiscale::verification::FutureLogicPropertyAttribute Class - Reference

Class for representing a "future" logic property attribute.

```
#include <FutureLogicPropertyAttribute.hpp>
```

### Public Attributes

- unsigned long [startTimepoint](#)
- unsigned long [endTimepoint](#)
- [LogicPropertyAttributeType logicProperty](#)

### 6.73.1 Detailed Description

Class for representing a "future" logic property attribute.

Definition at line 14 of file FutureLogicPropertyAttribute.hpp.

### 6.73.2 Member Data Documentation

#### 6.73.2.1 unsigned long multiscale::verification::FutureLogicPropertyAttribute::end- Timepoint

The considered end timepoint

Definition at line 19 of file FutureLogicPropertyAttribute.hpp.

#### 6.73.2.2 [LogicPropertyAttributeType multiscale::verification::FutureLogic- PropertyAttribute::logicProperty](#)

The logic property following the "future" operator

Definition at line 20 of file FutureLogicPropertyAttribute.hpp.

Referenced by multiscale::verification::LogicPropertyVisitor::evaluateTemporalLogicPropertyWithStartAndEndTimepoints().

### 6.73.2.3 unsigned long multiscale::verification::FutureLogicPropertyAttribute- ::startTimepoint

The considered start timepoint

Definition at line 18 of file FutureLogicPropertyAttribute.hpp.

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

- FutureLogicPropertyAttribute.hpp

## 6.74 multiscale::Geometry2D Class Reference

Two-dimensional geometric operations.

```
#include <Geometry2D.hpp>
```

### Static Public Member Functions

- static double **angleBtwPoints** (const cv::Point2f &a, const cv::Point2f &b, const cv::Point2f &c)  
*Compute the angle ABC between three points A, B and C.*
- static double **angleOfLineWrtOxAxis** (const cv::Point2f &a, const cv::Point2f &b)  
*Get the angle of the line measured from the Ox axis in counterclockwise direction.*
- static bool **isAngleBetween** (double angle1, double angle2, double angle3)  
*Check if angle1 lies between angles 2 and 3.*
- static bool **isOppositeAngleBetween** (double angle1, double angle2, double angle3)  
*Check if the opposite of angle1, ((angle1 + 180) % 360), lies between angles 2 and 3.*
- static bool **isAngleBetweenNonReflex** (double angle1, double angle2, double angle3)  
*Check if angle1 lies between non reflex angle determined by angles 2 and 3.*
- static bool **isOppositeAngleBetweenNonReflex** (double angle1, double angle2, double angle3)  
*Check if the opposite of angle1 lies between non reflex angle determined by angles 2 and 3.*
- static double **oppositeAngle** (double angle)  
*Return the angle opposite to the given angle.*
- static bool **slopeOfLine** (const cv::Point2f &a, const cv::Point2f &b, double &slope)  
*Compute the slope of the line defined by points "a" and "b".*

- template<typename LhsCoordinateType , typename RhsCoordinateType >  
 static double **distanceBtwPoints** (const cv::Point\_< LhsCoordinateType > &a,  
 const cv::Point\_< RhsCoordinateType > &b)  
*Compute the distance between two points.*
- static double **distanceBtwPoints** (double x1, double y1, double x2, double y2)  
*Compute the distance between two points.*
- static double **distanceFromPointToLine** (const cv::Point2f &a, const cv::Point2f  
 &linePointB, const cv::Point2f &linePointC)  
*Compute the distance from a point "a" to a line specified by two points "B" and "C".*
- static cv::Point2f **middlePoint** (const cv::Point2f &a, const cv::Point2f &b)  
*Get the point in the middle of the segment determined by points "a" and "b".*
- template<typename CoordinateType >  
 static cv::Point2f **centroid** (const std::vector< cv::Point\_< CoordinateType >>  
 &points)  
*Compute the centroid of the provided collection of points.*
- static bool **isPointInsidePolygon** (const cv::Point2f &point, const std::vector< cv-  
 ::Point2f > &polygon)  
*Check if the given point lies inside the polygon.*
- static bool **isConvexPolygon** (const std::vector< cv::Point2f > &polygon)  
*Check if the provided polygon is convex.*
- template<typename CoordinateType >  
 static std::vector< cv::Point\_< CoordinateType > > **computeConvexHull** (const  
 std::vector< cv::Point\_< CoordinateType > > &polygon, bool arePointsIn-  
 ClockwiseOrder=false)  
*Compute the convex hull for the provided polygon.*
- static void **tangentsFromPointToPolygon** (const std::vector< cv::Point2f >  
 &convexPolygon, const cv::Point2f &referencePoint, cv::Point2f &leftMost-  
 TangentPoint, cv::Point2f &rightMostTangentPoint)  
*Compute the polygon points where the tangents from a reference point touch the given  
 polygon.*
- static void **orthogonalLineToAnotherLineEdgePoints** (const cv::Point &a1, const  
 cv::Point &b1, cv::Point &a2, cv::Point &b2, int nrOfRows, int nrOfCols)  
*Find the points which are on the edge and on the line orthogonal to the line defined by  
 2 given points.*
- static double **sideOfLine** (const cv::Point2f &point, const cv::Point2f &lineStart-  
 PointA, const cv::Point2f &lineEndPointB)  
*Compute on which side of the line (A, B) the given point P lies.*
- static bool **isToTheRightOfLine** (const cv::Point2f &point, const cv::Point2f &line-  
 StartPointA, const cv::Point2f &lineEndPointB)  
*Check if the point P lies to the right of line (A, B)*
- static bool **isToTheLeftOfLine** (const cv::Point2f &point, const cv::Point2f &line-  
 StartPointA, const cv::Point2f &lineEndPointB)  
*Check if the point P lies to the left of line (A, B)*
- static bool **areOnTheSameSideOfLine** (const cv::Point2f &p1, const cv::Point2f  
 &p2, const cv::Point2f &a, const cv::Point2f &b)  
*Check if p1 and p2 are on the same side of the line determined by points a and b.*

- static void `lineEquationDeterminedByPoints` (const cv::Point2f &p, const cv::Point2f &q, double &a, double &b, double &c)
 

*Get the values of "a", "b" and "c" of the line equation  $ax + by + c = 0$ .*
- static bool `areIdenticalLines` (double a1, double b1, double c1, double a2, double b2, double c2)
 

*Check if two lines are identical.*
- static bool `areIdenticalLines` (const cv::Point2f &a1, const cv::Point2f &b1, const cv::Point2f &a2, const cv::Point2f &b2)
 

*Check if two lines are identical.*
- static bool `lineIntersection` (const cv::Point2f &a1, const cv::Point2f &b1, const cv::Point2f &a2, const cv::Point2f &b2, cv::Point2f &intersection)
 

*Determine the intersection point of two lines, if this point exists.*
- static bool `lineIntersection` (const cv::Point &a1, const cv::Point &b1, const cv::Point &a2, const cv::Point &b2, cv::Point &intersection)
 

*Determine the intersection point of two lines, if this point exists.*
- static bool `lineIntersection` (double a1, double b1, double c1, double a2, double b2, double c2, cv::Point2f &intersection)
 

*Determine the intersection point of two lines, if this point exists.*
- static bool `lineSegmentIntersection` (const cv::Point &a1, const cv::Point &b1, const cv::Point &a2, const cv::Point &b2, cv::Point &intersection)
 

*Determine the intersection point of two line segments, if this point exists.*
- static bool `lineCircleIntersection` (cv::Point2f a, cv::Point2f b, const cv::Point2f &circleOrigin, double radius, std::vector<cv::Point2f> &intersectionPoints)
 

*Determine if a line and a circle intersect and return the intersection points if they exist.*
- static bool `lineSegmentCircleIntersection` (const cv::Point2f &a, const cv::Point2f &b, const cv::Point2f &circleOrigin, double radius, std::vector<cv::Point2f> &intersectionPoints)
 

*Determine if a line segment and a circle intersect and return the intersection points if they exist.*
- static std::vector<cv::Point2f> `findPointsOnEdge` (const std::vector<cv::Point2f> &points, unsigned int nrOfRows, unsigned int nrOfCols)
 

*Find the subset of points from the given set of points which lie on the edge.*
- static unsigned int `minimumDistancePointIndex` (const std::vector<cv::Point> &points, const cv::Point2f &origin)
 

*Get the index of the point which is the closest to the origin.*
- template<typename PointCoordinateType>
 static unsigned int `minimumDistancePointIndex` (const std::vector<cv::Point<PointCoordinateType>> &points, const cv::Point2f &origin)
 

*Get the index of the point which is the closest to the origin.*
- static bool `isPointOnLineSegment` (const cv::Point2f &point, const cv::Point2f &lineSegmentStart, const cv::Point2f &lineSegmentEnd)
 

*Check if one point lies between two other points.*
- static bool `areEqualPoints` (const cv::Point2f &point1, const cv::Point2f &point2)
 

*Check if points point1 and point2 are equal or not.*
- static bool `areCollinear` (const cv::Point2f &point1, const cv::Point2f &point2, const cv::Point2f &point3)

*Check if the three points are collinear.*

- static double `areaOfTriangle` (const cv::Point2f &a, const cv::Point2f &b, const cv::Point2f &c)

*Compute the area of a triangle defined by three points.*

- template<typename SourceCoordinateType , typename DestinationCoordinateType >  
static std::vector< cv::Point\_ < DestinationCoordinateType > > `convertPoints`(const std::vector< cv::Point\_ < SourceCoordinateType > > &pointsCollection)

*Convert the coordinates from integers to floating point for the given points collection.*

## Static Public Attributes

- static const double `PI` = 3.14159265358979323846264338327950288419716939937510
- static const int `MATRIX_START_INDEX` = 0

## Static Private Member Functions

- static bool `isPointOnEdge` (const cv::Point2f &p, int nrOfRows, int nrOfCols)

*Check if the given point is on the edge.*

- template<typename T , typename U >  
static bool `isBetweenCoordinates` (T c, U c1, U c2)

*Check if the coordinate c lies between c1 and c2.*

- static void `translate` (cv::Point2f &point, const cv::Point2f &translation)

*Translate a point by the given values.*

- static void `inverseTranslate` (cv::Point2f &point, const cv::Point2f &translation)

*Inverse translate a point by the given values.*

- static void `lineCircleTwoIntersectionPoints` (const cv::Point2f &circleOrigin, double A, double B, double C, double delta, std::vector< cv::Point2f > &intersectionPoints)

*Treat the case when the line and circle intersect in two points.*

- static void `lineCircleOneIntersectionPoint` (const cv::Point2f &circleOrigin, double A, double B, double C, double delta, std::vector< cv::Point2f > &intersectionPoints)

*Treat the case when the line and circle intersect in one point.*

### 6.74.1 Detailed Description

Two-dimensional geometric operations.

Definition at line 13 of file Geometry2D.hpp.

## 6.74.2 Member Function Documentation

6.74.2.1 **double Geometry2D::angleBtwPoints ( const cv::Point2f & a, const cv::Point2f & b, const cv::Point2f & c ) [static]**

Compute the angle ABC between three points A, B and C.

Compute the angle between the lines determined by points (A, B) and (B, C)

### Parameters

<i>a</i>	cv::Point2f a
<i>b</i>	cv::Point2f b
<i>c</i>	cv::Point2f c

Definition at line 11 of file Geometry2D.cpp.

References PI.

Referenced by multiscale::analysis::Detector::computePolygonAngle().

6.74.2.2 **double Geometry2D::angleOfLineWrtOxAxis ( const cv::Point2f & a, const cv::Point2f & b ) [static]**

Get the angle of the line measured from the Ox axis in counterclockwise direction.

The line is specified by points "a" and "b". The value of the angle is expressed in degrees.

### Parameters

<i>a</i>	Point a
<i>b</i>	Point b

Definition at line 23 of file Geometry2D.cpp.

References PI.

Referenced by multiscale::MinEnclosingTriangleFinder::intersects(), multiscale::MinEnclosingTriangleFinder::intersectsAbove(), and multiscale::MinEnclosingTriangleFinder::intersectsBelow().

6.74.2.3 **double Geometry2D::areaOfTriangle ( const cv::Point2f & a, const cv::Point2f & b, const cv::Point2f & c ) [static]**

Compute the area of a triangle defined by three points.

The area is computed using the determinant method. An example is presented at <http://demonstrations.wolfram.com/TheAreaOfATriangle-UsingADeterminant/> (Last access: 10.07.2013)

**Parameters**

<i>a</i>	Point a
<i>b</i>	Point b
<i>c</i>	Point c

Definition at line 439 of file Geometry2D.cpp.

Referenced by multiscale::MinEnclosingTriangleFinder::returnMinEnclosingTriangle(), and multiscale::MinEnclosingTriangleFinder::updateMinEnclosingTriangle().

**6.74.2.4 bool Geometry2D::areCollinear ( const cv::Point2f & *point1*, const cv::Point2f & *point2*, const cv::Point2f & *point3* ) [static]**

Check if the three points are collinear.

**Parameters**

<i>point1</i>	The first point
<i>point2</i>	The second point
<i>point3</i>	The third point

Definition at line 432 of file Geometry2D.cpp.

References multiscale::Numeric::almostEqual().

Referenced by multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::isPointCRightMostTangentPoint().

**6.74.2.5 bool Geometry2D::areEqualPoints ( const cv::Point2f & *point1*, const cv::Point2f & *point2* ) [static]**

Check if points point1 and point2 are equal or not.

**Parameters**

<i>point1</i>	One point
<i>point2</i>	The other point

Definition at line 425 of file Geometry2D.cpp.

References multiscale::Numeric::almostEqual().

Referenced by multiscale::MinEnclosingTriangleFinder::isValidMinimalTriangle(), and lineEquationDeterminedByPoints().

**6.74.2.6 bool Geometry2D::areIdenticalLines ( double *a1*, double *b1*, double *c1*, double *a2*, double *b2*, double *c2* ) [static]**

Check if two lines are identical.

Lines are be specified in the following form:  $A_1x + B_1x = C_1$   $A_2x + B_2x = C_2$

If  $(A_1/A_2) == (B_1/B_2) == (C_1/C_2)$ , then the lines are identical else they are not

#### Parameters

<i>a1</i>	A1
<i>b1</i>	B1
<i>c1</i>	C1
<i>a2</i>	A2
<i>b2</i>	B2
<i>c2</i>	C2

Definition at line 242 of file Geometry2D.cpp.

References multiscale::Numeric::almostEqual().

Referenced by multiscale::MinEnclosingTriangleFinder::areIdenticalLines().

**6.74.2.7 bool Geometry2D::areIdenticalLines ( const cv::Point2f & *a1*, const cv::Point2f & *b1*, const cv::Point2f & *a2*, const cv::Point2f & *b2* ) [static]**

Check if two lines are identical.

The lines are specified by a pair of points each. If they are identical, then the function returns true, else it returns false.

Lines can be specified in the following form:  $A_1x + B_1x = C_1$   $A_2x + B_2x = C_2$

If  $(A_1/A_2) == (B_1/B_2) == (C_1/C_2)$ , then the lines are identical else they are not

#### Parameters

<i>a1</i>	First point for determining the first line
<i>b1</i>	Second point for determining the first line
<i>a2</i>	First point for determining the second line
<i>b2</i>	Second point for determining the second line

Definition at line 257 of file Geometry2D.cpp.

References multiscale::Numeric::almostEqual().

**6.74.2.8 bool Geometry2D::areOnTheSameSideOfLine ( const cv::Point2f & *p1*, const cv::Point2f & *p2*, const cv::Point2f & *a*, const cv::Point2f & *b* ) [static]**

Check if *p1* and *p2* are on the same side of the line determined by points *a* and *b*.

#### Parameters

<i>p1</i>	Point <i>p1</i>
<i>p2</i>	Point <i>p2</i>
<i>a</i>	First point for determining line
<i>b</i>	Second point for determining line

Definition at line 220 of file Geometry2D.cpp.

References lineEquationDeterminedByPoints(), and multiscale::Numeric::sign().

Referenced by multiscale::MinEnclosingTriangleFinder::findVertexCOnSideB(), and multiscale::MinEnclosingTriangleFinder::gamma().

```
6.74.2.9 template<typename CoordinateType> static cv::Point2f
multiscale::Geometry2D::centroid( const std::vector<cv::Point_<
CoordinateType>>& points ) [inline, static]
```

Compute the centroid of the provided collection of points.

#### Parameters

<i>points</i>	The considered collection of points
---------------	-------------------------------------

Definition at line 155 of file Geometry2D.hpp.

Referenced by multiscale::analysis::Cluster::updateCentrePoint().

```
6.74.2.10 template<typename CoordinateType> static std::vector<cv::Point_-<
CoordinateType>> multiscale::Geometry2D::computeConvexHull(
( const std::vector<cv::Point_-<CoordinateType>>& polygon, bool
arePointsInClockwiseOrder = false ) [inline, static]
```

Compute the convex hull for the provided polygon.

#### Parameters

<i>polygon</i>	The provided polygon
<i>arePointsIn- Clockwise- Order</i>	Flag indicating if the points in the convex hull are sorted in clockwise order

Definition at line 195 of file Geometry2D.hpp.

Referenced by multiscale::analysis::CircularityMeasure< PointType >::compute(), multiscale::analysis::Detector::computePolygonAngle(), multiscale::analysis::Cluster::getEntitiesConvexHull(), multiscale::MinEnclosingTriangleFinder::initialiseConvexPolygon(), multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::initialisePolygon(), multiscale::analysis::Region::isTriangularMeasure(), and multiscaletest::MinEnclosingTriangleFinderTest::RunTest().

```
6.74.2.11 template<typename SourceCoordinateType, typename DestinationCoordinateType
> static std::vector<cv::Point_-<DestinationCoordinateType>>
multiscale::Geometry2D::convertPoints( const std::vector<cv::Point_-<
SourceCoordinateType>>& pointsCollection ) [inline, static]
```

Convert the coordinates from integers to floating point for the given points collection.

**Parameters**

<i>points-Collection</i>	The given collection of points
--------------------------	--------------------------------

Definition at line 553 of file Geometry2D.hpp.

```
6.74.2.12 template<typename LhsCoordinateType , typename RhsCoordinateType > static
double multiscale::Geometry2D::distanceBtwPoints ( const cv::Point_<
LhsCoordinateType > & a, const cv::Point_< RhsCoordinateType > & b )
[inline, static]
```

Compute the distance between two points.

Compute the Euclidean distance between two points

**Parameters**

<i>a</i>	Point a
<i>b</i>	Point b

Definition at line 99 of file Geometry2D.hpp.

Referenced by multiscale::analysis::RegionDetector::computeAverageCentroid-Distance(), multiscale::analysis::Silhouette::computeAverageDissimilarityBtwEntityAnd-Cluster(), multiscale::analysis::Silhouette::computeAverageDissimilarityWithinCluster(), multiscale::analysis::Detector::computeDistanceFromOrigin(), EuclideanDataPoint-::distanceTo(), multiscale::analysis::Entity::distanceTo(), isPointOnLineSegment(), and minimumDistancePointIndex().

```
6.74.2.13 double Geometry2D::distanceBtwPoints ( double x1, double y1, double x2,
double y2 ) [static]
```

Compute the distance between two points.

Compute the Euclidean distance between two points

**Parameters**

<i>x1</i>	The x-coordinate of the first point
<i>y1</i>	The y-coordinate of the first point
<i>x2</i>	The x-coordinate of the second point
<i>y2</i>	The y-coordinate of the second point

Definition at line 89 of file Geometry2D.cpp.

```
6.74.2.14 double Geometry2D::distanceFromPointToLine ( const cv::Point2f & a, const
cv::Point2f & linePointB, const cv::Point2f & linePointC ) [static]
```

Compute the distance from a point "a" to a line specified by two points "B" and "C".

Formula used:

$$distance = \frac{|(x_c - x_b)(y_b - y_a) - (x_b - x_a)(y_c - y_b)|}{\sqrt{(x_c - x_b)^2 + (y_c - y_b)^2}}$$

Reference: <http://mathworld.wolfram.com/cv::Point-Line-Distance2-Dimensional.html>

#### Parameters

<i>a</i>	The point from which the distance is measured
<i>linePointB</i>	One of the points determining the line
<i>linePointC</i>	One of the points determining the line

Definition at line 101 of file Geometry2D.cpp.

References multiscale::Numeric::division().

Referenced by multiscale::MinEnclosingTriangleFinder::height().

```
6.74.2.15 std::vector< cv::Point2f > Geometry2D::findPointsOnEdge ( const
std::vector< cv::Point2f > & points, unsigned int nrOfRows, unsigned int nrOfCols )
[static]
```

Find the subset of points from the given set of points which lie on the edge.

A point "p" is considered to be on the edge if: ((p.x == 1) && (p.y > 1) && (p.y < nrOfCols)) OR ((p.x == nrOfRows) && (p.y > 1) && (p.y < nrOfCols)) OR ((p.y == 1) && (p.x > 1) && (p.x < nrOfRows)) OR ((p.y == nrOfCols) && (p.x > 1) && (p.x < nrOfRows))

#### Parameters

<i>points</i>	The set of points
<i>nrOfRows</i>	The number of rows
<i>nrOfCols</i>	The number of columns

Definition at line 402 of file Geometry2D.cpp.

References isPointOnEdge().

```
6.74.2.16 void Geometry2D::inverseTranslate ( cv::Point2f & point, const cv::Point2f &
translation ) [static, private]
```

Inverse translate a point by the given values.

#### Parameters

<i>point</i>	The point
<i>translation</i>	Translation values

Definition at line 469 of file Geometry2D.cpp.

Referenced by lineCircleOneIntersectionPoint(), and lineCircleTwoIntersectionPoints().

**6.74.2.17 bool Geometry2D::isAngleBetween ( double *angle1*, double *angle2*, double *angle3* ) [static]**

Check if angle1 lies between angles 2 and 3.

**Parameters**

<i>angle1</i>	The angle which lies between angle2 and angle3 or not
<i>angle2</i>	One of the boundary angles
<i>angle3</i>	The other boundary angle

Definition at line 33 of file Geometry2D.cpp.

Referenced by isAngleBetweenNonReflex(), and isOppositeAngleBetween().

**6.74.2.18 bool Geometry2D::isAngleBetweenNonReflex ( double *angle1*, double *angle2*, double *angle3* ) [static]**

Check if angle1 lies between non reflex angle determined by angles 2 and 3.

**Parameters**

<i>angle1</i>	The angle which lies between angle2 and angle3 or not
<i>angle2</i>	One of the boundary angles
<i>angle3</i>	The other boundary angle

Definition at line 47 of file Geometry2D.cpp.

References isAngleBetween(), and multiscale::Numeric::lessOrEqual().

Referenced by multiscale::MinEnclosingTriangleFinder::isFlushAngleBetweenPredecessorAndSuccessor(), multiscale::MinEnclosingTriangleFinder::isGammaAngleBetween(), and isOppositeAngleBetweenNonReflex().

**6.74.2.19 template<typename T , typename U > bool Geometry2D::is-BetweenCoordinates ( T *c*, U *c1*, U *c2* ) [static, private]**

Check if the coordinate c lies between c1 and c2.

**Parameters**

<i>c</i>	Coordinate c
<i>c1</i>	Coordinate c1
<i>c2</i>	Coordinate c2

Definition at line 460 of file Geometry2D.cpp.

---

**6.74.2.20 bool Geometry2D::isConvexPolygon ( const std::vector< cv::Point2f > & polygon ) [static]**

Check if the provided polygon is convex.

**Parameters**

<i>polygon</i>	The provided polygon
----------------	----------------------

Definition at line 129 of file Geometry2D.cpp.

Referenced by multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::initialisePolygon().

**6.74.2.21 bool Geometry2D::isOppositeAngleBetween ( double angle1, double angle2, double angle3 ) [static]**

Check if the opposite of angle1, ((angle1 + 180) % 360), lies between angles 2 and 3.

**Parameters**

<i>angle1</i>	The angle for which the opposite angle lies between angle2 and angle3 or not
<i>angle2</i>	One of the boundary angles
<i>angle3</i>	The other boundary angle

Definition at line 41 of file Geometry2D.cpp.

References isAngleBetween(), and oppositeAngle().

**6.74.2.22 bool Geometry2D::isOppositeAngleBetweenNonReflex ( double angle1, double angle2, double angle3 ) [static]**

Check if the opposite of angle1 lies between non reflex angle determined by angles 2 and 3.

Check if the opposite of angle1, ((angle1 + 180) % 360), lies between non reflex angle determined by angles 2 and 3

**Parameters**

<i>angle1</i>	The angle which lies between angle2 and angle3 or not
<i>angle2</i>	One of the boundary angles
<i>angle3</i>	The other boundary angle

Definition at line 65 of file Geometry2D.cpp.

References isAngleBetweenNonReflex(), and oppositeAngle().

Referenced by multiscale::MinEnclosingTriangleFinder::isFlushAngleBetweenPredecessorAndSuccessor().

**6.74.2.23 bool Geometry2D::isPointInsidePolygon ( const cv::Point2f & *point*, const std::vector< cv::Point2f > & *polygon* ) [static]**

Check if the given point lies inside the polygon.

**Parameters**

<i>point</i>	The given point
<i>polygon</i>	The given polygon

Definition at line 121 of file Geometry2D.cpp.

Referenced by multiscale::analysis::Detector::computeDistanceFromOrigin(), multiscale::analysis::Detector::computePolygonAngle(), and multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::computeTangentsPointsWithInitialisedAlgorithmVariables().

**6.74.2.24 bool Geometry2D::isPointOnEdge ( const cv::Point2f & *p*, int *nrOfRows*, int *nrOfCols* ) [static, private]**

Check if the given point is on the edge.

A point "p" is considered to be on the edge if: ((p.x == 0) && (p.y >= 1) && (p.y <= nrOfCols)) OR ((p.x == nrOfRows) && (p.y >= 1) && (p.y <= nrOfCols)) OR ((p.y == 0) && (p.x >= 1) && (p.x <= nrOfRows)) OR ((p.y == nrOfCols) && (p.x >= 1) && (p.x <= nrOfRows))

**Parameters**

<i>p</i>	The point p
<i>nrOfRows</i>	The number of rows
<i>nrOfCols</i>	The number of columns

Definition at line 450 of file Geometry2D.cpp.

References MATRIX\_START\_INDEX.

Referenced by findPointsOnEdge(), and orthogonalLineToAnotherLineEdgePoints().

**6.74.2.25 bool Geometry2D::isPointOnLineSegment ( const cv::Point2f & *point*, const cv::Point2f & *lineSegmentStart*, const cv::Point2f & *lineSegmentEnd* ) [static]**

Check if one point lies between two other points.

**Parameters**

<i>point</i>	Point lying possibly outside the line segment
<i>line-Segment-Start</i>	First point determining the line segment

<i>line-Segment-End</i>	Second point determining the line segment
-------------------------	---

Definition at line 416 of file Geometry2D.cpp.

References multiscale::Numeric::almostEqual(), and distanceBtwPoints().

Referenced by multiscaletest::MinEnclosingTriangleFinderTest::IsOneEdgeFlush(), multiscaletest::MinEnclosingTriangleFinderTest::IsTriangleTouchingPolygon(), and multiscale::MinEnclosingTriangleFinder::isValidMinimalTriangle().

#### 6.74.2.26 bool Geometry2D::isToTheLeftOfLine ( const cv::Point2f & *point*, const cv::Point2f & *lineStartPointA*, const cv::Point2f & *lineEndPointB* ) [static]

Check if the point P lies to the left of line (A, B)

##### Parameters

<i>point</i>	The considered point P
<i>lineStart-PointA</i>	The line start point A
<i>lineEnd-PointB</i>	The line end point B

Definition at line 213 of file Geometry2D.cpp.

References sideOfLine().

Referenced by multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::isFirstPolygonPointTheRightMostTangentPoint(), multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::isPointAAbovePointC(), multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::isPointCRightMostTangentPoint(), and multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::updateEdgeAUFlag().

#### 6.74.2.27 bool Geometry2D::isToTheRightOfLine ( const cv::Point2f & *point*, const cv::Point2f & *lineStartPointA*, const cv::Point2f & *lineEndPointB* ) [static]

Check if the point P lies to the right of line (A, B)

##### Parameters

<i>point</i>	The considered point P
<i>lineStart-PointA</i>	The line start point A
<i>lineEnd-PointB</i>	The line end point B

Definition at line 206 of file Geometry2D.cpp.

References sideOfLine().

Referenced by multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::isFirstPolygonPointTheLeftMostTangentPoint(), multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::isPointABelowPointC(), multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::isPointCLeftMostTangentPoint(), multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::updateEdgeADownFlag(), and multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::updateEdgeCFlag().

```
6.74.2.28 bool Geometry2D::lineCircleIntersection ( cv::Point2f a, cv::Point2f b,
const cv::Point2f & circleOrigin, double radius, std::vector<cv::Point2f> &
intersectionPoints ) [static]
```

Determine if a line and a circle intersect and return the intersection points if they exist.

We translate all the points such that the circle origin coincides with the origin of the coordinate system. When returning the results, the intersection points are inverse translated.

#### Parameters

<i>a</i>	First point for determining the line
<i>b</i>	Second point for determining the line
<i>circleOrigin</i>	Origin of the circle
<i>radius</i>	Radius of the circle
<i>intersectionPoints</i>	The intersection points between the circle and the line

< Two intersection points

< One intersection point

Definition at line 353 of file Geometry2D.cpp.

References multiscale::Numeric::almostEqual(), lineCircleOneIntersectionPoint(), lineCircleTwoIntersectionPoints(), and translate().

Referenced by lineSegmentCircleIntersection().

```
6.74.2.29 void Geometry2D::lineCircleOneIntersectionPoint ( const cv::Point2f &
circleOrigin, double A, double B, double C, double delta, std::vector<cv::Point2f>
& intersectionPoints ) [static, private]
```

Treat the case when the line and circle intersect in one point.

#### Parameters

<i>circleOrigin</i>	Origin of the circle
<i>A</i>	$y_2 - y_1$

<i>B</i>	$x_1 - x_2$
<i>C</i>	$A*x_1 + B*y_1$
<i>delta</i>	$(4 * B^2 * C^2) - (4 * (A^2 + B^2) * (C^2 - (R^2 * A^2)))$
<i>intersection-Points</i>	Intersection points

Definition at line 493 of file Geometry2D.cpp.

References multiscale::Numeric::division(), and inverseTranslate().

Referenced by lineCircleIntersection().

```
6.74.2.30 void Geometry2D::lineCircleTwoIntersectionPoints ( const cv::Point2f &
    circleOrigin, double A, double B, double C, double delta, std::vector<cv::Point2f>
    & intersectionPoints ) [static, private]
```

Treat the case when the line and circle intersect in two points.

#### Parameters

<i>circleOrigin</i>	Origin of the circle
<i>A</i>	$y_2 - y_1$
<i>B</i>	$x_1 - x_2$
<i>C</i>	$A*x_1 + B*y_1$
<i>delta</i>	$(4 * B^2 * C^2) - (4 * (A^2 + B^2) * (C^2 - (R^2 * A^2)))$
<i>intersection-Points</i>	Intersection points

Definition at line 474 of file Geometry2D.cpp.

References multiscale::Numeric::division(), and inverseTranslate().

Referenced by lineCircleIntersection().

```
6.74.2.31 void Geometry2D::lineEquationDeterminedByPoints ( const cv::Point2f & p,
    const cv::Point2f & q, double & a, double & b, double & c ) [static]
```

Get the values of "a", "b" and "c" of the line equation  $ax + by + c = 0$ .

Get the values of "a", "b" and "c" of the line equation  $ax + by + c = 0$  knowing that point "p" and "q" are on the line

$$a = q.y - p.y \quad b = p.x - q.x \quad c = -(p.x * a) - (p.y * b)$$

#### Parameters

<i>p</i>	Point p
<i>q</i>	Point q
<i>a</i>	Parameter "a" from the line equation
<i>b</i>	Parameter "b" from the line equation
<i>c</i>	Parameter "c" from the line equation

Definition at line 233 of file Geometry2D.cpp.

References areEqualPoints().

Referenced by areOnTheSameSideOfLine(), and multiscale::MinEnclosingTriangleFinder::lineEquationParameters().

```
6.74.2.32 bool Geometry2D::lineIntersection ( const cv::Point2f & a1, const cv::Point2f &
                                             b1, const cv::Point2f & a2, const cv::Point2f & b2, cv::Point2f & intersection )
[static]
```

Determine the intersection point of two lines, if this point exists.

Two lines intersect if they are not parallel (Parallel lines intersect at +/- infinity, but we do not consider this case here).

The lines are specified by a pair of points each. If they intersect, then the function returns true, else it returns false.

Lines can be specified in the following form:  $A_1x + B_1x = C_1$   $A_2x + B_2x = C_2$

If  $\det (= A_1xB_2 - A_2xB_1) == 0$ , then lines are parallel else they intersect

If they intersect, then let us denote the intersection point with  $P(x, y)$  where:  $x = (C_1xB_2 - C_2xB_1) / (\det)$   $y = (C_2xA_1 - C_1xA_2) / (\det)$

#### Parameters

<i>a1</i>	First point for determining the first line
<i>b1</i>	Second point for determining the first line
<i>a2</i>	First point for determining the second line
<i>b2</i>	Second point for determining the second line
<i>intersection</i>	The intersection point, if this point exists

Definition at line 303 of file Geometry2D.cpp.

References multiscale::Numeric::almostEqual().

Referenced by multiscale::MinEnclosingTriangleFinder::areIntersectingLines(), multiscale::MinEnclosingTriangleFinder::isLocalMinimalTriangle(), lineSegmentIntersection(), and multiscale::MinEnclosingTriangleFinder::middlePointOfSideB().

```
6.74.2.33 bool Geometry2D::lineIntersection ( const cv::Point & a1, const cv::Point & b1,
                                             const cv::Point & a2, const cv::Point & b2, cv::Point & intersection ) [static]
```

Determine the intersection point of two lines, if this point exists.

Two lines intersect if they are not parallel (Parallel lines intersect at +/- infinity, but we do not consider this case here).

The lines are specified by a pair of points each. If they intersect, then the function returns true, else it returns false.

Lines can be specified in the following form:  $A_1x + B_1x = C_1$   $A_2x + B_2x = C_2$

If  $\det (= A_1x_2 - A_2x_1) == 0$ , then lines are parallel else they intersect

If they intersect, then let us denote the intersection point with  $P(x, y)$  where:  $x = (C_1x_2 - C_2x_1) / (\det)$   $y = (C_2x_1 - C_1x_2) / (\det)$

#### Parameters

<i>a1</i>	First point for determining the first line
<i>b1</i>	Second point for determining the first line
<i>a2</i>	First point for determining the second line
<i>b2</i>	Second point for determining the second line
<i>intersection</i>	The intersection point, if this point exists

Definition at line 281 of file Geometry2D.cpp.

References multiscale::Numeric::almostEqual().

**6.74.2.34 bool Geometry2D::lineIntersection ( double *a1*, double *b1*, double *c1*, double *a2*, double *b2*, double *c2*, cv::Point2f & *intersection* ) [static]**

Determine the intersection point of two lines, if this point exists.

Two lines intersect if they are not parallel (Parallel lines intersect at +/- infinity, but we do not consider this case here).

The lines are specified in the following form:  $A_1x + B_1x = C_1$   $A_2x + B_2x = C_2$

If  $\det (= A_1x_2 - A_2x_1) == 0$ , then lines are parallel else they intersect

If they intersect, then let us denote the intersection point with  $P(x, y)$  where:  $x = (C_1x_2 - C_2x_1) / (\det)$   $y = (C_2x_1 - C_1x_2) / (\det)$

#### Parameters

<i>a1</i>	$A_1$
<i>b1</i>	$B_1$
<i>c1</i>	$C_1$
<i>a2</i>	$A_2$
<i>b2</i>	$B_2$
<i>c2</i>	$C_2$
<i>intersection</i>	The intersection point, if this point exists

Definition at line 325 of file Geometry2D.cpp.

References multiscale::Numeric::almostEqual().

**6.74.2.35 bool Geometry2D::lineSegmentCircleIntersection ( const cv::Point2f & *a*, const cv::Point2f & *b*, const cv::Point2f & *circleOrigin*, double *radius*, std::vector<cv::Point2f > & *intersectionPoints* ) [static]**

Determine if a line segment and a circle intersect and return the intersection points if they exist.

We translate all the points such that the circle origin coincides with the origin of the coordinate system. When returning the results, the intersection points are inverse translated.

#### Parameters

<i>a</i>	First point for determining the line
<i>b</i>	Second point for determining the line
<i>circleOrigin</i>	Origin of the circle
<i>radius</i>	Radius of the circle
<i>intersection-Points</i>	The intersection points between the circle and the line

Definition at line 382 of file Geometry2D.cpp.

References lineCircleIntersection().

**6.74.2.36 bool Geometry2D::lineSegmentIntersection ( const cv::Point & *a1*, const cv::Point & *b1*, const cv::Point & *a2*, const cv::Point & *b2*, cv::Point & *intersection* ) [static]**

Determine the intersection point of two line segments, if this point exists.

Find the intersection point of the lines, if this point exists. Let us assume that this point exists and let us denote it with P(x, y). Then, in order for the point to be the intersection of the segments and not of the lines, we have to verify the following conditions: 1.  $\min(a1.x, b1.x) \leq x \leq \max(a1.x, b1.x)$  -- x coordinate is valid for first line segment 2.  $\min(a2.x, b2.x) \leq x \leq \max(a2.x, b2.x)$  -- x coordinate is valid for second line segment 3.  $\min(a1.y, b1.y) \leq y \leq \max(a1.y, b1.y)$  -- y coordinate is valid for first line segment 4.  $\min(a2.y, b2.y) \leq y \leq \max(a2.y, b2.y)$  -- y coordinate is valid for second line segment

#### Parameters

<i>a1</i>	First point for determining the first line
<i>b1</i>	Second point for determining the first line
<i>a2</i>	First point for determining the second line
<i>b2</i>	Second point for determining the second line
<i>intersection</i>	The intersection point, if this point exists

Definition at line 339 of file Geometry2D.cpp.

References lineIntersection().

**6.74.2.37 cv::Point2f Geometry2D::middlePoint ( const cv::Point2f & *a*, const cv::Point2f & *b* ) [static]**

Get the point in the middle of the segment determined by points "a" and "b".

**Parameters**

<i>a</i>	Point a
<i>b</i>	Point b

Definition at line 114 of file Geometry2D.cpp.

Referenced by multiscaletest::MinEnclosingTriangleFinderTest::IsTriangleTouchingPolygon(), multiscale::MinEnclosingTriangleFinder::isValidMinimalTriangle(), and multiscale::MinEnclosingTriangleFinder::middlePointOfSideB().

**6.74.2.38 static unsigned int multiscale::Geometry2D::minimumDistancePointIndex ( const std::vector< cv::Point > & *points*, const cv::Point2f & *origin* ) [static]**

Get the index of the point which is the closest to the origin.

Get the index of the point P from the given set of points, such that for any point A from the set of points  $\text{dist}(A, \text{origin}) \geq \text{dist}(P, \text{origin})$ .

**Parameters**

<i>points</i>	The set of points
<i>origin</i>	The origin

Referenced by multiscale::analysis::Detector::computeDistanceFromOrigin().

**6.74.2.39 template<typename PointCoordinateType> static unsigned int multiscale::Geometry2D::minimumDistancePointIndex ( const std::vector< cv::Point< PointCoordinateType >> & *points*, const cv::Point2f & *origin* ) [inline, static]**

Get the index of the point which is the closest to the origin.

Get the index of the point P from the given set of points, such that for any point A from the set of points  $\text{dist}(A, \text{origin}) \geq \text{dist}(P, \text{origin})$ .

**Parameters**

<i>points</i>	The set of points
<i>origin</i>	The origin

Definition at line 489 of file Geometry2D.hpp.

References distanceBtwPoints().

**6.74.2.40 double Geometry2D::oppositeAngle ( double *angle* ) [static]**

Return the angle opposite to the given angle.

if (*angle* < 180) then return (*angle* + 180); else return (*angle* - 180); endif

**Parameters**

<i>angle</i>	Angle
--------------	-------

Definition at line 71 of file Geometry2D.cpp.

Referenced by multiscale::MinEnclosingTriangleFinder::isFlushAngleBetweenPredecessorAndSuccessor(), isOppositeAngleBetween(), and isOppositeAngleBetweenNonReflex().

**6.74.2.41 void Geometry2D::orthogonalLineToAnotherLineEdgePoints ( const cv::Point & *a1*, const cv::Point & *b1*, cv::Point & *a2*, cv::Point & *b2*, int *nrOfRows*, int *nrOfCols* ) [static]**

Find the points which are on the edge and on the line orthogonal to the line defined by 2 given points.

**Parameters**

<i>a1</i>	First point for determining the first line
<i>b1</i>	Second point for determining the first line
<i>a2</i>	First point for determining the second line
<i>b2</i>	Second point for determining the second line
<i>nrOfRows</i>	Maximum number of rows in the considered matrix
<i>nrOfCols</i>	Maximum number of columns in the considered matrix

Definition at line 149 of file Geometry2D.cpp.

References multiscale::Numeric::division(), and isPointOnEdge().

**6.74.2.42 double Geometry2D::sideOfLine ( const cv::Point2f & *point*, const cv::Point2f & *lineStartPointA*, const cv::Point2f & *lineEndPointB* ) [static]**

Compute on which side of the line (A, B) the given point P lies.

The function returns: <0, if P lies to the left of (A, B) 0, if P lies on the line (A, B) >0, if P lies to the right of (A, B)

**Parameters**

<i>point</i>	The considered point P
<i>lineStart-PointA</i>	The line start point A
<i>lineEnd-PointB</i>	The line end point B

Definition at line 198 of file Geometry2D.cpp.

Referenced by isToTheLeftOfLine(), and isToTheRightOfLine().

---

**6.74.2.43 bool Geometry2D::slopeOfLine ( const cv::Point2f & a, const cv::Point2f & b, double & slope ) [static]**

Compute the slope of the line defined by points "a" and "b".

Returns true if the slope of the line can be computed and false otherwise.

**Parameters**

<i>a</i>	Point a
<i>b</i>	Point b
<i>slope</i>	Slope of the line if it is different from (+/-)infinity

Definition at line 76 of file Geometry2D.cpp.

References multiscale::Numeric::almostEqual().

**6.74.2.44 void Geometry2D::tangentsFromPointToPolygon ( const std::vector< cv::Point2f > & convexPolygon, const cv::Point2f & referencePoint, cv::Point2f & leftMostTangentPoint, cv::Point2f & rightMostTangentPoint ) [static]**

Compute the polygon points where the tangents from a reference point touch the given polygon.

**Parameters**

<i>convex-Polygon</i>	The considered convex polygon
<i>reference-Point</i>	The reference point through which the polygon tangents pass
<i>leftMost-Tangent-Point</i>	The left most polygon point through which the tangent passes
<i>rightMost-Tangent-Point</i>	The right most polygon point through which the tangent passes

Definition at line 137 of file Geometry2D.cpp.

References multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::computeTangentsPointsForConvexPolygon().

Referenced by multiscale::analysis::Detector::computePolygonAngle().

**6.74.2.45 void Geometry2D::translate ( cv::Point2f & point, const cv::Point2f & translation ) [static, private]**

Translate a point by the given values.

**Parameters**

<i>point</i>	The point
<i>translation</i>	Translation values

Definition at line 464 of file Geometry2D.cpp.

Referenced by lineCircleIntersection().

### 6.74.3 Member Data Documentation

**6.74.3.1 const int Geometry2D::MATRIX\_START\_INDEX = 0 [static]**

Definition at line 637 of file Geometry2D.hpp.

Referenced by isPointOnEdge().

**6.74.3.2 const double Geometry2D::PI = 3.  
14159265358979323846264338327950288419716939937510  
[static]**

Definition at line 636 of file Geometry2D.hpp.

Referenced by angleBtwPoints(), angleOfLineWrtOxAxis(), multiscale::analysis::CircularityMeasure< PointType >::compute(), multiscale::analysis::Cluster::isCircularMeasure(), and multiscale::analysis::Region::isCircularMeasure().

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

- Geometry2D.hpp
- Geometry2D.cpp

## 6.75 multiscale::verification::GlobalLogicPropertyAttribute Class - Reference

Class for representing a "globally" logic property attribute.

```
#include <GlobalLogicPropertyAttribute.hpp>
```

### Public Attributes

- unsigned long [startTimepoint](#)
- unsigned long [endTimepoint](#)
- [LogicPropertyAttributeType logicProperty](#)

### 6.75.1 Detailed Description

Class for representing a "globally" logic property attribute.

Definition at line 14 of file GlobalLogicPropertyAttribute.hpp.

### 6.75.2 Member Data Documentation

#### 6.75.2.1 `unsigned long multiscale::verification::GlobalLogicPropertyAttribute::end-Timepoint`

The considered end timepoint

Definition at line 19 of file GlobalLogicPropertyAttribute.hpp.

#### 6.75.2.2 `LogicPropertyAttributeType multiscale::verification::GlobalLogicPropertyAttribute::logicProperty`

The logic property following the "globally" operator

Definition at line 20 of file GlobalLogicPropertyAttribute.hpp.

Referenced by `multiscale::verification::LogicPropertyVisitor::evaluateTemporalLogicPropertyWithStartAndEndTimepoints()`.

#### 6.75.2.3 `unsigned long multiscale::verification::GlobalLogicPropertyAttribute::::startTimepoint`

The considered start timepoint

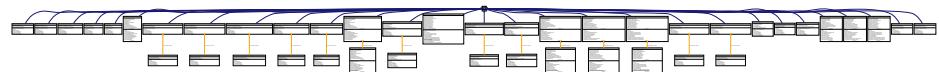
Definition at line 18 of file GlobalLogicPropertyAttribute.hpp.

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

- GlobalLogicPropertyAttribute.hpp

## 6.76 grammar Class Reference

Inheritance diagram for grammar:



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

- LogicPropertyGrammar.hpp

## 6.77 multiscale::verification::HeterogeneousTimeseriesComponent-Attribute Class Reference

Class for representing a heterogeneous timeseries component attribute.

```
#include <HeterogeneousTimeseriesComponentAttribute.hpp>
```

### Public Attributes

- HeterogeneousTimeseriesComponentType heterogeneousTimeseriesComponent

#### 6.77.1 Detailed Description

Class for representing a heterogeneous timeseries component attribute.

Definition at line 28 of file HeterogeneousTimeseriesComponentAttribute.hpp.

#### 6.77.2 Member Data Documentation

##### 6.77.2.1 HeterogeneousTimeseriesComponentType multiscale::verification::HeterogeneousTimeseriesComponentAttribute- ::heterogeneousTimeseriesComponent

The heterogeneous timeseries component

Definition at line 32 of file HeterogeneousTimeseriesComponentAttribute.hpp.

Referenced by multiscale::verification::TimeseriesComponentVisitor::operator()().

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

- HeterogeneousTimeseriesComponentAttribute.hpp

## 6.78 multiscale::verification::HeterogeneousTimeseriesComponent- TypeParser Struct Reference

Symbol table and parser for the heterogeneous timeseries component type.

```
#include <SymbolTables.hpp>
```

### Public Member Functions

- HeterogeneousTimeseriesComponentTypeParser ()

### 6.78.1 Detailed Description

Symbol table and parser for the heterogeneous timeseries component type.

Definition at line 114 of file SymbolTables.hpp.

### 6.78.2 Constructor & Destructor Documentation

**6.78.2.1 multiscale::verification::HeterogeneousTimeseriesComponentTypeParser::HeterogeneousTimeseriesComponentTypeParser ( )**  
`[inline]`

Definition at line 117 of file SymbolTables.hpp.

References multiscale::verification::Valley.

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

- SymbolTables.hpp

## 6.79 multiscale::verification::TimeseriesComponentEvaluator::HomogeneousComponentEvaluator< Relation > Class - Template Reference

```
#include <TimeseriesComponentEvaluator.hpp>
```

### Public Member Functions

- bool `isStartIndex` (std::size\_t index, std::size\_t nrOfValues, const std::vector<double> &values, const Relation &relation) const  
*Check if the given index is a homogeneous component start index.*
- std::size\_t `computeEndIndex` (std::size\_t startIndex, std::size\_t nrOfValues, const std::vector<double> &values, const Relation &relation) const  
*Compute the homogeneous component end index considering the given homogeneous component start index.*

### Private Member Functions

- std::size\_t `hasValidSuccessor` (std::size\_t index, std::size\_t nrOfValues, const std::vector<double> &values, const Relation &relation) const  
*Check if the provided homogeneous component index has a valid successor.*

**6.79.1 Detailed Description**

```
template<typename Relation>class multiscale::verification::TimeseriesComponentEvaluator::HomogeneousComponentEvaluator< Relation >
```

Definition at line 117 of file TimeseriesComponentEvaluator.hpp.

**6.79.2 Member Function Documentation**

```
6.79.2.1 template<typename Relation > std::size_t multiscale::verification::Timeseries-ComponentEvaluator::HomogeneousComponentEvaluator< Relation >::computeEndIndex ( std::size_t startIndex, std::size_t nrOfValues, const std::vector< double > & values, const Relation & relation ) const [inline]
```

Compute the homogeneous component end index considering the given homogeneous component start index.

A value located at index i in the collection,  $0 < i < (\text{values.size()} - 1)$ , is a nonuniform homogeneous component ending index if and only if  $\text{relation}(\text{values}[i - 1], \text{values}[i])$  and  $\neg \text{relation}(\text{values}[i], \text{values}[i + 1])$ , respectively  $\text{relation}(\text{values}[i - 1], \text{values}[i])$  if  $i = (\text{values.size()} - 1)$ .

**Parameters**

<i>startIndex</i>	The given start index
<i>nrOfValues</i>	The number of considered values
<i>values</i>	The collection of values
<i>relation</i>	The considered relation

Definition at line 167 of file TimeseriesComponentEvaluator.hpp.

References `multiscale::verification::TimeseriesComponentEvaluator::Homogeneous-ComponentEvaluator< Relation >::hasValidSuccessor()`, and `multiscale::verification::TimeseriesComponentEvaluator::HomogeneousComponentEvaluator< Relation >::is-StartIndex()`.

```
6.79.2.2 template<typename Relation > std::size_t multiscale::verification::Timeseries-ComponentEvaluator::HomogeneousComponentEvaluator< Relation >::hasValidSuccessor ( std::size_t index, std::size_t nrOfValues, const std::vector< double > & values, const Relation & relation ) const [inline, private]
```

Check if the provided homogeneous component index has a valid successor.

**Parameters**

<i>index</i>	The given index
<i>nrOfValues</i>	The number of considered values
<i>values</i>	The collection of values
<i>relation</i>	The considered relation

Definition at line 192 of file TimeseriesComponentEvaluator.hpp.

Referenced by multiscale::verification::TimeseriesComponentEvaluator::HomogeneousComponentEvaluator< Relation >::computeEndIndex(), and multiscale::verification::TimeseriesComponentEvaluator::HomogeneousComponentEvaluator< Relation >::isStartIndex().

```
6.79.2.3 template<typename Relation> bool multiscale::verification::Timeseries-
ComponentEvaluator::HomogeneousComponentEvaluator< Relation
>::isStartIndex ( std::size_t index, std::size_t nrOfValues, const std::vector<
double > & values, const Relation & relation ) const [inline]
```

Check if the given index is a homogeneous component start index.

A value located at index  $i$  in the collection,  $0 < i < (\text{values.size() - 1})$ , is a nonuniform homogeneous component starting index if and only if  $\text{!relation}(\text{values}[i - 1], \text{values}[i])$  and  $\text{relation}(\text{values}[i], \text{values}[i + 1])$ , respectively  $\text{relation}(\text{values}[i], \text{values}[i + 1])$  if  $i = 0$ .

#### Parameters

<i>index</i>	The given index
<i>nrOfValues</i>	The number of considered values
<i>values</i>	The collection of values
<i>relation</i>	The considered relation

Definition at line 132 of file TimeseriesComponentEvaluator.hpp.

References multiscale::verification::TimeseriesComponentEvaluator::HomogeneousComponentEvaluator< Relation >::hasValidSuccessor().

Referenced by multiscale::verification::TimeseriesComponentEvaluator::HomogeneousComponentEvaluator< Relation >::computeEndIndex().

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

- TimeseriesComponentEvaluator.hpp

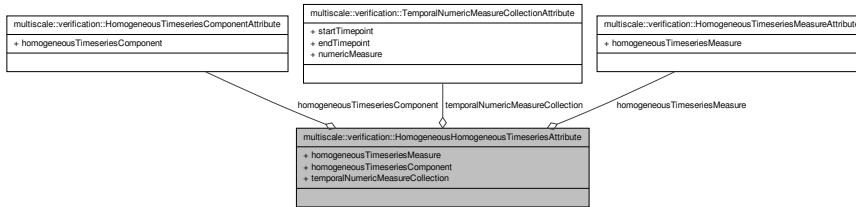
## 6.80 multiscale::verification::HomogeneousHomogeneousTimeseries-Attribute Class Reference

Class for representing a homogeneous homogeneous timeseries attribute.

```
#include <HomogeneousHomogeneousTimeseriesAttribute.hpp>
```

Collaboration diagram for multiscale::verification::HomogeneousHomogeneous-

TimeseriesAttribute:



## Public Attributes

- `HomogeneousTimeseriesMeasureAttribute homogeneousTimeseriesMeasure`
- `HomogeneousTimeseriesComponentAttribute homogeneousTimeseries-Component`
- `TemporalNumericMeasureCollectionAttribute temporalNumericMeasureCollection`

### 6.80.1 Detailed Description

Class for representing a homogeneous homogeneous timeseries attribute.

Definition at line 17 of file `HomogeneousHomogeneousTimeseriesAttribute.hpp`.

### 6.80.2 Member Data Documentation

#### 6.80.2.1 HomogeneousTimeseriesComponentAttribute

`multiscale::verification::HomogeneousHomogeneousTimeseries-Attribute::homogeneousTimeseriesComponent`

The homogeneous timeseries component

Definition at line 24 of file `HomogeneousHomogeneousTimeseriesAttribute.hpp`.

Referenced by `multiscale::verification::NumericMeasureCollectionVisitor::operator()()`.

#### 6.80.2.2 HomogeneousTimeseriesMeasureAttribute

`multiscale::verification::HomogeneousHomogeneousTimeseries-Attribute::homogeneousTimeseriesMeasure`

The homogeneous timeseries measure

Definition at line 22 of file `HomogeneousHomogeneousTimeseriesAttribute.hpp`.

Referenced by `multiscale::verification::NumericMeasureCollectionVisitor::operator()()`.

### 6.80.2.3 **TemporalNumericMeasureCollectionAttribute** multiscale::verification::HomogeneousHomogeneousTimeseries- Attribute::temporalNumericMeasureCollection

The temporal numeric measure collection

Definition at line 26 of file HomogeneousHomogeneousTimeseriesAttribute.hpp.

Referenced by multiscale::verification::NumericMeasureCollectionVisitor::operator()().

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

- HomogeneousHomogeneousTimeseriesAttribute.hpp

## 6.81 multiscale::verification::HomogeneousTimeseriesComponent- Attribute Class Reference

Class for representing a homogeneous timeseries component attribute.

```
#include <HomogeneousTimeseriesComponentAttribute.hpp>
```

### Public Attributes

- [HomogeneousTimeseriesComponentType homogeneousTimeseriesComponent](#)

### 6.81.1 Detailed Description

Class for representing a homogeneous timeseries component attribute.

Definition at line 31 of file HomogeneousTimeseriesComponentAttribute.hpp.

### 6.81.2 Member Data Documentation

#### 6.81.2.1 **HomogeneousTimeseriesComponentType** multiscale::verification::HomogeneousTimeseriesComponentAttribute- ::homogeneousTimeseriesComponent

The homogeneous timeseries component

Definition at line 35 of file HomogeneousTimeseriesComponentAttribute.hpp.

Referenced by multiscale::verification::TimeseriesComponentEvaluator::evaluate().

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

- HomogeneousTimeseriesComponentAttribute.hpp

## **6.82 multiscale::verification::HomogeneousTimeseriesComponentTypeParser Struct Reference**

Symbol table and parser for the homogeneous timeseries component type.

```
#include <SymbolTables.hpp>
```

### **Public Member Functions**

- [HomogeneousTimeseriesComponentTypeParser \(\)](#)

#### **6.82.1 Detailed Description**

Symbol table and parser for the homogeneous timeseries component type.

Definition at line 127 of file SymbolTables.hpp.

#### **6.82.2 Constructor & Destructor Documentation**

##### **6.82.2.1 multiscale::verification::HomogeneousTimeseriesComponentTypeParser::HomogeneousTimeseriesComponentTypeParser ( )** [inline]

Definition at line 130 of file SymbolTables.hpp.

References [multiscale::verification::Descent](#), and [multiscale::verification::UniformAscent](#).

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

- [SymbolTables.hpp](#)

## **6.83 multiscale::verification::HomogeneousTimeseriesMeasureAttribute Class Reference**

Class for representing a homogeneous timeseries measure attribute.

```
#include <HomogeneousTimeseriesMeasureAttribute.hpp>
```

### **Public Attributes**

- [HomogeneousTimeseriesMeasureType homogeneousTimeseriesMeasure](#)

### 6.83.1 Detailed Description

Class for representing a homogeneous timeseries measure attribute.

Definition at line 28 of file HomogeneousTimeseriesMeasureAttribute.hpp.

### 6.83.2 Member Data Documentation

#### 6.83.2.1 HomogeneousTimeseriesMeasureType multiscale::verification::HomogeneousTimeseriesMeasureAttribute::homogeneousTimeseriesMeasure

The homogeneous timeseries measure

Definition at line 32 of file HomogeneousTimeseriesMeasureAttribute.hpp.

Referenced by multiscale::verification::NumericMeasureCollectionVisitor::operator()().

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

- HomogeneousTimeseriesMeasureAttribute.hpp

## 6.84 multiscale::verification::HomogeneousTimeseriesMeasureTypeParser Struct Reference

Symbol table and parser for the homogeneous timeseries measure type.

```
#include <SymbolTables.hpp>
```

### Public Member Functions

- [HomogeneousTimeseriesMeasureTypeParser \(\)](#)

### 6.84.1 Detailed Description

Symbol table and parser for the homogeneous timeseries measure type.

Definition at line 143 of file SymbolTables.hpp.

### 6.84.2 Constructor & Destructor Documentation

#### 6.84.2.1 multiscale::verification::HomogeneousTimeseriesMeasureTypeParser::HomogeneousTimeseriesMeasureTypeParser ( ) [inline]

Definition at line 146 of file SymbolTables.hpp.

References multiscale::verification::Values.

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

- SymbolTables.hpp

## **6.85 multiscale::verification::ImplicationConstraintAttribute Class Reference**

Class for representing an "implication" constraint attribute.

```
#include <ImplicationConstraintAttribute.hpp>
```

### **Public Attributes**

- [ConstraintAttributeType constraint](#)

#### **6.85.1 Detailed Description**

Class for representing an "implication" constraint attribute.

Definition at line 14 of file ImplicationConstraintAttribute.hpp.

#### **6.85.2 Member Data Documentation**

##### **6.85.2.1 ConstraintAttributeType multiscale::verification::ImplicationConstraintAttribute::constraint**

The constraint following the "implication" operator

Definition at line 18 of file ImplicationConstraintAttribute.hpp.

Referenced by multiscale::verification::ConstraintVisitor::operator()().

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

- ImplicationConstraintAttribute.hpp

## **6.86 multiscale::verification::ImplicationLogicPropertyAttribute - Class Reference**

Class for representing an "implication" logic property attribute.

```
#include <ImplicationLogicPropertyAttribute.hpp>
```

## Public Attributes

- [LogicPropertyAttributeType logicProperty](#)

### 6.86.1 Detailed Description

Class for representing an "implication" logic property attribute.

Definition at line 14 of file ImplicationLogicPropertyAttribute.hpp.

### 6.86.2 Member Data Documentation

#### 6.86.2.1 LogicPropertyAttributeType multiscale::verification::ImplicationLogicPropertyAttribute::logicProperty

The logical property following the "implication" operator

Definition at line 18 of file ImplicationLogicPropertyAttribute.hpp.

Referenced by multiscale::verification::LogicPropertyVisitor::operator()().

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

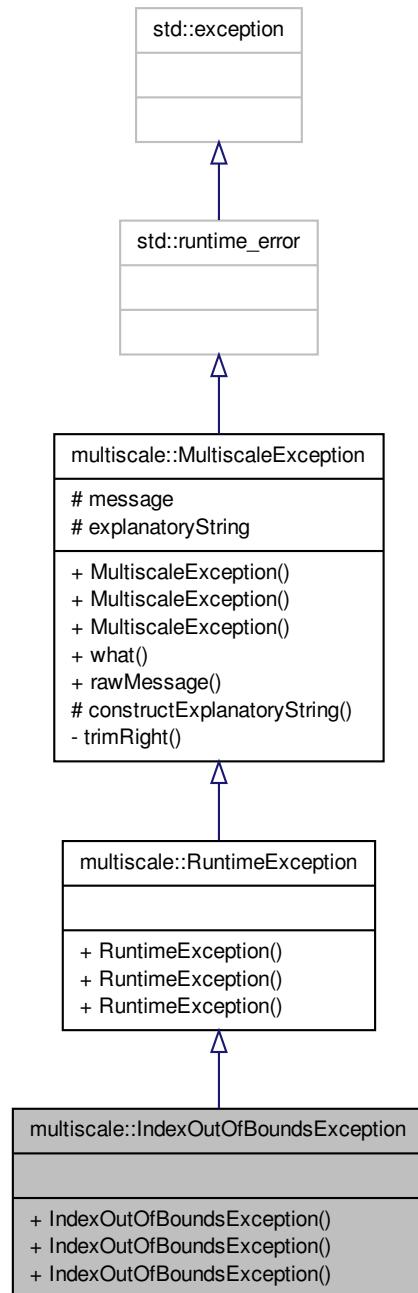
- [ImplicationLogicPropertyAttribute.hpp](#)

## 6.87 multiscale::IndexOutOfBoundsException Class Reference

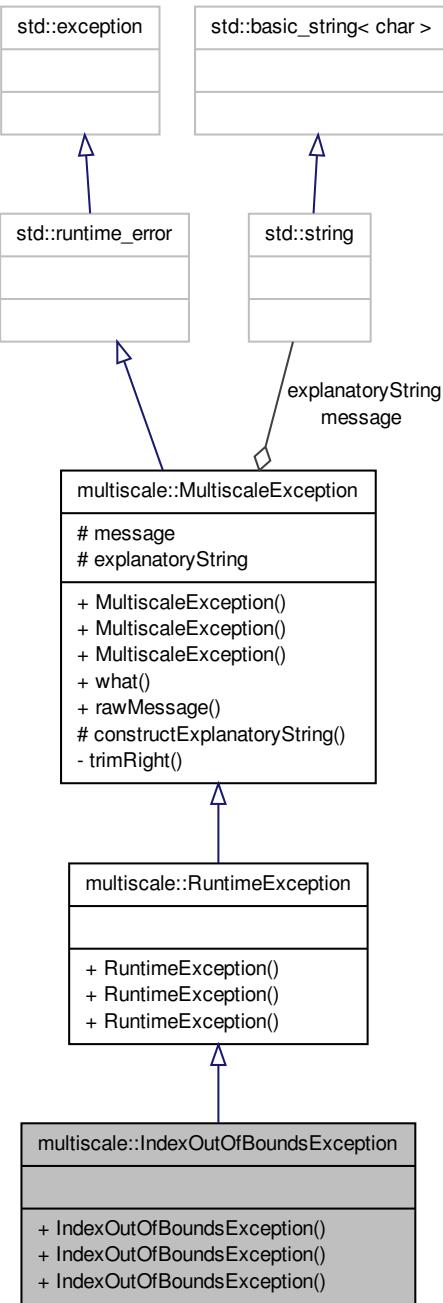
Class for representing an index out of bounds exception.

```
#include <IndexOutOfBoundsException.hpp>
```

Inheritance diagram for multiscale::IndexOutOfBoundsException:



Collaboration diagram for multiscale::IndexOutOfBoundsException:



## Public Member Functions

- [IndexOutOfBoundsException \(\)](#)
- [IndexOutOfBoundsException \(const std::string &file, int line, const std::string &msg\)](#)
- [IndexOutOfBoundsException \(const std::string &file, int line, const char \\*msg\)](#)

### 6.87.1 Detailed Description

Class for representing an index out of bounds exception.

Definition at line 12 of file IndexOutOfBoundsException.hpp.

### 6.87.2 Constructor & Destructor Documentation

#### 6.87.2.1 multiscale::IndexOutOfBoundsException::IndexOutOfBoundsException ( ) [inline]

Definition at line 16 of file IndexOutOfBoundsException.hpp.

#### 6.87.2.2 multiscale::IndexOutOfBoundsException::IndexOutOfBoundsException ( const std::string & file, int line, const std::string & msg ) [inline, explicit]

Definition at line 23 of file IndexOutOfBoundsException.hpp.

References multiscale::ERR\_INDEX\_OUT\_OF\_BOUNDS\_BEGIN, and multiscale::ERR\_INDEX\_OUT\_OF\_BOUNDS\_END.

#### 6.87.2.3 multiscale::IndexOutOfBoundsException::IndexOutOfBoundsException ( const std::string & file, int line, const char \* msg ) [inline, explicit]

Definition at line 34 of file IndexOutOfBoundsException.hpp.

References multiscale::ERR\_INDEX\_OUT\_OF\_BOUNDS\_BEGIN, and multiscale::ERR\_INDEX\_OUT\_OF\_BOUNDS\_END.

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

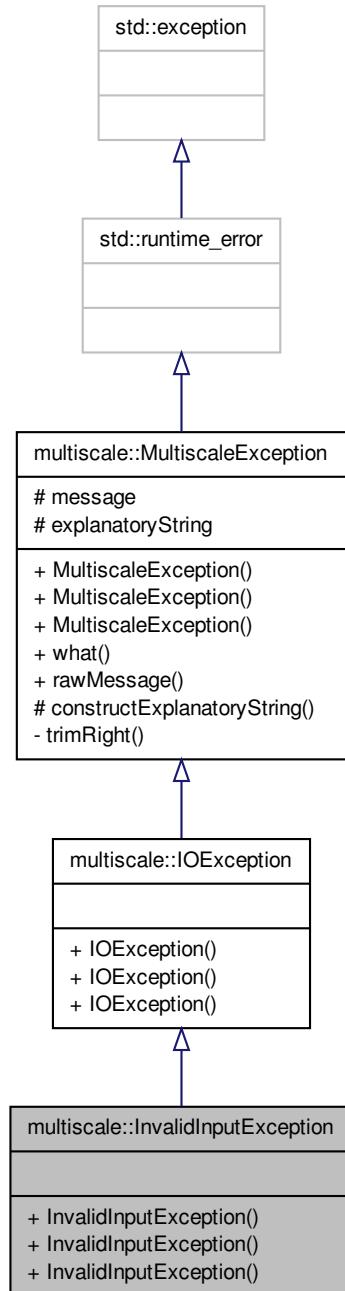
- [IndexOutOfBoundsException.hpp](#)

## 6.88 multiscale::InvalidInputException Class Reference

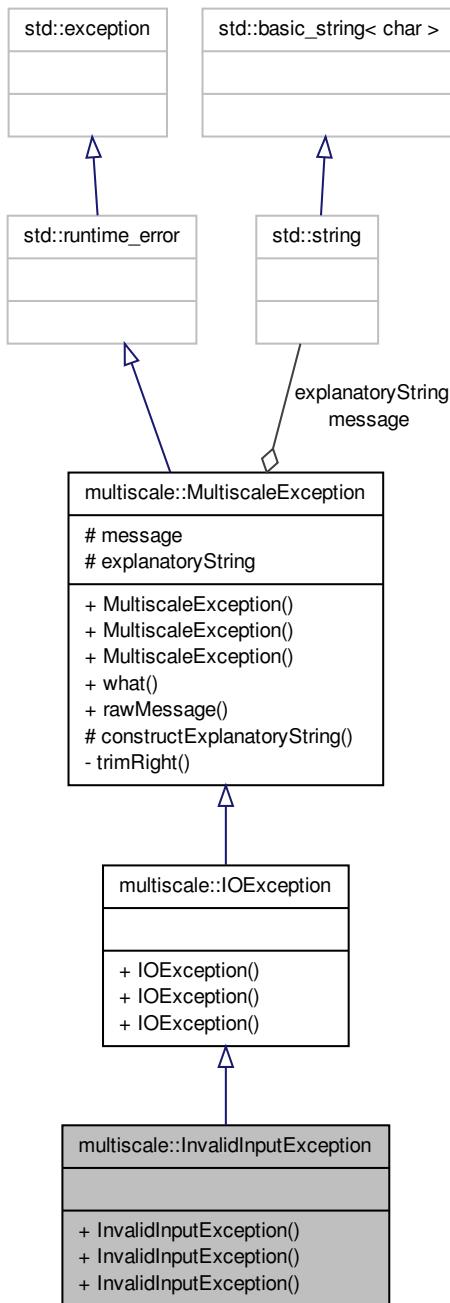
Class for representing invalid input exceptions.

```
#include <InvalidInputException.hpp>
```

Inheritance diagram for multiscale::InvalidInputException:



Collaboration diagram for multiscale::InvalidInputException:



## Public Member Functions

- [InvalidInputException \(\)](#)
- [InvalidInputException \(const std::string &file, int line, const std::string &msg\)](#)
- [InvalidInputException \(const std::string &file, int line, const char \\*msg\)](#)

### 6.88.1 Detailed Description

Class for representing invalid input exceptions.

Definition at line 12 of file InvalidInputException.hpp.

### 6.88.2 Constructor & Destructor Documentation

#### 6.88.2.1 multiscale::InvalidInputException::InvalidInputException ( ) [inline]

Definition at line 16 of file InvalidInputException.hpp.

#### 6.88.2.2 multiscale::InvalidInputException::InvalidInputException ( const std::string & file, int line, const std::string & msg ) [inline, explicit]

Definition at line 18 of file InvalidInputException.hpp.

#### 6.88.2.3 multiscale::InvalidInputException::InvalidInputException ( const std::string & file, int line, const char \* msg ) [inline, explicit]

Definition at line 23 of file InvalidInputException.hpp.

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

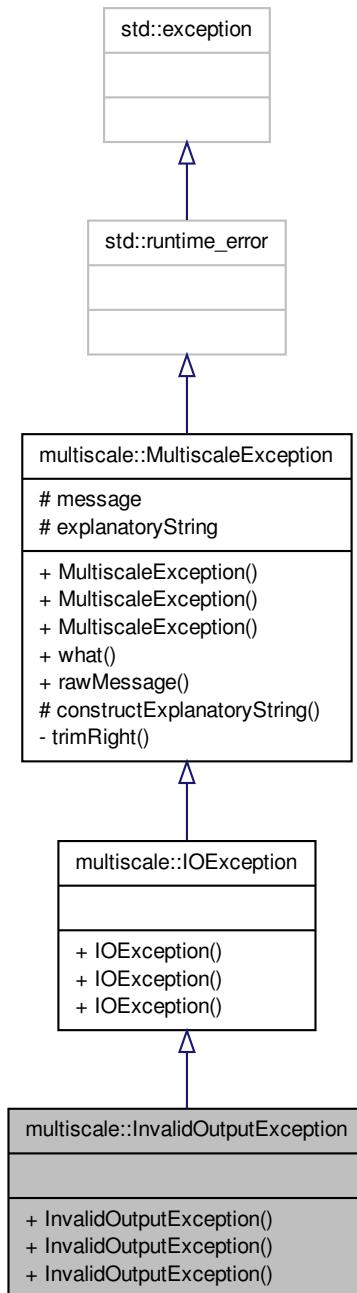
- [InvalidInputException.hpp](#)

## 6.89 multiscale::InvalidOutputException Class Reference

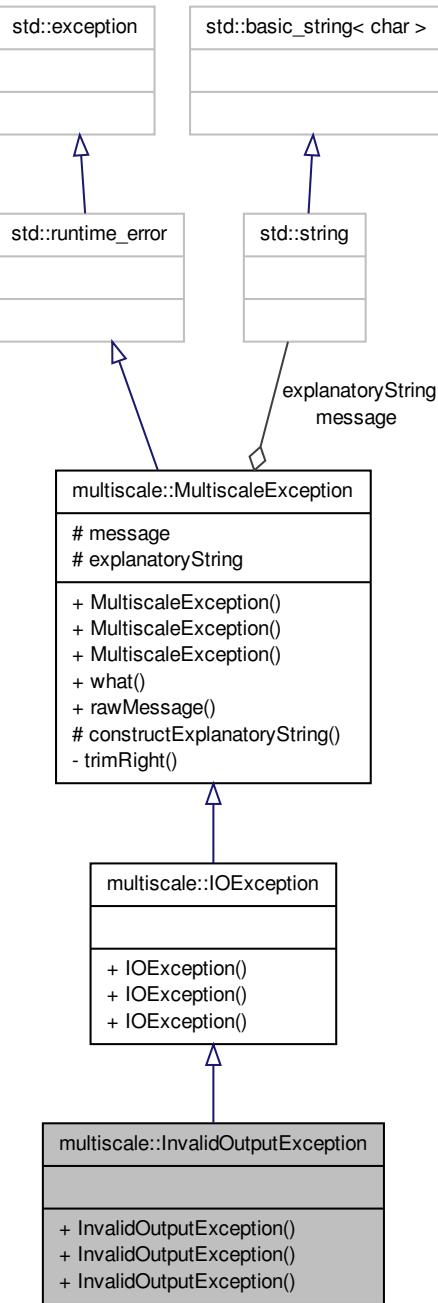
Class for representing invalid output exceptions.

```
#include <InvalidOutputException.hpp>
```

## Inheritance diagram for multiscale::InvalidOutputException:



Collaboration diagram for multiscale::InvalidOutputException:



## Public Member Functions

- [InvalidOutputException \(\)](#)
- [InvalidOutputException \(const std::string &file, int line, const std::string &msg\)](#)
- [InvalidOutputException \(const std::string &file, int line, const char \\*msg\)](#)

### 6.89.1 Detailed Description

Class for representing invalid output exceptions.

Definition at line 12 of file InvalidOutputException.hpp.

### 6.89.2 Constructor & Destructor Documentation

#### 6.89.2.1 `multiscale::InvalidOutputException::InvalidOutputException( ) [inline]`

Definition at line 16 of file InvalidOutputException.hpp.

#### 6.89.2.2 `multiscale::InvalidOutputException::InvalidOutputException( const std::string & file, int line, const std::string & msg ) [inline, explicit]`

Definition at line 18 of file InvalidOutputException.hpp.

#### 6.89.2.3 `multiscale::InvalidOutputException::InvalidOutputException( const std::string & file, int line, const char * msg ) [inline, explicit]`

Definition at line 23 of file InvalidOutputException.hpp.

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

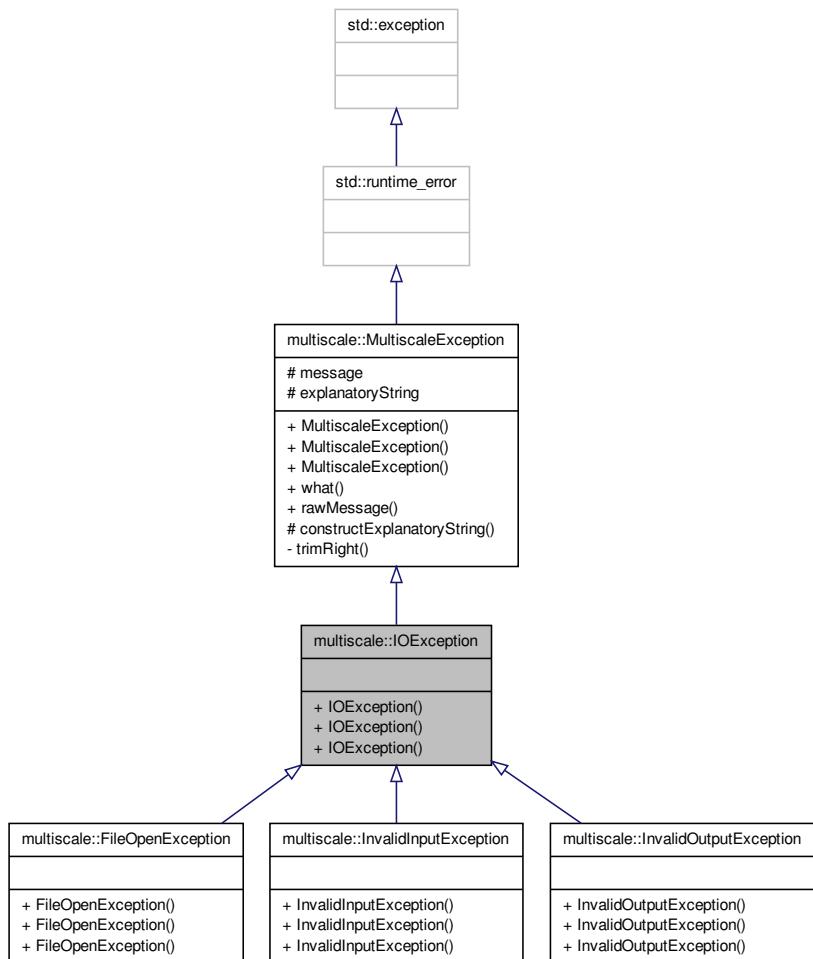
- [InvalidOutputException.hpp](#)

## 6.90 multiscale::IOException Class Reference

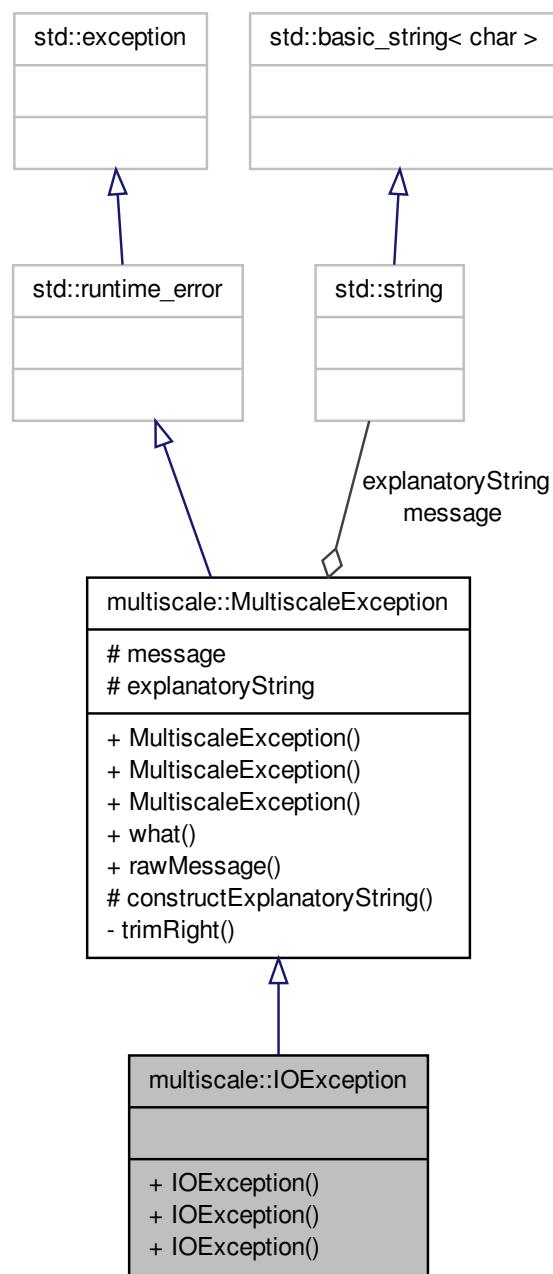
Class for representing input and output exceptions.

```
#include <IOException.hpp>
```

Inheritance diagram for multiscale::IOException:



Collaboration diagram for multiscale::IOException:



## Public Member Functions

- [IOException \(\)](#)
- [IOException \(const std::string &file, int line, const std::string &msg\)](#)
- [IOException \(const std::string &file, int line, const char \\*msg\)](#)

### 6.90.1 Detailed Description

Class for representing input and output exceptions.

Definition at line 12 of file IOException.hpp.

### 6.90.2 Constructor & Destructor Documentation

#### 6.90.2.1 multiscale::IOException::IOException( ) [inline]

Definition at line 16 of file IOException.hpp.

#### 6.90.2.2 multiscale::IOException::IOException( const std::string & file, int line, const std::string & msg ) [inline, explicit]

Definition at line 18 of file IOException.hpp.

#### 6.90.2.3 multiscale::IOException::IOException( const std::string & file, int line, const char \* msg ) [inline, explicit]

Definition at line 23 of file IOException.hpp.

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

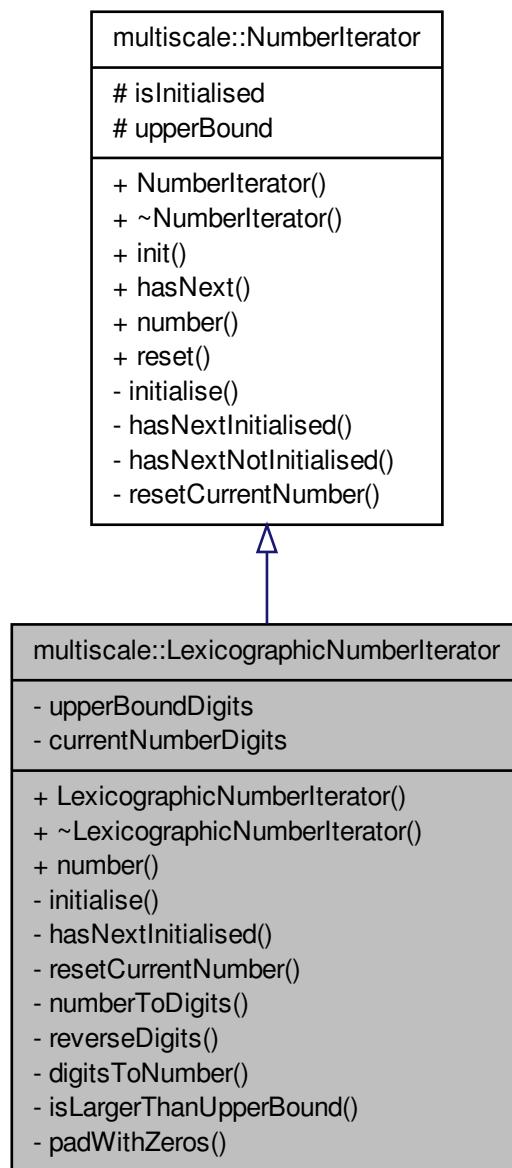
- [IOException.hpp](#)

## 6.91 multiscale::LexicographicNumberIterator Class Reference

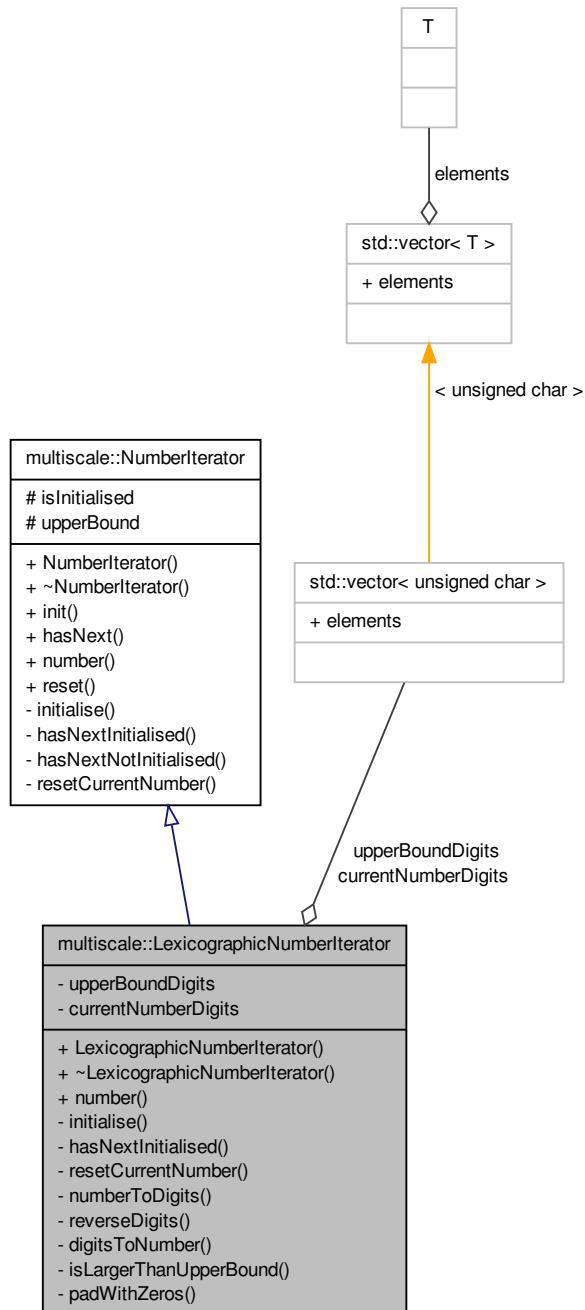
Iterator class starting at 1 and ending at the provided upper bound considering that each number is followed by an "\_".

```
#include <LexicographicNumberIterator.hpp>
```

Inheritance diagram for multiscale::LexicographicNumberIterator:



Collaboration diagram for multiscale::LexicographicNumberIterator:



## Public Member Functions

- `LexicographicNumberIterator (unsigned int upperBound)`
- `~LexicographicNumberIterator ()`
- `unsigned int number ()`

*Get the number pointed by the iterator.*

## Private Member Functions

- `void initialise ()`  
*Initialise the vectors of digits.*
- `bool hasNextInitialised ()`  
*Check if there is a next number when in initialised state.*
- `void resetCurrentNumber ()`  
*Reset the digits of the current number to the initial value.*
- `void numberToDigits (unsigned int number, std::vector< unsigned char > &digits)`  
*Convert the number to a vector of digits.*
- `void reverseDigits (std::vector< unsigned char > &digits)`  
*Reverse the order of the digits.*
- `unsigned int digitsToNumber (std::vector< unsigned char > &digits)`  
*Convert the vector of digits to the number they represent.*
- `bool isLargerThanUpperBound (unsigned char lastDigit)`  
*Check if the current number with the provided last digit is greater than the upper bound.*
- `void padWithZeros ()`  
*Pad the current number with zeros.*

## Private Attributes

- `std::vector< unsigned char > upperBoundDigits`
- `std::vector< unsigned char > currentNumberDigits`

### 6.91.1 Detailed Description

Iterator class starting at 1 and ending at the provided upper bound considering that each number is followed by an "\_".

Definition at line 12 of file LexicographicNumberIterator.hpp.

### 6.91.2 Constructor & Destructor Documentation

6.91.2.1 **LexicographicNumberIterator::LexicographicNumberIterator ( unsigned int *upperBound* )**

Definition at line 6 of file LexicographicNumberIterator.cpp.

References initialise(), and multiscale::NumberIterator::reset().

6.91.2.2 **LexicographicNumberIterator::~LexicographicNumberIterator ( )**

Definition at line 11 of file LexicographicNumberIterator.cpp.

References currentNumberDigits, and upperBoundDigits.

### 6.91.3 Member Function Documentation

6.91.3.1 **unsigned int LexicographicNumberIterator::digitsToNumber ( std::vector< unsigned char > & *digits* ) [private]**

Convert the vector of digits to the number they represent.

#### Parameters

<i>digits</i>	The digits
---------------	------------

Definition at line 74 of file LexicographicNumberIterator.cpp.

References number().

Referenced by number(), and padWithZeros().

6.91.3.2 **bool LexicographicNumberIterator::hasNextInitialised ( ) [private, virtual]**

Check if there is a next number when in initialised state.

Implements [multiscale::NumberIterator](#).

Definition at line 26 of file LexicographicNumberIterator.cpp.

References currentNumberDigits, isLargerThanUpperBound(), and padWithZeros().

6.91.3.3 **void LexicographicNumberIterator::initialise ( ) [private, virtual]**

Initialise the vectors of digits.

Implements [multiscale::NumberIterator](#).

Definition at line 20 of file LexicographicNumberIterator.cpp.

References currentNumberDigits, numberToDigits(), multiscale::NumberIterator::upperBound, and upperBoundDigits.

Referenced by LexicographicNumberIterator().

#### 6.91.3.4 bool LexicographicNumberIterator::isLargerThanUpperBound ( unsigned char *lastDigit* ) [private]

Check if the current number with the provided last digit is greater than the upper bound.

Check if the current number is greater than the upper bound when replacing the last digit of the current number with the provided digit

##### Parameters

<i>lastDigit</i>	The last digit
------------------	----------------

Definition at line 86 of file LexicographicNumberIterator.cpp.

References currentNumberDigits, and upperBoundDigits.

Referenced by hasNextInitialised().

#### 6.91.3.5 unsigned int LexicographicNumberIterator::number ( ) [virtual]

Get the number pointed by the iterator.

Implements [multiscale::NumberIterator](#).

Definition at line 16 of file LexicographicNumberIterator.cpp.

References currentNumberDigits, and digitsToNumber().

Referenced by digitsToNumber().

#### 6.91.3.6 void LexicographicNumberIterator::numberToDigits ( unsigned int *number*, std::vector< unsigned char > & *digits* ) [private]

Convert the number to a vector of digits.

##### Parameters

<i>number</i>	The number
<i>digits</i>	The digits of the number

Definition at line 53 of file LexicographicNumberIterator.cpp.

References reverseDigits().

Referenced by initialise().

**6.91.3.7 void LexicographicNumberIterator::padWithZeros( ) [private]**

Pad the current number with zeros.

Pad the current number with the maximum number of zeros such that it does not to become larger than the upper bound

Definition at line 107 of file LexicographicNumberIterator.cpp.

References currentNumberDigits, digitsToNumber(), multiscale::NumberIterator::upperBound, and upperBoundDigits.

Referenced by hasNextInitialised().

**6.91.3.8 void LexicographicNumberIterator::resetCurrentNumber( ) [private, virtual]**

Reset the digits of the current number to the initial value.

Implements [multiscale::NumberIterator](#).

Definition at line 42 of file LexicographicNumberIterator.cpp.

References currentNumberDigits, and upperBoundDigits.

**6.91.3.9 void LexicographicNumberIterator::reverseDigits( std::vector< unsigned char > & digits ) [private]**

Reverse the order of the digits.

Reverse the order of the digits such that the first one is swapped with the last one, the second one is swapped with the last but one and so on.

**Parameters**

<i>digits</i>	The digits
---------------	------------

Definition at line 63 of file LexicographicNumberIterator.cpp.

Referenced by numberToDigits().

**6.91.4 Member Data Documentation****6.91.4.1 std::vector<unsigned char> multiscale::LexicographicNumberIterator::currentNumberDigits [private]**

The digits of the number to which the iterator points

Definition at line 17 of file LexicographicNumberIterator.hpp.

Referenced by hasNextInitialised(), initialise(), isLargerThanUpperBound(), number(), padWithZeros(), resetCurrentNumber(), and ~LexicographicNumberIterator().

6.91.4.2 std::vector<unsigned char> multiscale::LexicographicNumberIterator-  
::upperBoundDigits [private]

The digits of the upper bound

Definition at line 16 of file LexicographicNumberIterator.hpp.

Referenced by initialise(), isLargerThanUpperBound(), padWithZeros(), resetCurrent-  
Number(), and ~LexicographicNumberIterator().

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

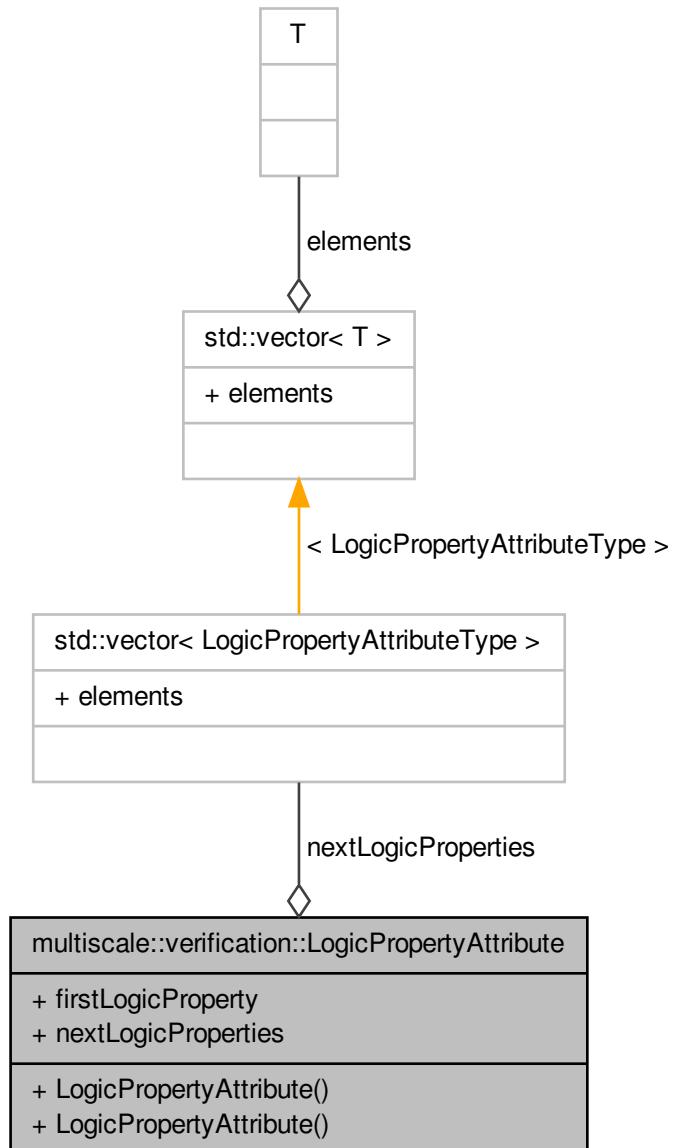
- LexicographicNumberIterator.hpp
  
- LexicographicNumberIterator.cpp

## 6.92 multiscale::verification::LogicPropertyAttribute Class Reference

Class for representing a logic property attribute.

```
#include <LogicPropertyAttribute.hpp>
```

Collaboration diagram for multiscale::verification::LogicPropertyAttribute:



## Public Member Functions

- `LogicPropertyAttribute ()`
- `LogicPropertyAttribute (const LogicPropertyAttributeType &firstLogicProperty, const std::vector< LogicPropertyAttributeType > &nextLogicProperties)`

## Public Attributes

- `LogicPropertyAttributeType firstLogicProperty`
- `std::vector < LogicPropertyAttributeType > nextLogicProperties`

### 6.92.1 Detailed Description

Class for representing a logic property attribute.

Definition at line 40 of file LogicPropertyAttribute.hpp.

### 6.92.2 Constructor & Destructor Documentation

#### 6.92.2.1 multiscale::verification::LogicPropertyAttribute::LogicPropertyAttribute ( ) [inline]

Definition at line 49 of file LogicPropertyAttribute.hpp.

#### 6.92.2.2 multiscale::verification::LogicPropertyAttribute::LogicPropertyAttribute ( const LogicPropertyAttributeType & firstLogicProperty, const std::vector< LogicPropertyAttributeType > & nextLogicProperties ) [inline]

Definition at line 51 of file LogicPropertyAttribute.hpp.

References `firstLogicProperty`, and `nextLogicProperties`.

### 6.92.3 Member Data Documentation

#### 6.92.3.1 LogicPropertyAttributeType multiscale::verification::LogicProperty- Attribute::firstLogicProperty

The first logic property

Definition at line 44 of file LogicPropertyAttribute.hpp.

Referenced by `multiscale::verification::LogicPropertyVisitor::constructEvaluationLogicProperty()`, `LogicPropertyAttribute()`, and `multiscale::verification::LogicPropertyVisitor::operator()()`.

### 6.92.3.2 std::vector<LogicPropertyAttributeType> multiscale::verification::LogicPropertyAttribute::nextLogicProperties

The next logic properties

Definition at line 45 of file LogicPropertyAttribute.hpp.

Referenced by multiscale::verification::LogicPropertyVisitor::evaluateNextLogicProperties(), and LogicPropertyAttribute().

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

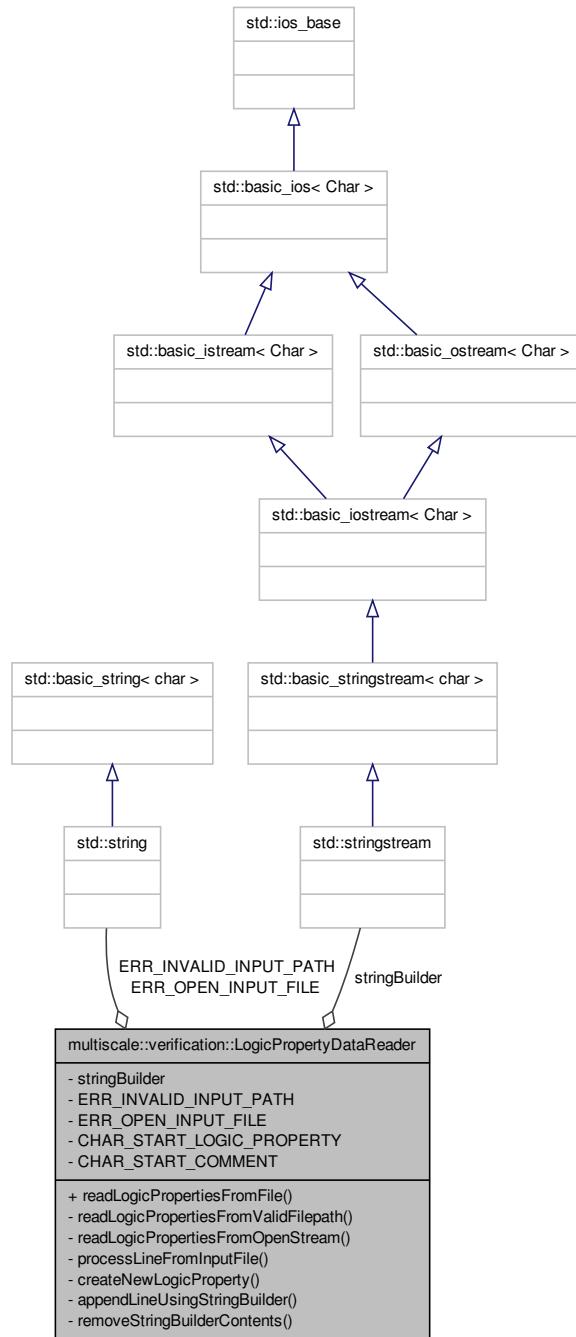
- LogicPropertyAttribute.hpp

## 6.93 multiscale::verification::LogicPropertyDataReader Class Reference

Class used to input logic properties.

```
#include <LogicPropertyDataReader.hpp>
```

Collaboration diagram for multiscale::verification::LogicPropertyDataReader:



## Public Member Functions

- std::vector< std::string > **readLogicPropertiesFromFile** (const std::string &inputFilepath)

*Return the logic properties read from a file.*

## Private Member Functions

- std::vector< std::string > **readLogicPropertiesFromValidFilepath** (const std::string &fin)

*Read the logic properties from the given file.*

- std::vector< std::string > **readLogicPropertiesFromOpenStream** (std::ifstream &fin)

*Read the logic properties from the given already opened input stream.*

- void **processLineFromInputFile** (const std::string &line, std::vector< std::string > &logicProperties)

*Process a line from the input file.*

- void **createNewLogicProperty** (std::vector< std::string > &logicProperties)

*Create a new logic property from the string builder contents.*

- void **appendLineUsingStringBuilder** (const std::string &line)

*Append the given line to the string builder contents.*

- void **removeStringBuilderContents** ()

*Remove the contents of the string builder.*

## Private Attributes

- std::stringstream **stringBuilder**

## Static Private Attributes

- static const std::string **ERR\_INVALID\_INPUT\_PATH** = "The path to the file containing the logic queries is invalid. Please change."
- static const std::string **ERR\_OPEN\_INPUT\_FILE** = "The file containing the logic queries could not be opened. Please make sure it is not used by another process."
- static const char **CHAR\_START\_LOGIC\_PROPERTY** = 'P'
- static const char **CHAR\_START\_COMMENT** = '#'

### 6.93.1 Detailed Description

Class used to input logic properties.

Definition at line 15 of file LogicPropertyDataReader.hpp.

### 6.93.2 Member Function Documentation

6.93.2.1 void LogicPropertyDataReader::appendLineUsingStringBuilder ( const std::string & *line* ) [private]

Append the given line to the string builder contents.

#### Parameters

<i>line</i>	The given line
-------------	----------------

Definition at line 70 of file LogicPropertyDataReader.cpp.

References stringBuilder.

Referenced by processLineFromInputFile().

6.93.2.2 void LogicPropertyDataReader::createNewLogicProperty ( std::vector< std::string > & *logicProperties* ) [private]

Create a new logic property from the string builder contents.

A new logic property is created only if the size of the string builder contents is greater than 0.

#### Parameters

<i>logicProperties</i>	The collection of logic properties obtained from the input file
------------------------	---

Definition at line 58 of file LogicPropertyDataReader.cpp.

References removeStringBuilderContents(), and stringBuilder.

Referenced by processLineFromInputFile(), and readLogicPropertiesFromOpenStream().

6.93.2.3 void LogicPropertyDataReader::processLineFromInputFile ( const std::string & *line*, std::vector< std::string > & *logicProperties* ) [private]

Process a line from the input file.

#### Parameters

<i>line</i>	The line read from the input file
<i>logicProperties</i>	The collection of logic properties obtained from the input file

Definition at line 44 of file LogicPropertyDataReader.cpp.

References appendLineUsingStringBuilder(), CHAR\_START\_COMMENT, CHAR\_START\_LOGIC\_PROPERTY, and createNewLogicProperty().

Referenced by `readLogicPropertiesFromOpenStream()`.

**6.93.2.4** `std::vector< std::string > LogicPropertyDataReader::readLogicPropertiesFromFile ( const std::string & inputfilepath )`

Return the logic properties read from a file.

All lines which start with "#" are used to write comments. All lines which start with "P" introduce a new logic property.

**Parameters**

<code><i>inputfilepath</i></code>	The path to the input file
-----------------------------------	----------------------------

Definition at line 9 of file `LogicPropertyDataReader.cpp`.

References `ERR_INVALID_INPUT_PATH`, `multiscale::Filesystem::isValidFilePath()`, and `readLogicPropertiesFromValidFilepath()`.

Referenced by `multiscale::verification::ModelCheckingManager::initialiseLogicProperties()`.

**6.93.2.5** `std::vector< std::string > LogicPropertyDataReader::readLogicPropertiesFromOpenStream ( std::ifstream & fin ) [private]`

Read the logic properties from the given already opened input stream.

**Parameters**

<code><i>fin</i></code>	The input stream
-------------------------	------------------

Definition at line 31 of file `LogicPropertyDataReader.cpp`.

References `createNewLogicProperty()`, and `processLineFromInputFile()`.

Referenced by `readLogicPropertiesFromValidFilepath()`.

**6.93.2.6** `std::vector< std::string > LogicPropertyDataReader::readLogicPropertiesFromValidFilepath ( const std::string & fin ) [private]`

Read the logic properties from the given file.

Definition at line 17 of file `LogicPropertyDataReader.cpp`.

References `ERR_OPEN_INPUT_FILE`, and `readLogicPropertiesFromOpenStream()`.

Referenced by `readLogicPropertiesFromFile()`.

6.93.2.7 `void LogicPropertyDataReader::removeStringBuilderContents( )`  
[private]

Remove the contents of the string builder.

Definition at line 74 of file LogicPropertyDataReader.cpp.

References stringBuilder.

Referenced by createNewLogicProperty().

### 6.93.3 Member Data Documentation

6.93.3.1 `const char LogicPropertyDataReader::CHAR_START_COMMENT = '#'`  
[static, private]

Definition at line 78 of file LogicPropertyDataReader.hpp.

Referenced by processLineFromInputFile().

6.93.3.2 `const char LogicPropertyDataReader::CHAR_START_LOGIC_PROPERTY  
= 'P'` [static, private]

Definition at line 77 of file LogicPropertyDataReader.hpp.

Referenced by processLineFromInputFile().

6.93.3.3 `const std::string LogicPropertyDataReader::ERR_INVALID_INPUT_PATH  
= "The path to the file containing the logic queries is invalid. Please change."`  
[static, private]

Definition at line 74 of file LogicPropertyDataReader.hpp.

Referenced by readLogicPropertiesFromFile().

6.93.3.4 `const std::string LogicPropertyDataReader::ERR_OPEN_INPUT_FILE = "The  
file containing the logic queries could not be opened. Please make sure it is not used by  
another process."` [static, private]

Definition at line 75 of file LogicPropertyDataReader.hpp.

Referenced by readLogicPropertiesFromValidFilepath().

6.93.3.5 `std::stringstream multiscale::verification::LogicPropertyDataReader-  
::stringBuilder` [private]

The string builder used to concatenate strings

Definition at line 19 of file LogicPropertyDataReader.hpp.

Referenced by appendLineUsingStringBuilder(), createNewLogicProperty(), and removeStringBuilderContents().

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

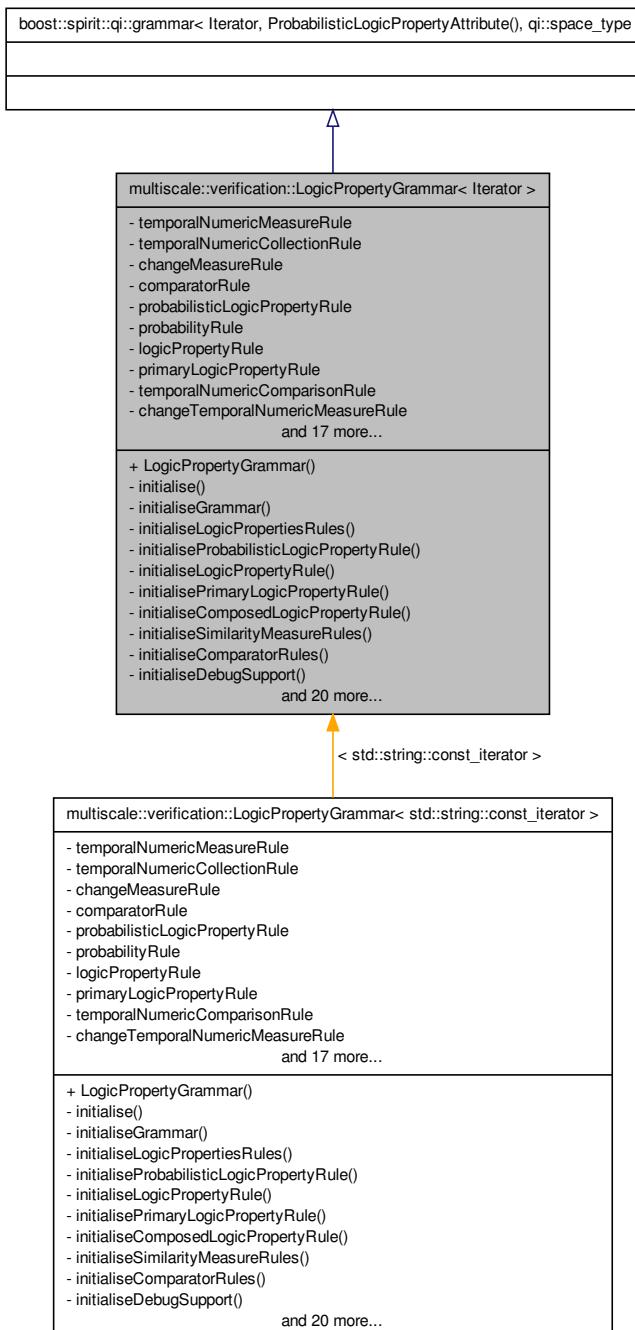
- LogicPropertyDataReader.hpp
  
- LogicPropertyDataReader.cpp

## 6.94 multiscale::verification::LogicPropertyGrammar< Iterator > - Class Template Reference

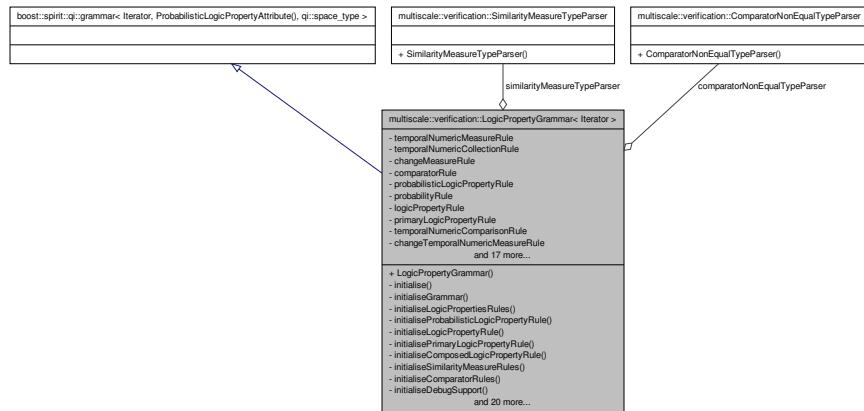
The grammar for parsing logic properties.

```
#include <LogicPropertyGrammar.hpp>
```

Inheritance diagram for multiscale::verification::LogicPropertyGrammar< Iterator >:



Collaboration diagram for multiscale::verification::LogicPropertyGrammar< Iterator >:



## Public Member Functions

- [LogicPropertyGrammar \(\)](#)

## Private Member Functions

- void [initialise \(\)](#)  
*Initialisation function.*
- void [initialiseGrammar \(\)](#)  
*Initialise the grammar.*
- void [initialiseLogicPropertiesRules \(\)](#)  
*Initialise the logic properties rules.*
- void [initialiseProbabilisticLogicPropertyRule \(\)](#)  
*Initialise the probabilistic logic property rule.*
- void [initialiseLogicPropertyRule \(\)](#)  
*Initialise the logic property rule.*
- void [initialisePrimaryLogicPropertyRule \(\)](#)  
*Initialise the primary logic property rule.*
- void [initialiseComposedLogicPropertyRule \(\)](#)  
*Initialise the composed logic property rule.*
- void [initialiseSimilarityMeasureRules \(\)](#)  
*Initialise the similarity measure rules.*
- void [initialiseComparatorRules \(\)](#)  
*Initialise the comparator rules.*
- void [initialiseDebugSupport \(\)](#)  
*Initialise debug support.*

- void [assignNamesToRules \(\)](#)  
*Assign names to the rules.*
- void [assignNamesToLogicPropertiesRules \(\)](#)  
*Assign names to logic properties rules.*
- void [assignNamesToProbabilisticLogicPropertyRules \(\)](#)  
*Assign names to the probabilistic logic property rules.*
- void [assignNamesToLogicPropertyRules \(\)](#)  
*Assign names to the logic property rules.*
- void [assignNamesToPrimaryLogicPropertyRules \(\)](#)  
*Assign names to the primary logic property rules.*
- void [assignNamesToComposedLogicPropertyRules \(\)](#)  
*Assign names to the composed logic property rules.*
- void [assignNamesToSimilarityMeasureRules \(\)](#)  
*Assign names to the similarity measure rules.*
- void [assignNamesToComparatorRules \(\)](#)  
*Assign names to the comparator rules.*
- void [initialiseRulesDebugging \(\)](#)  
*Initialise the debugging of rules.*
- void [initialiseLogicPropertiesRulesDebugging \(\)](#)  
*Initialise the debugging of the logic properties rules.*
- void [initialiseProbabilisticLogicPropertyRuleDebugging \(\)](#)  
*Initialise debugging for the probabilistic logic property rule.*
- void [initialiseLogicPropertyRuleDebugging \(\)](#)  
*Initialise debugging for the logic property rule.*
- void [initialisePrimaryLogicPropertyRuleDebugging \(\)](#)  
*Initialise debugging for the primary logic property rule.*
- void [initialiseComposedLogicPropertyRuleDebugging \(\)](#)  
*Initialise debugging for the composed logic property rule.*
- void [initialiseSimilarityMeasureRuleDebugging \(\)](#)  
*Initialise debugging for the similarity measure rule.*
- void [initialiseComparatorRuleDebugging \(\)](#)  
*Initialise debugging for the comparator rule.*
- void [initialiseErrorHandlerSupport \(\)](#)  
*Initialise the error handling routines.*
- void [initialiseLogicPropertiesErrorHandlerSupport \(\)](#)  
*Initialise the logic properties error handling support.*
- void [initialiseProbabilisticLogicPropertyErrorHandlerSupport \(\)](#)  
*Initialise the probabilistic logic property error handling support.*
- void [initialisePrimaryLogicPropertyErrorHandlerSupport \(\)](#)  
*Initialise the primary logic property error handling support.*
- void [initialiseComposedLogicPropertyErrorHandlerSupport \(\)](#)  
*Initialise the composed logic property error handling support.*

### Private Attributes

- `TemporalNumericMeasureGrammar < Iterator > temporalNumericMeasureRule`
- `TemporalNumericCollectionGrammar < Iterator > temporalNumericCollectionRule`
- `ChangeMeasureGrammar < Iterator > changeMeasureRule`
- `ComparatorGrammar < Iterator > comparatorRule`
- `qi::rule< Iterator, ProbabilisticLogicPropertyAttribute(), qi::space_type > probabilisticLogicPropertyRule`
- `qi::rule< Iterator, double(), qi::space_type > probabilityRule`
- `qi::rule< Iterator, LogicPropertyAttribute(), qi::space_type > logicPropertyRule`
- `qi::rule< Iterator, PrimaryLogicPropertyAttribute(), qi::space_type > primaryLogicPropertyRule`
- `qi::rule< Iterator, TemporalNumericComparisonAttribute(), qi::space_type > temporalNumericComparisonRule`
- `qi::rule< Iterator, ChangeTemporalNumericMeasureAttribute(), qi::space_type > changeTemporalNumericMeasureRule`
- `qi::rule< Iterator, SimilarityTemporalNumericCollectionAttribute(), qi::space_type > similarityTemporalNumericCollectionRule`
- `qi::rule< Iterator, SimilarityMeasureAttribute(), qi::space_type > similarityMeasureRule`
- `qi::rule< Iterator, NotLogicPropertyAttribute(), qi::space_type > notLogicPropertyRule`
- `qi::rule< Iterator, FutureLogicPropertyAttribute(), qi::space_type > futureLogicPropertyRule`
- `qi::rule< Iterator, GlobalLogicPropertyAttribute(), qi::space_type > globalLogicPropertyRule`
- `qi::rule< Iterator, NextLogicPropertyAttribute(), qi::space_type > nextLogicPropertyRule`
- `qi::rule< Iterator, NextKLogicPropertyAttribute(), qi::space_type > nextKLogicPropertyRule`
- `qi::rule< Iterator, AndLogicPropertyAttribute(), qi::space_type > andLogicPropertyRule`
- `qi::rule< Iterator, OrLogicPropertyAttribute(), qi::space_type > orLogicPropertyRule`
- `qi::rule< Iterator, ImplicationLogicPropertyAttribute(), qi::space_type > implicationLogicPropertyRule`
- `qi::rule< Iterator, EquivalenceLogicPropertyAttribute(), qi::space_type > equivalenceLogicPropertyRule`
- `qi::rule< Iterator, UntilLogicPropertyAttribute(), qi::space_type > untilLogicPropertyRule`
- `qi::rule< Iterator, PrimaryNumericMeasureAttribute(), qi::space_type > primaryNumericMeasureRule`
- `qi::rule< Iterator, UnaryNumericNumericAttribute(), qi::space_type > unaryNumericNumericRule`
- `qi::rule< Iterator, BinaryNumericNumericAttribute(), qi::space_type > binaryNumericNumericRule`

- [qi::rule< Iterator, ComparatorAttribute\(\), qi::space\\_type > probabilisticLogicPropertyComparatorRule](#)
- [SimilarityMeasureTypeParser similarityMeasureTypeParser](#)
- [ComparatorNonEqualTypeParser comparatorNonEqualTypeParser](#)

#### 6.94.1 Detailed Description

```
template<typename Iterator>class multiscale::verification::LogicPropertyGrammar< Iterator >
```

The grammar for parsing logic properties.

Definition at line 29 of file LogicPropertyGrammar.hpp.

#### 6.94.2 Constructor & Destructor Documentation

```
6.94.2.1 template<typename Iterator > multiscale::verification::Logic-
    PropertyGrammar< Iterator >::LogicPropertyGrammar (
    )
```

Definition at line 27 of file LogicPropertyGrammarDefinition.hpp.

References multiscale::verification::LogicPropertyGrammar< Iterator >::initialise().

#### 6.94.3 Member Function Documentation

```
6.94.3.1 template<typename Iterator > void multiscale::verification::Logic-
    PropertyGrammar< Iterator >::assignNamesToComparatorRules( )
    [private]
```

Assign names to the comparator rules.

Definition at line 288 of file LogicPropertyGrammarDefinition.hpp.

```
6.94.3.2 template<typename Iterator > void multiscale::verification::LogicProperty-
    Grammar< Iterator >::assignNamesToComposedLogicPropertyRules( )
    [private]
```

Assign names to the composed logic property rules.

Definition at line 272 of file LogicPropertyGrammarDefinition.hpp.

```
6.94.3.3 template<typename Iterator > void multiscale::verification::LogicProperty-
    Grammar< Iterator >::assignNamesToLogicPropertiesRules( )
    [private]
```

Assign names to logic properties rules.

Definition at line 237 of file LogicPropertyGrammarDefinition.hpp.

6.94.3.4 template<typename Iterator > void multiscale::verification::LogicPropertyGrammar< Iterator >::assignNamesToLogicPropertyRules( )  
[private]

Assign names to the logic property rules.

Definition at line 253 of file LogicPropertyGrammarDefinition.hpp.

6.94.3.5 template<typename Iterator > void multiscale::verification::LogicPropertyGrammar< Iterator >::assignNamesToPrimaryLogicPropertyRules( )  
[private]

Assign names to the primary logic property rules.

Definition at line 259 of file LogicPropertyGrammarDefinition.hpp.

6.94.3.6 template<typename Iterator > void multiscale::verification::LogicPropertyGrammar< Iterator >::assignNamesToProbabilisticLogicPropertyRules( ) [private]

Assign names to the probabilistic logic property rules.

Definition at line 246 of file LogicPropertyGrammarDefinition.hpp.

6.94.3.7 template<typename Iterator > void multiscale::verification::LogicPropertyGrammar< Iterator >::assignNamesToRules( )  
[private]

Assign names to the rules.

Definition at line 229 of file LogicPropertyGrammarDefinition.hpp.

6.94.3.8 template<typename Iterator > void multiscale::verification::LogicPropertyGrammar< Iterator >::assignNamesToSimilarityMeasureRules( )  
[private]

Assign names to the similarity measure rules.

Definition at line 282 of file LogicPropertyGrammarDefinition.hpp.

6.94.3.9 template<typename Iterator > void multiscale::verification::LogicPropertyGrammar< Iterator >::initialise( )  
[private]

Initialisation function.

Definition at line 37 of file LogicPropertyGrammarDefinition.hpp.

Referenced by multiscale::verification::LogicPropertyGrammar< Iterator >::LogicPropertyGrammar().

**6.94.3.10 template<typename Iterator> void multiscale::verification::LogicPropertyGrammar< Iterator >::initialiseComparatorRuleDebugging( ) [private]**

Initialise debugging for the comparator rule.

Definition at line 353 of file LogicPropertyGrammarDefinition.hpp.

**6.94.3.11 template<typename Iterator> void multiscale::verification::LogicPropertyGrammar< Iterator >::initialiseComparatorRules( ) [private]**

Initialise the comparator rules.

Definition at line 213 of file LogicPropertyGrammarDefinition.hpp.

**6.94.3.12 template<typename Iterator> void multiscale-::verification::LogicPropertyGrammar< Iterator >::initialiseComposedLogicPropertyErrorHandlingSupport( ) [private]**

Initialise the compose logic property error handling support.

Definition at line 423 of file LogicPropertyGrammarDefinition.hpp.

References multiscale::verification::handleUnexpectedTokenError.

**6.94.3.13 template<typename Iterator> void multiscale::verification::LogicPropertyGrammar< Iterator >::initialiseComposedLogicPropertyRule( ) [private]**

Initialise the composed logic property rule.

Definition at line 177 of file LogicPropertyGrammarDefinition.hpp.

**6.94.3.14 template<typename Iterator> void multiscale::verification::LogicPropertyGrammar< Iterator >::initialiseComposedLogicPropertyRuleDebugging( ) [private]**

Initialise debugging for the composed logic property rule.

Definition at line 337 of file LogicPropertyGrammarDefinition.hpp.

6.94.3.15 **template<typename Iterator > void multiscale::verification::LogicPropertyGrammar< Iterator >::initialiseDebugSupport( )**  
[private]

Initialise debug support.

Definition at line 220 of file LogicPropertyGrammarDefinition.hpp.

6.94.3.16 **template<typename Iterator > void multiscale::verification::LogicPropertyGrammar< Iterator >::initialiseErrorHandlingSupport( )**  
[private]

Initialise the error handling routines.

Definition at line 359 of file LogicPropertyGrammarDefinition.hpp.

6.94.3.17 **template<typename Iterator > void multiscale::verification::LogicPropertyGrammar< Iterator >::initialiseGrammar( )**  
[private]

Initialise the grammar.

Definition at line 45 of file LogicPropertyGrammarDefinition.hpp.

6.94.3.18 **template<typename Iterator > void multiscale::verification::LogicPropertyGrammar< Iterator >::initialiseLogicPropertiesErrorHandlingSupport( )**  
[private]

Initialise the logic properties error handling support.

Definition at line 365 of file LogicPropertyGrammarDefinition.hpp.

6.94.3.19 **template<typename Iterator > void multiscale::verification::LogicPropertyGrammar< Iterator >::initialiseLogicPropertiesRules( )**  
[private]

Initialise the logic properties rules.

Definition at line 53 of file LogicPropertyGrammarDefinition.hpp.

6.94.3.20 **template<typename Iterator > void multiscale::verification::LogicPropertyGrammar< Iterator >::initialiseLogicPropertiesRulesDebugging( )**  
[private]

Initialise the debugging of the logic properties rules.

Definition at line 302 of file LogicPropertyGrammarDefinition.hpp.

**6.94.3.21 template<typename Iterator > void multiscale::verification::LogicPropertyGrammar< Iterator >::initialiseLogicPropertyRule( ) [private]**

Initialise the logic property rule.

Definition at line 80 of file LogicPropertyGrammarDefinition.hpp.

**6.94.3.22 template<typename Iterator > void multiscale::verification::LogicPropertyGrammar< Iterator >::initialiseLogicPropertyRuleDebugging( ) [private]**

Initialise debugging for the logic property rule.

Definition at line 318 of file LogicPropertyGrammarDefinition.hpp.

**6.94.3.23 template<typename Iterator > void multiscale::verification::LogicPropertyGrammar< Iterator >::initialisePrimaryLogicPropertyErrorHandlingSupport( ) [private]**

Initialise the primary logic property error handling support.

Definition at line 386 of file LogicPropertyGrammarDefinition.hpp.

References multiscale::verification::handleUnexpectedTokenError.

**6.94.3.24 template<typename Iterator > void multiscale::verification::LogicPropertyGrammar< Iterator >::initialisePrimaryLogicPropertyRule( ) [private]**

Initialise the primary logic property rule.

Definition at line 94 of file LogicPropertyGrammarDefinition.hpp.

**6.94.3.25 template<typename Iterator > void multiscale::verification::LogicPropertyGrammar< Iterator >::initialisePrimaryLogicPropertyRuleDebugging( ) [private]**

Initialise debugging for the primary logic property rule.

Definition at line 324 of file LogicPropertyGrammarDefinition.hpp.

**6.94.3.26 template<typename Iterator > void multiscale::verification::LogicPropertyGrammar< Iterator >::initialiseProbabilisticLogicPropertyErrorHandlingSupport( ) [private]**

Initialise the probabilistic logic property error handling support.

Definition at line 373 of file LogicPropertyGrammarDefinition.hpp.

References multiscale::verification::handleProbabilityError, and multiscale::verification::handleUnexpectedTokenError.

```
6.94.3.27 template<typename Iterator> void multiscale::verification::LogicProperty-
Grammar<Iterator>::initialiseProbabilisticLogicPropertyRule( )
[private]
```

Initialise the probabilistic logic property rule.

Definition at line 62 of file LogicPropertyGrammarDefinition.hpp.

```
6.94.3.28 template<typename Iterator> void multiscale-
::verification::LogicPropertyGrammar<Iterator
>::initialiseProbabilisticLogicPropertyRuleDebugging( )
[private]
```

Initialise debugging for the probabilistic logic property rule.

Definition at line 311 of file LogicPropertyGrammarDefinition.hpp.

```
6.94.3.29 template<typename Iterator> void multiscale::verification::Logic-
PropertyGrammar<Iterator>::initialiseRulesDebugging( )
[private]
```

Initialise the debugging of rules.

Definition at line 294 of file LogicPropertyGrammarDefinition.hpp.

```
6.94.3.30 template<typename Iterator> void multiscale::verification::LogicProperty-
Grammar<Iterator>::initialiseSimilarityMeasureRuleDebugging( )
[private]
```

Initialise debugging for the similarity measure rule.

Definition at line 347 of file LogicPropertyGrammarDefinition.hpp.

```
6.94.3.31 template<typename Iterator> void multiscale::verification::Logic-
PropertyGrammar<Iterator>::initialiseSimilarityMeasureRules( )
[private]
```

Initialise the similarity measure rules.

Definition at line 206 of file LogicPropertyGrammarDefinition.hpp.

#### 6.94.4 Member Data Documentation

---

**6.94.4.1 template<typename Iterator> qi::rule<Iterator, AndLogicPropertyAttribute(), qi::space\_type> multiscale::verification::LogicPropertyGrammar< Iterator >::andLogicPropertyRule [private]**

The rule for parsing an "and" logic property

Definition at line 84 of file LogicPropertyGrammar.hpp.

**6.94.4.2 template<typename Iterator> qi::rule<Iterator, BinaryNumericNumericAttribute(), qi::space\_type> multiscale::verification::LogicPropertyGrammar< Iterator >::binaryNumericNumericRule [private]**

The rule for parsing a binary numeric numeric attribute

Definition at line 103 of file LogicPropertyGrammar.hpp.

**6.94.4.3 template<typename Iterator> ChangeMeasureGrammar<Iterator> multiscale::verification::LogicPropertyGrammar< Iterator >::changeMeasureRule [private]**

The grammar for parsing a change measure

Definition at line 45 of file LogicPropertyGrammar.hpp.

**6.94.4.4 template<typename Iterator> qi::rule<Iterator, ChangeTemporalNumericMeasureAttribute(), qi::space\_type> multiscale::verification::LogicPropertyGrammar< Iterator >::changeTemporalNumericMeasureRule [private]**

The rule for parsing a change temporal numeric measure

Definition at line 65 of file LogicPropertyGrammar.hpp.

**6.94.4.5 template<typename Iterator> ComparatorNonEqualTypeParser multiscale::verification::LogicPropertyGrammar< Iterator >::comparatorNonEqualTypeParser [private]**

The comparator type parser which does not accept the "=" symbol

Definition at line 116 of file LogicPropertyGrammar.hpp.

**6.94.4.6 template<typename Iterator> ComparatorGrammar<Iterator> multiscale::verification::LogicPropertyGrammar< Iterator >::comparatorRule [private]**

The grammar for parsing a comparator

Definition at line 47 of file LogicPropertyGrammar.hpp.

```
6.94.4.7 template<typename Iterator> qi::rule<Iterator,
    EquivalenceLogicPropertyAttribute(), qi::space_type>
    multiscale::verification::LogicPropertyGrammar< Iterator
    >::equivalenceLogicPropertyRule [private]
```

The rule for parsing an "equivalence" logic property

Definition at line 91 of file LogicPropertyGrammar.hpp.

```
6.94.4.8 template<typename Iterator> qi::rule<Iterator, FutureLogicPropertyAttribute(),
    qi::space_type> multiscale::verification::LogicPropertyGrammar< Iterator
    >::futureLogicPropertyRule [private]
```

The rule for parsing a "future" logic property

Definition at line 75 of file LogicPropertyGrammar.hpp.

```
6.94.4.9 template<typename Iterator> qi::rule<Iterator, GlobalLogicPropertyAttribute(),
    qi::space_type> multiscale::verification::LogicPropertyGrammar< Iterator
    >::globalLogicPropertyRule [private]
```

The rule for parsing a "global" logic property

Definition at line 77 of file LogicPropertyGrammar.hpp.

```
6.94.4.10 template<typename Iterator> qi::rule<Iterator,
    ImplicationLogicPropertyAttribute(), qi::space_type>
    multiscale::verification::LogicPropertyGrammar< Iterator
    >::implicationLogicPropertyRule [private]
```

The rule for parsing an "implication" logic property

Definition at line 88 of file LogicPropertyGrammar.hpp.

```
6.94.4.11 template<typename Iterator> qi::rule<Iterator, LogicPropertyAttribute(),
    qi::space_type> multiscale::verification::LogicPropertyGrammar< Iterator
    >::logicPropertyRule [private]
```

The rule for parsing a logic property

Definition at line 57 of file LogicPropertyGrammar.hpp.

```
6.94.4.12 template<typename Iterator> qi::rule<Iterator, NextKLogicPropertyAttribute(),
    qi::space_type> multiscale::verification::LogicPropertyGrammar< Iterator
    >::nextKLogicPropertyRule [private]
```

The rule for parsing a "next K" logic property

Definition at line 81 of file LogicPropertyGrammar.hpp.

```
6.94.4.13 template<typename Iterator> qi::rule<Iterator, NextLogicPropertyAttribute(),  
qi::space_type> multiscale::verification::LogicPropertyGrammar< Iterator  
>::nextLogicPropertyRule [private]
```

The rule for parsing a "next" logic property

Definition at line 79 of file LogicPropertyGrammar.hpp.

```
6.94.4.14 template<typename Iterator> qi::rule<Iterator, NotLogicPropertyAttribute(),  
qi::space_type> multiscale::verification::LogicPropertyGrammar< Iterator  
>::notLogicPropertyRule [private]
```

The rule for parsing a "not" logic property

Definition at line 73 of file LogicPropertyGrammar.hpp.

```
6.94.4.15 template<typename Iterator> qi::rule<Iterator, OrLogicPropertyAttribute(),  
qi::space_type> multiscale::verification::LogicPropertyGrammar< Iterator  
>::orLogicPropertyRule [private]
```

The rule for parsing an "or" logic property

Definition at line 86 of file LogicPropertyGrammar.hpp.

```
6.94.4.16 template<typename Iterator> qi::rule<Iterator,  
PrimaryLogicPropertyAttribute(), qi::space_type> multiscale::verification-  
::LogicPropertyGrammar< Iterator >::primaryLogicPropertyRule  
[private]
```

The rule for parsing a primary logic property

Definition at line 60 of file LogicPropertyGrammar.hpp.

```
6.94.4.17 template<typename Iterator> qi::rule<Iterator,  
PrimaryNumericMeasureAttribute(), qi::space_type>  
multiscale::verification::LogicPropertyGrammar< Iterator  
>::primaryNumericMeasureRule [private]
```

The rule for parsing a primary numeric numeric attribute

Definition at line 97 of file LogicPropertyGrammar.hpp.

```
6.94.4.18 template<typename Iterator> qi::rule<Iterator, ComparatorAttribute(),  
qi::space_type> multiscale::verification::LogicPropertyGrammar< Iterator  
>::probabilisticLogicPropertyComparatorRule [private]
```

The rule for parsing a comparator for a probabilistic logic property

Definition at line 107 of file LogicPropertyGrammar.hpp.

---

```
6.94.4.19 template<typename Iterator> qi::rule<Iterator,
    ProbabilisticLogicPropertyAttribute(), qi::space_type>
    multiscale::verification::LogicPropertyGrammar< Iterator
    >::probabilisticLogicPropertyRule [private]
```

The rule for parsing a probabilistic logic property

Definition at line 52 of file LogicPropertyGrammar.hpp.

```
6.94.4.20 template<typename Iterator> qi::rule<Iterator, double(), qi::space_type>
    multiscale::verification::LogicPropertyGrammar< Iterator
    >::probabilityRule [private]
```

The rule for parsing a probability value

Definition at line 55 of file LogicPropertyGrammar.hpp.

```
6.94.4.21 template<typename Iterator> qi::rule<Iterator, SimilarityMeasureAttribute(),
    qi::space_type> multiscale::verification::LogicPropertyGrammar< Iterator
    >::similarityMeasureRule [private]
```

The rule for parsing a similarity measure

Definition at line 71 of file LogicPropertyGrammar.hpp.

```
6.94.4.22 template<typename Iterator> SimilarityMeasureTypeParser
    multiscale::verification::LogicPropertyGrammar< Iterator
    >::similarityMeasureTypeParser [private]
```

The similarity measure type parser

Definition at line 113 of file LogicPropertyGrammar.hpp.

```
6.94.4.23 template<typename Iterator> qi::rule<Iterator, Similarity-
    TemporalNumericCollectionAttribute(), qi::space_type>
    multiscale::verification::LogicPropertyGrammar< Iterator
    >::similarityTemporalNumericCollectionRule [private]
```

The rule for parsing a similarity temporal numeric collection attribute

Definition at line 68 of file LogicPropertyGrammar.hpp.

```
6.94.4.24 template<typename Iterator> TemporalNumericCollectionGrammar<-
    Iterator> multiscale::verification::LogicPropertyGrammar< Iterator
    >::temporalNumericCollectionRule [private]
```

The grammar for parsing a temporal numeric collection

Definition at line 41 of file LogicPropertyGrammar.hpp.

```
6.94.4.25 template<typename Iterator> qi::rule<Iterator, Temporal-
NumericComparisonAttribute(), qi::space_type>
multiscale::verification::LogicPropertyGrammar< Iterator
>::temporalNumericComparisonRule [private]
```

The rule for parsing a temporal numeric comparison

Definition at line 62 of file LogicPropertyGrammar.hpp.

```
6.94.4.26 template<typename Iterator> TemporalNumericMeasureGrammar<Iterator>
multiscale::verification::LogicPropertyGrammar< Iterator
>::temporalNumericMeasureRule [private]
```

The grammar for parsing a temporal numeric measure

Definition at line 37 of file LogicPropertyGrammar.hpp.

```
6.94.4.27 template<typename Iterator> qi::rule<Iterator,
UnaryNumericNumericAttribute(), qi::space_type>
multiscale::verification::LogicPropertyGrammar< Iterator
>::unaryNumericNumericRule [private]
```

The rule for parsing a unary numeric numeric attribute

Definition at line 100 of file LogicPropertyGrammar.hpp.

```
6.94.4.28 template<typename Iterator> qi::rule<Iterator, UntilLogicPropertyAttribute(),
qi::space_type> multiscale::verification::LogicPropertyGrammar< Iterator
>::untilLogicPropertyRule [private]
```

The rule for parsing an "until" logic property

Definition at line 94 of file LogicPropertyGrammar.hpp.

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

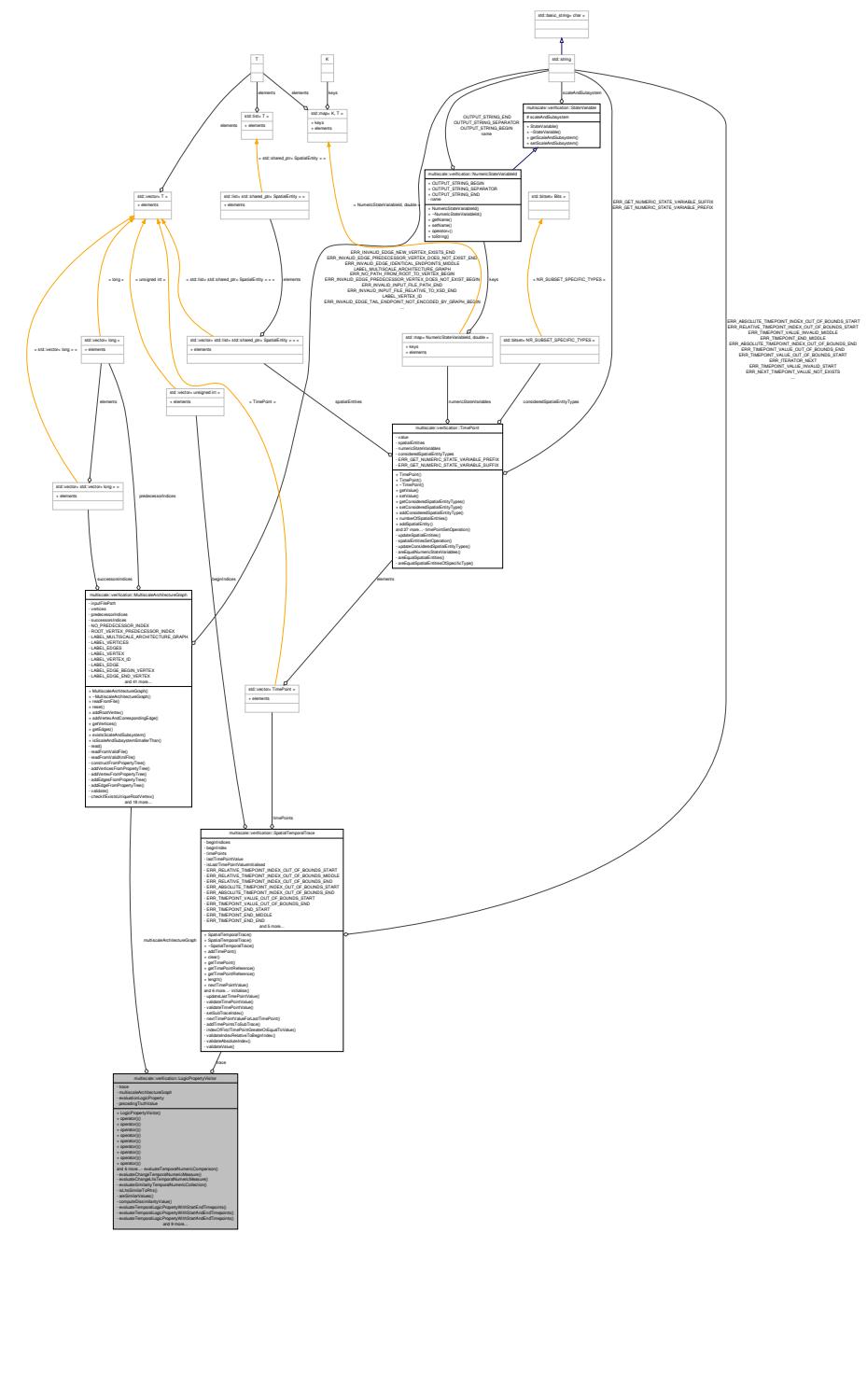
- LogicPropertyGrammar.hpp
- LogicPropertyGrammarDefinition.hpp

## 6.95 multiscale::verification::LogicPropertyVisitor Class Reference

Class used to evaluate logic properties.

```
#include <LogicPropertyVisitor.hpp>
```

## Collaboration diagram for multiscale::verification::LogicPropertyVisitor:



## Public Member Functions

- `LogicPropertyVisitor (SpatialTemporalTrace &trace, const MultiscaleArchitectureGraph &multiscaleArchitectureGraph, bool precedingTruthValue=true)`
- template<typename T>  
`bool operator() (const Nil &logicProperty, const T &lhsLogicProperty) const`  
*Overloading the "(" operator for the `Nil` alternative.*
- template<typename T>  
`bool operator() (const LogicPropertyAttribute &logicProperty, const T &lhsLogicProperty) const`  
*Overloading the "(" operator for the `LogicPropertyAttribute` alternative.*
- template<typename T>  
`bool operator() (const OrLogicPropertyAttribute &logicProperty, const T &lhsLogicProperty) const`  
*Overloading the "(" operator for the `OrLogicPropertyAttribute` alternative.*
- template<typename T>  
`bool operator() (const AndLogicPropertyAttribute &logicProperty, const T &lhsLogicProperty) const`  
*Overloading the "(" operator for the `AndLogicPropertyAttribute` alternative.*
- template<typename T>  
`bool operator() (const ImplicationLogicPropertyAttribute &logicProperty, const T &lhsLogicProperty) const`  
*Overloading the "(" operator for the `ImplicationLogicPropertyAttribute` alternative.*
- template<typename T>  
`bool operator() (const EquivalenceLogicPropertyAttribute &logicProperty, const T &lhsLogicProperty) const`  
*Overloading the "(" operator for the `EquivalenceLogicPropertyAttribute` alternative.*
- template<typename T>  
`bool operator() (const UntilLogicPropertyAttribute &logicProperty, const T &lhsLogicProperty) const`  
*Overloading the operator "(" for the `UntilLogicPropertyAttribute` alternative.*
- template<typename T>  
`bool operator() (const PrimaryLogicPropertyAttribute &logicProperty, const T &lhsLogicProperty) const`  
*Overloading the "(" operator for the `PrimaryLogicPropertyAttribute` alternative.*
- template<typename T>  
`bool operator() (const TemporalNumericComparisonAttribute &primaryLogicProperty, const T &lhsLogicProperty) const`  
*Overloading the "(" operator for the `TemporalNumericComparisonAttribute` alternative.*
- template<typename T>  
`bool operator() (const ChangeTemporalNumericMeasureAttribute &primaryLogicProperty, const T &lhsLogicProperty) const`  
*Overloading the "(" operator for the `ChangeTemporalNumericMeasureAttribute` alternative.*
- template<typename T>  
`bool operator() (const SimilarityTemporalNumericCollectionAttribute &primaryLogicProperty, const T &lhsLogicProperty) const`

- Overloading the "(" operator for the `SimilarityTemporalNumericCollectionAttribute` alternative.
- template<typename T >  
`bool operator()` (const `NotLogicPropertyAttribute` &primaryLogicProperty, const T &lhsLogicProperty) const  
*Overloading the "(" operator for the `NotLogicPropertyAttribute` alternative.*
- template<typename T >  
`bool operator()` (const `FutureLogicPropertyAttribute` &primaryLogicProperty, const T &lhsLogicProperty) const  
*Overloading the "(" operator for the `FutureLogicPropertyAttribute` alternative.*
- template<typename T >  
`bool operator()` (const `GlobalLogicPropertyAttribute` &primaryLogicProperty, const T &lhsLogicProperty) const  
*Overloading the "(" operator for the `GlobalLogicPropertyAttribute` alternative.*
- template<typename T >  
`bool operator()` (const `NextLogicPropertyAttribute` &primaryLogicProperty, const T &lhsLogicProperty) const  
*Overloading the "(" operator for the `NextLogicPropertyAttribute` alternative.*
- template<typename T >  
`bool operator()` (const `NextKLogicPropertyAttribute` &primaryLogicProperty, const T &lhsLogicProperty) const  
*Overloading the "(" operator for the `NextKLogicPropertyAttribute` alternative.*

## Private Member Functions

- template<typename T >  
`bool evaluateTemporalNumericComparison` (const `TemporalNumericComparisonAttribute` &comparisonAttribute, const T &lhsLogicProperty) const  
*Evaluate the given `TemporalNumericComparisonAttribute`.*
- template<typename T >  
`bool evaluateChangeTemporalNumericMeasure` (const `ChangeTemporalNumericMeasureAttribute` &changeAttribute, const T &lhsLogicProperty) const  
*Evaluate the given `ChangeTemporalNumericMeasureAttribute`.*
- double `evaluateChangeLhsTemporalNumericMeasure` (const `ChangeTemporalNumericMeasureAttribute` &changeAttribute) const  
*Evaluate the left hand side temporal numeric measure of the given `ChangeTemporalNumericMeasure`.*
- bool `evaluateSimilarityTemporalNumericCollection` (const `SimilarityMeasureType` &similarityMeasureType, const std::vector< double > &lhsTemporalNumericCollectionValues, const std::vector< double > &rhsTemporalNumericCollectionValues, double toleratedSimilarityDifference) const  
*Evaluate the given `SimilarityTemporalNumericCollectionAttribute`.*
- bool `isLhsSimilarToRhs` (const std::vector< double > &lhsTemporalNumericCollectionValues, const std::vector< double > &rhsTemporalNumericCollectionValues, double toleratedSimilarityDifference, const `SimilarityMeasureType` &similarityMeasureType) const

*Check if the left- and right-hand side collections of values are similar.*

- bool `areSimilarValues` (double lhsValue, double rhsValue, double toleratedSimilarityDifference, const `SimilarityMeasureType` &similarityMeasureType) const

*Check if two values are similar considering the given similarity measure type.*

- double `computeDissimilarityValue` (double lhsValue, double rhsValue, const `SimilarityMeasureType` &similarityMeasureType) const

*Compute the dissimilarity between two values considering the given similarity measure type.*

- template<typename T, typename U>  
bool `evaluateTemporalLogicPropertyWithStartEndTimepoints` (const T &temporalLogicProperty, const U &lhsLogicProperty) const

*Evaluate the given temporal logic property having start and end timepoints.*

- template<typename U>  
bool `evaluateTemporalLogicPropertyWithStartAndEndTimepoints` (const `UntilLogicPropertyAttribute` &temporalLogicProperty, const U &lhsLogicProperty, unsigned long startTime, unsigned long endTime) const

*Evaluate the given temporal logic property having start and end timepoints.*

- template<typename U>  
bool `evaluateTemporalLogicPropertyWithStartAndEndTimepoints` (const `FutureLogicPropertyAttribute` &temporalLogicProperty, const U &lhsLogicProperty, unsigned long startTime, unsigned long endTime) const

*Evaluate the given temporal logic property having start and end timepoints.*

- template<typename U>  
bool `evaluateTemporalLogicPropertyWithStartAndEndTimepoints` (const `GlobalLogicPropertyAttribute` &temporalLogicProperty, const U &lhsLogicProperty, unsigned long startTime, unsigned long endTime) const

*Evaluate the given temporal logic property having start and end timepoints.*

- template<typename T>  
bool `evaluateNextLogicProperty` (const `NextLogicPropertyAttribute` &nextLogicProperty, const T &lhsLogicProperty) const

*Evaluate the given `NextLogicPropertyAttribute`.*

- template<typename T>  
bool `evaluateNextKLogicProperty` (const `NextKLogicPropertyAttribute` &nextKLogicProperty, const T &lhsLogicProperty) const

*Evaluate the given `NextKLogicPropertyAttribute`.*

- template<typename T>  
bool `evaluateNextKLogicProperty` (const `LogicPropertyAttributeType` &logicProperty, const T &lhsLogicProperty, unsigned long kValue) const

*Evaluate the given `NextKLogicPropertyAttribute`.*

- bool `evaluate` (const `LogicPropertyAttributeType` &logicProperty, `SpatialTemporalTrace` &trace) const

*Evaluate the logic property considering the given spatial temporal trace.*

- bool `evaluate` (const `PrimaryLogicPropertyAttributeType` &primaryLogicProperty, `SpatialTemporalTrace` &trace) const

*Evaluate the logic property considering the given spatial temporal trace.*

- bool `evaluateNextLogicProperties` (const `LogicPropertyAttribute` &logicProperty, bool truthValue) const  
*Evaluate the next logic properties.*
- `LogicPropertyAttribute constructEvaluationLogicProperty` (const `LogicPropertyAttribute` &logicProperty, const std::vector< `LogicPropertyAttributeType` > &evaluationLogicProperties) const  
*Construct a new logic property attribute using the evaluation logic properties.*
- double `evaluateTemporalNumericMeasure` (const `TemporalNumericMeasureType` &temporalNumericMeasure, `SpatialTemporalTrace` &trace, unsigned int timePointIndex=0) const  
*Evaluate the temporal numeric measure considering the given spatial temporal trace.*
- bool `printExceptionMessage` (const std::string &message) const  
*Print a warning message regarding the exception and return false.*

### Private Attributes

- `SpatialTemporalTrace` & `trace`
- const `MultiscaleArchitectureGraph` & `multiscaleArchitectureGraph`
- `LogicPropertyAttributeType` `evaluationLogicProperty`
- bool `precedingTruthValue`

#### 6.95.1 Detailed Description

Class used to evaluate logic properties.

Definition at line 24 of file LogicPropertyVisitor.hpp.

#### 6.95.2 Constructor & Destructor Documentation

6.95.2.1 `multiscale::verification::LogicPropertyVisitor::LogicPropertyVisitor`  
`( SpatialTemporalTrace & trace, const MultiscaleArchitectureGraph &`  
`multiscaleArchitectureGraph, bool precedingTruthValue = true ) [inline]`

Definition at line 40 of file LogicPropertyVisitor.hpp.

Referenced by `evaluate()`, and `evaluateNextLogicProperties()`.

#### 6.95.3 Member Function Documentation

6.95.3.1 `bool multiscale::verification::LogicPropertyVisitor::areSimilarValues`  
`( double lhsValue, double rhsValue, double toleratedSimilarityDifference, const`  
`SimilarityMeasureType & similarityMeasureType ) const [inline,`  
`private]`

Check if two values are similar considering the given similarity measure type.

Two values are considered similar if their dissimilarity value is smaller or equal to toleratedSimilarityDifference.

#### Parameters

<i>lhsValue</i>	The left hand side value
<i>rhsValue</i>	The right hand side value
<i>tolerated-Similarity-Difference</i>	The maximum tolerated similarity difference between two values
<i>similarity-Measure-Type</i>	The specific similarity measure type

Definition at line 458 of file LogicPropertyVisitor.hpp.

References computeDissimilarityValue(), and multiscale::Numeric::lessOrEqual().

Referenced by isLhsSimilarToRhs().

```
6.95.3.2 double multiscale::verification::LogicPropertyVisitor::compute-
DissimilarityValue ( double lhsValue, double rhsValue, const
SimilarityMeasureType & similarityMeasureType ) const [inline,
private]
```

Compute the dissimilarity between two values considering the given similarity measure type.

#### Parameters

<i>lhsValue</i>	The left hand side value
<i>rhsValue</i>	The right hand side value
<i>similarity-Measure-Type</i>	The specific similarity measure type

Definition at line 475 of file LogicPropertyVisitor.hpp.

References multiscale::ERR\_UNDEFINED\_ENUM\_VALUE.

Referenced by areSimilarValues().

```
6.95.3.3 LogicPropertyAttribute multiscale::verification::LogicPropertyVisitor-
::constructEvaluationLogicProperty ( const LogicPropertyAttribute
& logicProperty, const std::vector< LogicPropertyAttributeType > &
evaluationLogicProperties ) const [inline, private]
```

Construct a new logic property attribute using the evaluation logic properties.

**Parameters**

<i>logic-Property</i>	The logic property containing the currently evaluated logic subproperty
<i>evaluation-Logic-Properties</i>	The logic properties preceding the currently evaluated logic subproperty

Definition at line 737 of file LogicPropertyVisitor.hpp.

References multiscale::verification::LogicPropertyAttribute::firstLogicProperty.

Referenced by evaluateNextLogicProperties().

```
6.95.3.4 bool multiscale::verification::LogicPropertyVisitor::evaluate ( const
    LogicPropertyAttributeType & logicProperty, SpatialTemporalTrace & trace )
    const [inline, private]
```

Evaluate the logic property considering the given spatial temporal trace.

**Parameters**

<i>logic-Property</i>	The logic property
<i>trace</i>	The given spatial temporal trace

Definition at line 669 of file LogicPropertyVisitor.hpp.

References evaluationLogicProperty, LogicPropertyVisitor(), and multiscaleArchitectureGraph.

Referenced by evaluateChangeLhsTemporalNumericMeasure(), evaluateChangeTemporalNumericMeasure(), evaluateNextKLogicProperty(), evaluateTemporalLogicPropertyWithStartAndEndTimepoints(), evaluateTemporalNumericComparison(), and operator()().

```
6.95.3.5 bool multiscale::verification::LogicPropertyVisitor::evaluate (
    const PrimaryLogicPropertyAttributeType & primaryLogicProperty,
    SpatialTemporalTrace & trace ) const [inline, private]
```

Evaluate the logic property considering the given spatial temporal trace.

**Parameters**

<i>primary-Logic-Property</i>	The primary logic property
<i>trace</i>	The given spatial temporal trace

Definition at line 685 of file LogicPropertyVisitor.hpp.

References `evaluationLogicProperty`, `LogicPropertyVisitor()`, and `multiscaleArchitectureGraph`.

```
6.95.3.6 double multiscale::verification::LogicPropertyVisitor-
           ::evaluateChangeLhsTemporalNumericMeasure ( const
           ChangeTemporalNumericMeasureAttribute & changeAttribute ) const
           [inline, private]
```

Evaluate the left hand side temporal numeric measure of the given `ChangeTemporalNumericMeasure`.

#### Parameters

<i>change-Attribute</i>	The change temporal numeric measure attribute
-------------------------	---

Definition at line 355 of file `LogicPropertyVisitor.hpp`.

References `multiscale::verification::ChangeTemporalNumericMeasureAttribute::changeMeasure`, `multiscale::verification::ChangeMeasureAttribute::changeMeasureType`, `evaluate()`, `evaluateTemporalNumericMeasure()`, `multiscale::verification::SpatialTemporalTrace::getTimePointReference()`, `multiscale::verification::TimePoint::getValue()`, `multiscale::verification::ChangeTemporalNumericMeasureAttribute::lhsTemporalNumericMeasure`, and `trace`.

Referenced by `evaluateChangeTemporalNumericMeasure()`.

```
6.95.3.7 template<typename T> bool multiscale::verification::LogicProperty-
           Visitor::evaluateChangeTemporalNumericMeasure ( const
           ChangeTemporalNumericMeasureAttribute & changeAttribute, const T &
           lhsLogicProperty ) const [inline, private]
```

Evaluate the given `ChangeTemporalNumericMeasureAttribute`.

#### Parameters

<i>change-Attribute</i>	The change temporal numeric measure attribute
<i>lhsLogic-Property</i>	The left hand side logic property

Definition at line 338 of file `LogicPropertyVisitor.hpp`.

References `multiscale::verification::ChangeTemporalNumericMeasureAttribute::comparator`, `multiscale::verification::ComparatorAttribute::comparatorType`, `evaluate()`, `evaluateChangeLhsTemporalNumericMeasure()`, `evaluateTemporalNumericMeasure()`, `multiscale::verification::ChangeTemporalNumericMeasureAttribute::rhsTemporalNumericMeasure`, and `trace`.

Referenced by `operator()()`.

---

6.95.3.8 template<typename T> bool multiscale::verification::LogicPropertyVisitor::evaluateNextKLogicProperty ( const NextKLogicPropertyAttribute & *nextKLogicProperty*, const T & *lhsLogicProperty* ) const [inline, private]

Evaluate the given [NextKLogicPropertyAttribute](#).

**Parameters**

<i>nextKLogic-Property</i>	The next "k" logic property
<i>lhsLogic-Property</i>	The left hand side logic property

Definition at line 633 of file LogicPropertyVisitor.hpp.

References multiscale::verification::NextKLogicPropertyAttribute::logicProperty, multiscale::verification::NextKLogicPropertyAttribute::nrOfTimepointsAhead, and trace.

Referenced by evaluateNextLogicProperty(), and operator()().

6.95.3.9 template<typename T> bool multiscale::verification::LogicPropertyVisitor::evaluateNextKLogicProperty ( const LogicPropertyAttributeType & *logicProperty*, const T & *lhsLogicProperty*, unsigned long *kValue* ) const [inline, private]

Evaluate the given [NextKLogicPropertyAttribute](#).

**Parameters**

<i>logic-Property</i>	The logic property enclosed by the next "k" logic property
<i>lhsLogic-Property</i>	The left hand side logic property
<i>kValue</i>	The value of "k"

Definition at line 646 of file LogicPropertyVisitor.hpp.

References multiscale::verification::SpatialTemporalTrace::advanceTraceBeginIndex(), evaluate(), multiscale::verification::SpatialTemporalTrace::restoreSubTraceBeginIndex(), multiscale::verification::SpatialTemporalTrace::storeSubTraceBeginIndex(), and trace.

6.95.3.10 bool multiscale::verification::LogicPropertyVisitor::evaluateNextLogicProperties ( const LogicPropertyAttribute & *logicProperty*, bool *truthValue* ) const [inline, private]

Evaluate the next logic properties.

Evaluate the next logic properties considering the given logic property, spatial temporal trace and truth value

**Parameters**

<i>logic- Property</i>	The given logic property
<i>truthValue</i>	The given truth value

Definition at line 704 of file LogicPropertyVisitor.hpp.

References `constructEvaluationLogicProperty()`, `LogicPropertyVisitor()`, `multiscale-ArchitectureGraph`, `multiscale::verification::LogicPropertyAttribute::nextLogicProperties`, and `trace`.

Referenced by `operator()()`.

**6.95.3.11 template<typename T > bool multiscale::verification::LogicPropertyVisitor-  
::evaluateNextLogicProperty ( const NextLogicPropertyAttribute &  
nextLogicProperty, const T & lhsLogicProperty ) const [inline, private]**

Evaluate the given `NextLogicPropertyAttribute`.

**Parameters**

<i>nextLogic- Property</i>	The next logic property
<i>lhsLogic- Property</i>	The left hand side logic property

Definition at line 622 of file LogicPropertyVisitor.hpp.

References `evaluateNextKLogicProperty()`, `multiscale::verification::NextLogicProperty-Attribute::logicProperty`, and `trace`.

Referenced by `operator()()`.

**6.95.3.12 bool multiscale::verification::LogicPropertyVisitor-  
::evaluateSimilarityTemporalNumericCollection ( const  
SimilarityMeasureType & similarityMeasureType, const std::vector< double  
> & lhsTemporalNumericCollectionValues, const std::vector< double > &  
rhsTemporalNumericCollectionValues, double toleratedSimilarityDifference ) const  
[inline, private]**

Evaluate the given `SimilarityTemporalNumericCollectionAttribute`.

**Parameters**

<i>similarity- Measure- Type</i>	The specific similarity measure type
<i>lhsTemporal- Numeric- Collection- Values</i>	The left hand side temporal numeric collection values

<i>rhs-Temporal-Numeric-Collection-Values</i>	The right hand side temporal numeric collection values
<i>tolerated-Similarity-Difference</i>	The maximum tolerated similarity difference between two values

Definition at line 390 of file LogicPropertyVisitor.hpp.

References [isLhsSimilarToRhs\(\)](#).

Referenced by [operator\(\)\(\)](#).

```
6.95.3.13 template<typename U> bool multiscale::verification::LogicPropertyVisitor-  
::evaluateTemporalLogicPropertyWithStartAndEndTimepoints (  
const UntilLogicPropertyAttribute & temporalLogicProperty, const U &  
lhsLogicProperty, unsigned long startTime, unsigned long endTime ) const  
[inline, private]
```

Evaluate the given temporal logic property having start and end timepoints.

The considered temporal logic property corresponds to the Until temporal operator.

#### Parameters

<i>temporal-Logic-Property</i>	The given temporal logic property (corresponding to the Until temporal operator)
<i>lhsLogic-Property</i>	The left hand side logic property
<i>startTime</i>	The considered start timepoint
<i>endTime</i>	The considered end timepoint

Definition at line 534 of file LogicPropertyVisitor.hpp.

References [evaluate\(\)](#), [multiscale::verification::UntilLogicPropertyAttribute::logic-Property](#), [multiscale::verification::SpatialTemporalTrace::nextTimePointValue\(\)](#), [multiscale::verification::SpatialTemporalTrace::setSubTraceWithTimepointsValuesGreaterOr-EqualTo\(\)](#), and [trace](#).

Referenced by [evaluateTemporalLogicPropertyWithStartEndTimepoints\(\)](#).

```
6.95.3.14 template<typename U > bool multiscale::verification::LogicProperty-
Visitor::evaluateTemporalLogicPropertyWithStartAndEndTimepoints
( const FutureLogicPropertyAttribute & temporalLogicProperty, const U &
lhsLogicProperty, unsigned long startTime, unsigned long endTime ) const
[inline, private]
```

Evaluate the given temporal logic property having start and end timepoints.

The considered temporal logic property corresponds to the Future temporal operator.

#### Parameters

<i>temporal- Logic- Property</i>	The given temporal logic property (corresponding to the Future tempo- ral operator)
<i>lhsLogic- Property</i>	The left hand side logic property
<i>startTime</i>	The considered start timepoint
<i>endTime</i>	The considered end timepoint

Definition at line 568 of file LogicPropertyVisitor.hpp.

References `evaluate()`, `multiscale::verification::FutureLogicPropertyAttribute::logic-  
Property`, `multiscale::verification::SpatialTemporalTrace::nextTimePointValue()`, `multiscale-  
::verification::SpatialTemporalTrace::setSubTraceWithTimepointsValuesGreaterOr-  
EqualTo()`, and `trace`.

```
6.95.3.15 template<typename U > bool multiscale::verification::LogicProperty-
Visitor::evaluateTemporalLogicPropertyWithStartAndEndTimepoints
( const GlobalLogicPropertyAttribute & temporalLogicProperty, const U &
lhsLogicProperty, unsigned long startTime, unsigned long endTime ) const
[inline, private]
```

Evaluate the given temporal logic property having start and end timepoints.

The considered temporal logic property corresponds to the Global temporal operator.

#### Parameters

<i>temporal- Logic- Property</i>	The given temporal logic property (corresponding to the Global tempo- ral operator)
<i>lhsLogic- Property</i>	The left hand side logic property
<i>startTime</i>	The considered start timepoint
<i>endTime</i>	The considered end timepoint

Definition at line 597 of file LogicPropertyVisitor.hpp.

References `evaluate()`, `multiscale::verification::GlobalLogicPropertyAttribute::logic-  
Property`, `multiscale::verification::SpatialTemporalTrace::nextTimePointValue()`, `multiscale-`

::verification::SpatialTemporalTrace::setSubTraceWithTimepointsValuesGreaterOrEqualTo(), and trace.

**6.95.3.16 template<typename T , typename U > bool multiscale::verification::LogicPropertyVisitor::evaluateTemporalLogicPropertyWithStartEndTimepoints ( const T & *temporalLogicProperty*, const U & *IhsLogicProperty* ) const [inline, private]**

Evaluate the given temporal logic property having start and end timepoints.

The considered temporal logic properties correspond to the Future, Global and Until temporal operators.

#### Parameters

<i>temporal-Logic-Property</i>	The given temporal logic property (corresponding to either the Future, Global or Until temporal operator)
<i>IhsLogic-Property</i>	The left hand side logic property

Definition at line 503 of file LogicPropertyVisitor.hpp.

References evaluateTemporalLogicPropertyWithStartAndEndTimepoints(), multiscale::verification::SpatialTemporalTrace::restoreSubTraceBeginIndex(), multiscale::verification::SpatialTemporalTrace::storeSubTraceBeginIndex(), and trace.

Referenced by operator()().

**6.95.3.17 template<typename T > bool multiscale::verification::LogicPropertyVisitor::evaluateTemporalNumericComparison ( const TemporalNumericComparisonAttribute & *comparisonAttribute*, const T & *IhsLogicProperty* ) const [inline, private]**

Evaluate the given [TemporalNumericComparisonAttribute](#).

#### Parameters

<i>comparison-Attribute</i>	The numeric numeric comparison attribute
<i>IhsLogic-Property</i>	The left hand side logic property

Definition at line 315 of file LogicPropertyVisitor.hpp.

References multiscale::verification::TemporalNumericComparisonAttribute::comparator, multiscale::verification::ComparatorAttribute::comparatorType, evaluate(), evaluateTemporalNumericMeasure(), multiscale::verification::TemporalNumericComparisonAttribute::lhsTemporalNumericMeasure, multiscale::verification::TemporalNumericComparisonAttribute::rhsTemporalNumericMeasure, and trace.

Referenced by operator()().

```
6.95.3.18 double multiscale::verification::LogicPropertyVisitor::evaluate-
TemporalNumericMeasure ( const TemporalNumericMeasureType
& temporalNumericMeasure, SpatialTemporalTrace & trace, unsigned int
timePointIndex = 0 ) const [inline, private]
```

Evaluate the temporal numeric measure considering the given spatial temporal trace.

#### Parameters

<i>temporal-Numeric-Measure</i>	The given temporal numeric measure
<i>trace</i>	The given spatial temporal trace
<i>timePoint-Index</i>	The index of the considered starting timepoint from the trace

Definition at line 752 of file LogicPropertyVisitor.hpp.

References multiscale::verification::SpatialTemporalTrace::advanceTraceBeginIndex(), multiscaleArchitectureGraph, multiscale::verification::SpatialTemporalTrace::restoreSubTraceBeginIndex(), and multiscale::verification::SpatialTemporalTrace::storeSubTraceBeginIndex().

Referenced by evaluateChangeLhsTemporalNumericMeasure(), evaluateChangeTemporalNumericMeasure(), and evaluateTemporalNumericComparison().

```
6.95.3.19 bool multiscale::verification::LogicPropertyVisitor::isLhsSimilarToRhs
( const std::vector< double > & lhsTemporalNumericCollectionValues,
const std::vector< double > & rhsTemporalNumericCollectionValues,
double toleratedSimilarityDifference, const SimilarityMeasureType &
similarityMeasureType ) const [inline, private]
```

Check if the left- and right-hand side collections of values are similar.

#### Parameters

<i>similarity-Measure-Type</i>	The specific similarity measure type
<i>lhsTemporal-Numeric-Collection-Values</i>	The left hand side temporal numeric collection values
<i>rhs-Temporal-Numeric-Collection-Values</i>	The right hand side temporal numeric collection values

<i>tolerated-Similarity-Difference</i>	The maximum tolerated similarity difference between two values
--	--

Definition at line 419 of file LogicPropertyVisitor.hpp.

References [areSimilarValues\(\)](#).

Referenced by [evaluateSimilarityTemporalNumericCollection\(\)](#).

```
6.95.3.20 template<typename T> bool multiscale::verification::LogicPropertyVisitor::operator()
( const Nil & logicProperty, const T & lhsLogicProperty ) const [inline]
```

Overloading the "(" operator for the [Nil](#) alternative.

#### Parameters

<i>logic-Property</i>	The logic property
<i>lhsLogic-Property</i>	The left hand side logic property

Definition at line 52 of file LogicPropertyVisitor.hpp.

```
6.95.3.21 template<typename T> bool multiscale::verification::LogicPropertyVisitor::operator()
( const LogicPropertyAttribute & logicProperty, const T & lhsLogicProperty )
const [inline]
```

Overloading the "(" operator for the [LogicPropertyAttribute](#) alternative.

#### Parameters

<i>logic-Property</i>	The logic property
<i>lhsLogic-Property</i>	The left hand side logic property

Definition at line 62 of file LogicPropertyVisitor.hpp.

References [evaluate\(\)](#), [evaluateNextLogicProperties\(\)](#), [multiscale::verification::LogicPropertyAttribute::firstLogicProperty](#), and [trace](#).

```
6.95.3.22 template<typename T> bool multiscale::verification::LogicPropertyVisitor::operator()
( const OrLogicPropertyAttribute & logicProperty, const T & lhsLogicProperty )
const [inline]
```

Overloading the "(" operator for the [OrLogicPropertyAttribute](#) alternative.

Remark: Lazy evaluation is performed for efficiency purposes.

**Parameters**

<i>logic- Property</i>	The logic property
<i>lhsLogic- Property</i>	The left hand side logic property

Definition at line 76 of file LogicPropertyVisitor.hpp.

References evaluate(), multiscale::verification::OrLogicPropertyAttribute::logicProperty, precedingTruthValue, and trace.

**6.95.3.23 template<typename T > bool multiscale::verification::LogicPropertyVisitor::operator()  
( const AndLogicPropertyAttribute & *logicProperty*, const T & *lhsLogicProperty*  
 ) const [inline]**

Overloading the "(") operator for the [AndLogicPropertyAttribute](#) alternative.

Remark: Lazy evaluation is performed for efficiency purposes.

**Parameters**

<i>logic- Property</i>	The logic property
<i>lhsLogic- Property</i>	The left hand side logic property

Definition at line 93 of file LogicPropertyVisitor.hpp.

References evaluate(), multiscale::verification::AndLogicPropertyAttribute::logicProperty, precedingTruthValue, and trace.

**6.95.3.24 template<typename T > bool multiscale::verification::LogicPropertyVisitor::operator()  
( const ImplicationLogicPropertyAttribute & *logicProperty*, const T &  
 *lhsLogicProperty* ) const [inline]**

Overloading the "(") operator for the [ImplicationLogicPropertyAttribute](#) alternative.

Remark: Lazy evaluation is performed for efficiency purposes.

**Parameters**

<i>logic- Property</i>	The logic property
<i>lhsLogic- Property</i>	The left hand side logic property

Definition at line 110 of file LogicPropertyVisitor.hpp.

References evaluate(), multiscale::verification::ImplicationLogicPropertyAttribute::logicProperty, precedingTruthValue, and trace.

```
6.95.3.25 template<typename T> bool multiscale::verification::LogicPropertyVisitor::operator()
( const EquivalenceLogicPropertyAttribute & logicProperty, const T &
lhsLogicProperty ) const [inline]
```

Overloading the "(" operator for the [EquivalenceLogicPropertyAttribute](#) alternative.

#### Parameters

<i>logic- Property</i>	The logic property
<i>lhsLogic- Property</i>	The left hand side logic property

Definition at line 127 of file LogicPropertyVisitor.hpp.

References evaluate(), multiscale::verification::EquivalenceLogicPropertyAttribute::logicProperty, precedingTruthValue, and trace.

```
6.95.3.26 template<typename T> bool multiscale::verification::LogicPropertyVisitor::operator()
( const UntilLogicPropertyAttribute & logicProperty, const T & lhsLogicProperty
) const [inline]
```

Overloading the operator "(" for the [UntilLogicPropertyAttribute](#) alternative.

#### Parameters

<i>logic- Property</i>	The logic property
<i>lhsLogic- Property</i>	The left hand side logic property

Definition at line 142 of file LogicPropertyVisitor.hpp.

References evaluateTemporalLogicPropertyWithStartEndTimepoints(), printExceptionMessage(), and multiscale::MultiscaleException::rawMessage().

```
6.95.3.27 template<typename T> bool multiscale::verification::LogicPropertyVisitor::operator()
( const PrimaryLogicPropertyAttribute & logicProperty, const T &
lhsLogicProperty ) const [inline]
```

Overloading the "(" operator for the [PrimaryLogicPropertyAttribute](#) alternative.

#### Parameters

<i>logic- Property</i>	The logic property
----------------------------	--------------------

<i>IhsLogic- Property</i>	The left hand side logic property
-------------------------------	-----------------------------------

Definition at line 161 of file LogicPropertyVisitor.hpp.

References evaluate(), multiscale::verification::PrimaryLogicPropertyAttribute::primaryLogicProperty, and trace.

```
6.95.3.28 template<typename T> bool multiscale::verification::LogicPropertyVisitor::operator()
( const TemporalNumericComparisonAttribute & primaryLogicProperty,
  const T & IhsLogicProperty ) const [inline]
```

Overloading the "(" operator for the [TemporalNumericComparisonAttribute](#) alternative.

#### Parameters

<i>primary- Logic- Property</i>	The primary logic property
<i>IhsLogic- Property</i>	The left hand side logic property

Definition at line 172 of file LogicPropertyVisitor.hpp.

References evaluateTemporalNumericComparison(), printExceptionMessage(), and multiscale::MultiscaleException::rawMessage().

```
6.95.3.29 template<typename T> bool multiscale::verification::LogicPropertyVisitor::operator()
( const ChangeTemporalNumericMeasureAttribute & primaryLogicProperty,
  const T & IhsLogicProperty ) const [inline]
```

Overloading the "(" operator for the [ChangeTemporalNumericMeasureAttribute](#) alternative.

#### Parameters

<i>primary- Logic- Property</i>	The primary logic property
<i>IhsLogic- Property</i>	The left hand side logic property

Definition at line 187 of file LogicPropertyVisitor.hpp.

References evaluateChangeTemporalNumericMeasure(), printExceptionMessage(), and multiscale::MultiscaleException::rawMessage().

---

6.95.3.30 template<typename T> bool multiscale::verification::LogicPropertyVisitor::operator()  
 ( const SimilarityTemporalNumericCollectionAttribute &  
*primaryLogicProperty*, const T & *lhsLogicProperty* ) const [inline]

Overloading the "(" operator for the [SimilarityTemporalNumericCollectionAttribute](#) alternative.

**Parameters**

<i>primary- Logic- Property</i>	The primary logic property
<i>lhsLogic- Property</i>	The left hand side logic property

Definition at line 202 of file LogicPropertyVisitor.hpp.

References [evaluateSimilarityTemporalNumericCollection\(\)](#), [multiscale::verification::NumericMeasureCollectionEvaluator::evaluateTemporalNumericCollection\(\)](#), [multiscale::verification::SimilarityTemporalNumericCollectionAttribute::lhsTemporalNumericCollection](#), [multiscaleArchitectureGraph](#), [multiscale::verification::SimilarityTemporalNumericCollectionAttribute::rhsTemporalNumericCollection](#), [multiscale::verification::SimilarityTemporalNumericCollectionAttribute::similarityMeasure](#), [multiscale::verification::SimilarityMeasureAttribute::similarityMeasure](#), [multiscale::verification::SimilarityTemporalNumericCollectionAttribute::toleratedSimilarityDifference](#), and [trace](#).

6.95.3.31 template<typename T> bool multiscale::verification::LogicPropertyVisitor::operator()  
 ( const NotLogicPropertyAttribute & *primaryLogicProperty*, const T &  
*lhsLogicProperty* ) const [inline]

Overloading the "(" operator for the [NotLogicPropertyAttribute](#) alternative.

**Parameters**

<i>primary- Logic- Property</i>	The primary logic property
<i>lhsLogic- Property</i>	The left hand side logic property

Definition at line 230 of file LogicPropertyVisitor.hpp.

References [evaluate\(\)](#), [multiscale::verification::NotLogicPropertyAttribute::logicProperty](#), and [trace](#).

6.95.3.32 template<typename T> bool multiscale::verification::LogicPropertyVisitor::operator()  
 ( const FutureLogicPropertyAttribute & *primaryLogicProperty*, const T &  
*lhsLogicProperty* ) const [inline]

Overloading the "(" operator for the [FutureLogicPropertyAttribute](#) alternative.

**Parameters**

<i>primary- Logic- Property</i>	The primary logic property
<i>lhsLogic- Property</i>	The left hand side logic property

Definition at line 243 of file LogicPropertyVisitor.hpp.

References evaluateTemporalLogicPropertyWithStartEndTimepoints(), printExceptionMessage(), and multiscale::MultiscaleException::rawMessage().

```
6.95.3.33 template<typename T> bool multiscale::verification::LogicPropertyVisitor::operator()
( const GlobalLogicPropertyAttribute & primaryLogicProperty, const T &
lhsLogicProperty ) const [inline]
```

Overloading the "(" operator for the [GlobalLogicPropertyAttribute](#) alternative.

**Parameters**

<i>primary- Logic- Property</i>	The primary logic property
<i>lhsLogic- Property</i>	The left hand side logic property

Definition at line 263 of file LogicPropertyVisitor.hpp.

References evaluateTemporalLogicPropertyWithStartEndTimepoints(), printExceptionMessage(), and multiscale::MultiscaleException::rawMessage().

```
6.95.3.34 template<typename T> bool multiscale::verification::LogicPropertyVisitor::operator()
( const NextLogicPropertyAttribute & primaryLogicProperty, const T &
lhsLogicProperty ) const [inline]
```

Overloading the "(" operator for the [NextLogicPropertyAttribute](#) alternative.

**Parameters**

<i>primary- Logic- Property</i>	The primary logic property
<i>lhsLogic- Property</i>	The left hand side logic property

Definition at line 283 of file LogicPropertyVisitor.hpp.

References evaluateNextLogicProperty(), printExceptionMessage(), and multiscale::MultiscaleException::rawMessage().

---

6.95.3.35 template<typename T> bool multiscale::verification::LogicPropertyVisitor::operator()  
 ( const NextKLogicPropertyAttribute & *primaryLogicProperty*, const T &  
*lhsLogicProperty* ) const [inline]

Overloading the "(" operator for the [NextKLogicPropertyAttribute](#) alternative.

**Parameters**

<i>primary- Logic- Property</i>	The primary logic property
<i>lhsLogic- Property</i>	The left hand side logic property

Definition at line 298 of file LogicPropertyVisitor.hpp.

References [evaluateNextKLogicProperty\(\)](#), [printExceptionMessage\(\)](#), and [multiscale::MultiscaleException::rawMessage\(\)](#).

6.95.3.36 bool multiscale::verification::LogicPropertyVisitor::print-  
 ExceptionMessage ( const std::string & *message* ) const [inline,  
 private]

Print a warning message regarding the exception and return false.

**Parameters**

<i>message</i>	The exception message
----------------	-----------------------

Definition at line 778 of file LogicPropertyVisitor.hpp.

References [multiscale::ConsolePrinter::printWarningMessage\(\)](#), [multiscale::verification::WRN\\_LOGIC\\_PROPERTY\\_EVAL\\_FALSE](#), and [multiscale::verification::WRN\\_OUTPUT\\_SEPARATOR](#).

Referenced by [operator\(\)\(\)](#).

## 6.95.4 Member Data Documentation

6.95.4.1 LogicPropertyAttributeType multiscale::verification-  
 ::LogicPropertyVisitor::evaluationLogicProperty  
 [private]

The logic property used only for evaluation purposes

Definition at line 34 of file LogicPropertyVisitor.hpp.

Referenced by [evaluate\(\)](#).

6.95.4.2 `const MultiscaleArchitectureGraph& multiscale::verification-  
::LogicPropertyVisitor::multiscaleArchitectureGraph  
[private]`

The multiscale architecture graph

Definition at line 31 of file LogicPropertyVisitor.hpp.

Referenced by `evaluate()`, `evaluateNextLogicProperties()`, `evaluateTemporalNumericMeasure()`, and `operator()()`.

6.95.4.3 `bool multiscale::verification::LogicPropertyVisitor::precedingTruthValue  
[private]`

The truth value of the preceding logic property

Definition at line 36 of file LogicPropertyVisitor.hpp.

Referenced by `operator()()`.

6.95.4.4 `SpatialTemporalTrace& multiscale::verification::LogicPropertyVisitor-  
::trace [private]`

A reference to the spatial temporal trace

Definition at line 29 of file LogicPropertyVisitor.hpp.

Referenced by `evaluateChangeLhsTemporalNumericMeasure()`, `evaluateChangeTemporalNumericMeasure()`, `evaluateNextKLogicProperty()`, `evaluateNextLogicProperties()`, `evaluateNextLogicProperty()`, `evaluateTemporalLogicPropertyWithStartAndEndTimepoints()`, `evaluateTemporalLogicPropertyWithStartEndTimepoints()`, `evaluateTemporalNumericComparison()`, and `operator()()`.

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

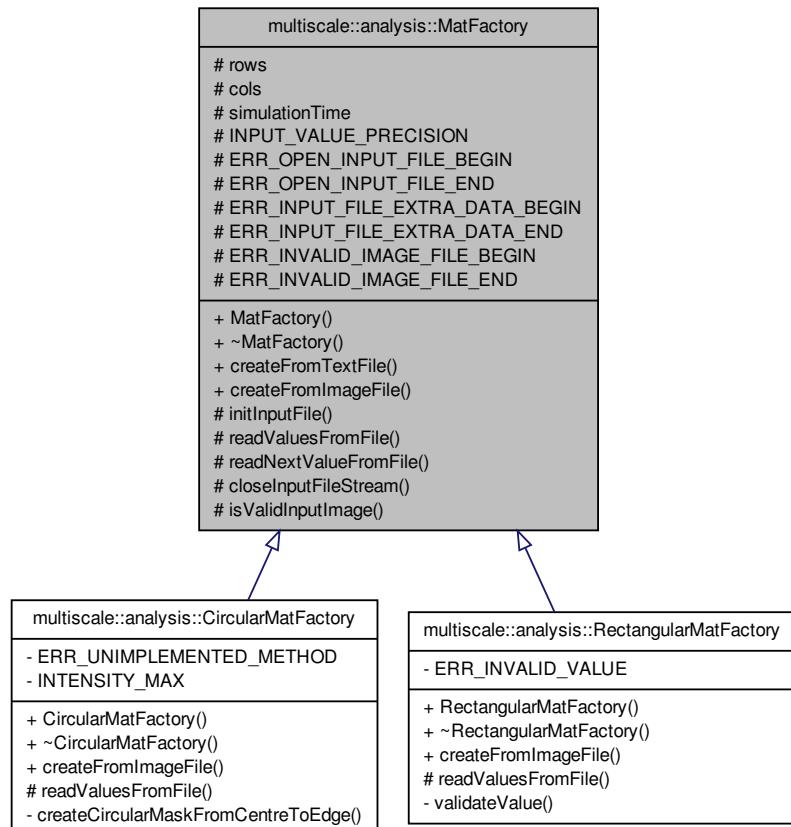
- LogicPropertyVisitor.hpp

## 6.96 multiscale::analysis::MatFactory Class Reference

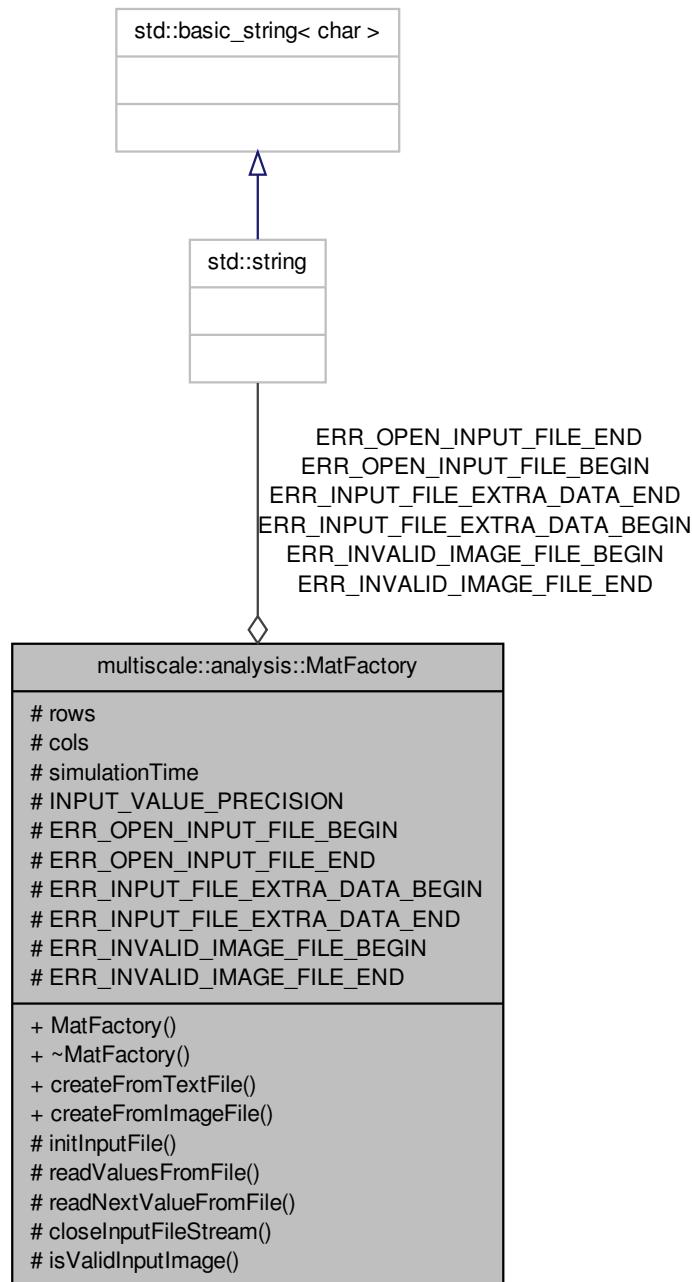
Class for creating a cv::Mat object.

```
#include <MatFactory.hpp>
```

Inheritance diagram for multiscale::analysis::MatFactory:



Collaboration diagram for multiscale::analysis::MatFactory:



## Public Member Functions

- `MatFactory ()`
- `virtual ~MatFactory ()`
- `cv::Mat createFromTextFile (const std::string &inputFilePath)`  
`Create a cv::Mat object from the input file.`
- `virtual cv::Mat createFromImageFile (const std::string &inputFile)=0`  
`Create a Mat object from the provided image file.`

## Protected Member Functions

- `void initInputFile (std::ifstream &fin, const std::string &inputFilePath)`  
`Initialise the input file.`
- `virtual void readValuesFromFile (std::ifstream &fin, cv::Mat &image)=0`  
`Read image values from the given file.`
- `void readNextValueFromFile (std::ifstream &fin, float &nextValue)`  
`Read the next image value from the provided input file stream.`
- `void closeInputStream (std::ifstream &fin, const std::string &inputFilePath)`  
`Close the input file stream.`
- `bool isValidInputImage (const cv::Mat &image, const std::string &imageFilePath)`  
`Check if the provided image represented as a Mat object is valid.`

## Protected Attributes

- `unsigned int rows`
- `unsigned int cols`
- `double simulationTime`

## Static Protected Attributes

- `static const int INPUT_VALUE_PRECISION = std::numeric_limits<float>::max_digits10`
- `static const std::string ERR_OPEN_INPUT_FILE_BEGIN = "The input file ("`
- `static const std::string ERR_OPEN_INPUT_FILE_END = ") could not be opened."`
- `static const std::string ERR_INPUT_FILE_EXTRA_DATA_BEGIN = "The input file ("`
- `static const std::string ERR_INPUT_FILE_EXTRA_DATA_END = ") contains more data than required. Please change."`
- `static const std::string ERR_INVALID_IMAGE_FILE_BEGIN = "The provided image input file ("`
- `static const std::string ERR_INVALID_IMAGE_FILE_END = ") is invalid. Please change."`

### 6.96.1 Detailed Description

Class for creating a cv::Mat object.

Definition at line 14 of file MatFactory.hpp.

### 6.96.2 Constructor & Destructor Documentation

#### 6.96.2.1 MatFactory::MatFactory( )

Definition at line 12 of file MatFactory.cpp.

#### 6.96.2.2 MatFactory::~MatFactory( ) [virtual]

Definition at line 14 of file MatFactory.cpp.

### 6.96.3 Member Function Documentation

#### 6.96.3.1 void MatFactory::closeInputStream ( std::ifstream & *fin*, const std::string & *inputFilePath* ) [protected]

Close the input file stream.

If the file contains more data than expected throw an exception.

##### Parameters

<i>fin</i>	The input file stream
<i>inputFilePath</i>	The path to the input file from which the data is read

Definition at line 60 of file MatFactory.cpp.

References ERR\_INPUT\_FILE\_EXTRA\_DATA\_BEGIN, and ERR\_INPUT\_FILE\_EXTRA\_DATA\_END.

Referenced by createFromTextFile().

#### 6.96.3.2 virtual cv::Mat multiscale::analysis::MatFactory::createFromImageFile ( const std::string & *inputFile* ) [pure virtual]

Create a Mat object from the provided image file.

Create a Mat instance from the given image file

##### Parameters

<i>inputFile</i>	The path to the image file
------------------	----------------------------

Implemented in [multiscale::analysis::CircularMatFactory](#), and [multiscale::analysis::RectangularMatFactory](#).

#### 6.96.3.3 cv::Mat MatFactory::createFromTextFile ( const std::string & *inputFilePath* )

Create a cv::Mat object from the input file.

Create the cv::Mat instance from the values given in the input file

FORMAT OF INPUT FILE:

- 1st line contains two positive integers and a real value: nr\_rows, nr\_cols and simulation\_time
- 2nd - (nr\_rows + 1)th lines contain the values of the positions in the grid

**Parameters**

<i>inputFile-Path</i>	The path to the input file
-----------------------	----------------------------

Definition at line 16 of file MatFactory.cpp.

References [closeInputStream\(\)](#), [cols](#), [initInputFile\(\)](#), [readValuesFromFile\(\)](#), and [rows](#).

#### 6.96.3.4 void MatFactory::initInputFile ( std::ifstream & *fin*, const std::string & *inputFilePath* ) [protected]

Initialise the input file.

Initialise the input file. Open an input file stream to the given input file path.

**Parameters**

<i>fin</i>	An input stream for reading data from the input file
<i>inputFile-Path</i>	The path to the input file

Definition at line 36 of file MatFactory.cpp.

References [cols](#), [ERR\\_OPEN\\_INPUT\\_FILE\\_BEGIN](#), [ERR\\_OPEN\\_INPUT\\_FILE\\_END](#), [rows](#), and [simulationTime](#).

Referenced by [createFromTextFile\(\)](#).

#### 6.96.3.5 bool MatFactory::isValidInputImage ( const cv::Mat & *image*, const std::string & *imageFilePath* ) [protected]

Check if the provided image represented as a Mat object is valid.

**Parameters**

<i>image</i>	The image represented as a Mat object
<i>imageFile-Path</i>	The path to the file from which the image was read

Definition at line 77 of file MatFactory.cpp.

References ERR\_OPEN\_INPUT\_FILE\_BEGIN, and ERR\_OPEN\_INPUT\_FILE\_END.

Referenced by multiscale::analysis::CircularMatFactory::createFromImageFile(), and multiscale::analysis::RectangularMatFactory::createFromImageFile().

**6.96.3.6 void MatFactory::readNextValueFromFile ( std::ifstream & *fin*, float & *nextValue* ) [protected]**

Read the next image value from the provided input file stream.

**Parameters**

<i>fin</i>	The input file stream from which the values are read
<i>nextValue</i>	The next image value read from the file

Definition at line 53 of file MatFactory.cpp.

References INPUT\_VALUE\_PRECISION, and multiscale::Numeric::round().

Referenced by multiscale::analysis::RectangularMatFactory::readValuesFromFile().

**6.96.3.7 virtual void multiscale::analysis::MatFactory::readValuesFromFile ( std::ifstream & *fin*, cv::Mat & *image* ) [protected, pure virtual]**

Read image values from the given file.

Read the image values (in [0, 1]) from the given file.

**Parameters**

<i>fin</i>	The input file stream from which the values are read
<i>image</i>	The image to which the values are written

Implemented in multiscale::analysis::RectangularMatFactory, and multiscale::analysis::CircularMatFactory.

Referenced by createFromTextFile().

## 6.96.4 Member Data Documentation

**6.96.4.1 unsigned int multiscale::analysis::MatFactory::cols [protected]**

Number of columns in the Mat object

Definition at line 19 of file MatFactory.hpp.

Referenced by `createFromTextFile()`, `initInputFile()`, and `multiscale::analysis::RectangularMatFactory::readValuesFromFile()`.

**6.96.4.2 const std::string MatFactory::ERR\_INPUT\_FILE\_EXTRA\_DATA\_BEGIN =**  
"The input file (" [static, protected]

Definition at line 97 of file MatFactory.hpp.

Referenced by `closeInputStream()`.

**6.96.4.3 const std::string MatFactory::ERR\_INPUT\_FILE\_EXTRA\_DATA\_END =**"  
contains more data than required. Please change." [static, protected]

Definition at line 98 of file MatFactory.hpp.

Referenced by `closeInputStream()`.

**6.96.4.4 const std::string MatFactory::ERR\_INVALID\_IMAGE\_FILE\_BEGIN =**"The  
provided image input file (" [static, protected]

Definition at line 100 of file MatFactory.hpp.

**6.96.4.5 const std::string MatFactory::ERR\_INVALID\_IMAGE\_FILE\_END =**") is invalid.  
Please change." [static, protected]

Definition at line 101 of file MatFactory.hpp.

**6.96.4.6 const std::string MatFactory::ERR\_OPEN\_INPUT\_FILE\_BEGIN =**"The input  
file (" [static, protected]

Definition at line 94 of file MatFactory.hpp.

Referenced by `initInputFile()`, and `isValidInputImage()`.

**6.96.4.7 const std::string MatFactory::ERR\_OPEN\_INPUT\_FILE\_END =**") could not be  
opened." [static, protected]

Definition at line 95 of file MatFactory.hpp.

Referenced by `initInputFile()`, and `isValidInputImage()`.

**6.96.4.8 const int MatFactory::INPUT\_VALUE\_PRECISION =**  
`std::numeric_limits<float>::max_digits10` [static, protected]

Definition at line 92 of file MatFactory.hpp.

Referenced by `readNextValueFromFile()`.

#### 6.96.4.9 `unsigned int multiscale::analysis::MatFactory::rows` [protected]

Number of rows in the Mat object

Definition at line 18 of file `MatFactory.hpp`.

Referenced by `createFromTextFile()`, `initInputFile()`, and `multiscale::analysis::RectangularMatFactory::readValuesFromFile()`.

#### 6.96.4.10 `double multiscale::analysis::MatFactory::simulationTime` [protected]

Simulation time read from the input file

Definition at line 20 of file `MatFactory.hpp`.

Referenced by `initInputFile()`.

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

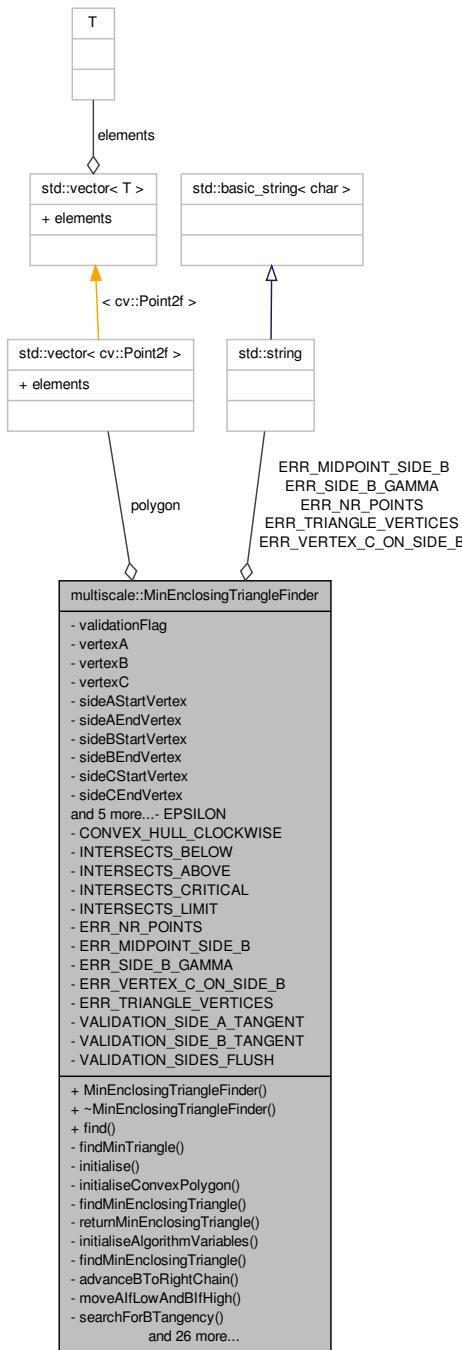
- `MatFactory.hpp`
- `MatFactory.cpp`

## 6.97 multiscale::MinEnclosingTriangleFinder Class Reference

Class for computing the minimum area enclosing triangle for a given polygon.

```
#include <MinEnclosingTriangleFinder.hpp>
```

Collaboration diagram for multiscale::MinEnclosingTriangleFinder:



## Public Member Functions

- `MinEnclosingTriangleFinder ()`
- `~MinEnclosingTriangleFinder ()`
- `double find (const std::vector< cv::Point2f > &points, std::vector< cv::Point2f > &minEnclosingTriangle)`

*Find the minimum area enclosing triangle for the given 2D point set.*

## Private Member Functions

- `double findMinTriangle (const std::vector< cv::Point2f > &points, std::vector< cv::Point2f > &minEnclosingTriangle)`

*Find the minimum area enclosing triangle for the given 2D point set.*

- `void initialise (const std::vector< cv::Point2f > &points, std::vector< cv::Point2f > &minEnclosingTriangle)`

*Initialisation function for the class.*

- `void initialiseConvexPolygon (const std::vector< cv::Point2f > &points)`

*Initialise polygon as the convex hull of the given set of points.*

- `double findMinEnclosingTriangle (const std::vector< cv::Point2f > &polygon, std::vector< cv::Point2f > &minEnclosingTriangle)`

*Find the minimum area enclosing triangle for the given polygon.*

- `double returnMinEnclosingTriangle (const std::vector< cv::Point2f > &polygon, std::vector< cv::Point2f > &minEnclosingTriangle)`

*Return the minimum area enclosing triangle in case the given polygon has at most three points.*

- `void initialiseAlgorithmVariables ()`

*Initialisation of the algorithm variables.*

- `void findMinEnclosingTriangle (std::vector< cv::Point2f > &minEnclosingTriangle, double &minEnclosingTriangleArea)`

*Find the minimum area enclosing triangle for the given polygon.*

- `void advanceBToRightChain ()`

*Advance b to the right chain.*

- `void moveAIfLowAndBIfHigh ()`

*Move "a" if it is low and "b" if it is high.*

- `void searchForBTangency ()`

*Search for the tangency of side B.*

- `bool isNotBTangency ()`

*Check if tangency for side B was not obtained.*

- `void updateSidesCA ()`

*Update sides A and C.*

- `void updateSidesBA ()`

*Update sides B and possibly A if tangency for side B was not obtained.*

- `void updateSideB ()`

*Set side B if tangency for side B was obtained.*

- bool `isLocalMinimalTriangle ()`  
*Update the triangle vertices after all sides were set and check if a local minimal triangle was found.*
- bool `isValidMinimalTriangle ()`  
*Check if the found minimal triangle is valid.*
- void `updateMinEnclosingTriangle (std::vector< cv::Point2f > &minEnclosingTriangle, double &minEnclosingTriangleArea)`  
*Update the current minimum area enclosing triangle if the newly obtained one has a smaller area.*
- bool `middlePointOfSideB (cv::Point2f &middlePointOfSideB)`  
*Return the middle point of side B.*
- bool `intersectsBelow (const cv::Point2f &gammaPoint, unsigned int polygonPointIndex)`  
*Check if the line determined by gammaPoint and polygon[polygonPointIndex] intersects the polygon below.*
- bool `intersectsAbove (const cv::Point2f &gammaPoint, unsigned int polygonPointIndex)`  
*Check if the line determined by gammaPoint and polygon[polygonPointIndex] intersects the polygon above.*
- unsigned int `intersects (double angleOfGammaAndPoint, unsigned int polygonPointIndex)`  
*Check if/where the line determined by gammaPoint and polygon[polygonPointIndex] intersects the polygon.*
- unsigned int `intersectsAboveOrBelow (unsigned int successorOrPredecessorIndex, unsigned int pointIndex)`  
*If (gamma(x) x) intersects P between successorOrPredecessorIndex and pointIndex is it above/below?*
- bool `isFlushAngleBetweenPredecessorAndSuccessor (double &angleFlushEdge, double anglePredecessor, double angleSuccessor)`  
*Check if the flush edge (opposite) angle lies between the predecessor and successor angle.*
- bool `isGammaAngleBetween (double &gammaAngle, double angle1, double angle2)`  
*Check if the angle of the line (gamma(p) p) or its opposite angle lie between angle1 and angle2.*
- bool `isGammaAngleEqualTo (double &gammaAngle, double angle)`  
*Check if the angle of the line (gamma(p) p) or its opposite angle is equal to the given angle.*
- double `height (unsigned int polygonPointIndex)`  
*Compute the height of the point specified by the given index.*
- double `height (const cv::Point2f &polygonPoint)`  
*Compute the height of the point.*
- bool `gamma (unsigned int polygonPointIndex, cv::Point2f &gammaPoint)`  
*Find gamma for a given point "p" specified by its index.*
- cv::Point2f `findVertexCOnSideB ()`  
*Find vertex C which lies on side B at a distance = 2 \* height(a-1) from side C.*

- bool [findGammaIntersectionPoints](#) (unsigned int polygonPointIndex, const cv::Point2f &side1StartVertex, const cv::Point2f &side1EndVertex, const cv::Point2f &side2StartVertex, const cv::Point2f &side2EndVertex, cv::Point2f &intersectionPoint1, cv::Point2f &intersectionPoint2)
 

*Find the intersection points to compute gamma(point)*
- bool [areIdenticalLines](#) (const std::vector< double > &side1Params, const std::vector< double > &side2Params, double sideCExtraParam)
 

*Check if the given lines are identical or not.*
- bool [areIntersectingLines](#) (const std::vector< double > &side1Params, const std::vector< double > &side2Params, double sideCExtraParam, cv::Point2f &intersectionPoint1, cv::Point2f &intersectionPoint2)
 

*Check if the given lines intersect or not. If the lines intersect find their intersection points.*
- std::vector< double > [lineEquationParameters](#) (const cv::Point2f &p, const cv::Point2f &q)
 

*Get the line equation parameters "a", "b" and "c" for the line determined by points "p" and "q".*
- void [advance](#) (unsigned int &index)
 

*Advance the given index with one position.*
- unsigned int [successor](#) (unsigned int index)
 

*Return the successor of the provided point index.*
- unsigned int [predecessor](#) (unsigned int index)
 

*Return the predecessor of the provided point index.*
- bool [almostEqual](#) (float lhsNumber, float rhsNumber)
 

*Check if the provided real numbers are "almost" equal.*

### Private Attributes

- unsigned int [validationFlag](#)
- cv::Point2f [vertexA](#)
- cv::Point2f [vertexB](#)
- cv::Point2f [vertexC](#)
- cv::Point2f [sideAStartVertex](#)
- cv::Point2f [sideAEndVertex](#)
- cv::Point2f [sideBStartVertex](#)
- cv::Point2f [sideBEndVertex](#)
- cv::Point2f [sideCStartVertex](#)
- cv::Point2f [sideCEndVertex](#)
- double [area](#)
- unsigned int [a](#)
- unsigned int [b](#)
- unsigned int [c](#)
- unsigned int [nrOfPoints](#)
- std::vector< cv::Point2f > [polygon](#)

## Static Private Attributes

- static const float `EPSILON` = 1E-5
- static const bool `CONVEX_HULL_CLOCKWISE` = true
- static const unsigned int `INTERSECTS_BELOW` = 1
- static const unsigned int `INTERSECTS_ABOVE` = 2
- static const unsigned int `INTERSECTS_CRITICAL` = 3
- static const unsigned int `INTERSECTS_LIMIT` = 4
- static const std::string `ERR_NR_POINTS` = "The number of 2D points in the input std::vector should be greater than 0."
- static const std::string `ERR_MIDPOINT_SIDE_B` = "The position of the middle point of side B could not be determined."
- static const std::string `ERR_SIDE_B_GAMMA` = "The position of side B could not be determined, because `gamma(b)` could not be computed."
- static const std::string `ERR_VERTEX_C_ON_SIDE_B` = "The position of the vertex C on side B could not be determined, because the considered lines do not intersect."
- static const std::string `ERR_TRIANGLE_VERTICES` = "The position of the triangle vertices could not be determined, because the sides of the triangle do not intersect."
- static const unsigned int `VALIDATION_SIDE_A_TANGENT` = 0
- static const unsigned int `VALIDATION_SIDE_B_TANGENT` = 1
- static const unsigned int `VALIDATION_SIDES_FLUSH` = 2

### 6.97.1 Detailed Description

Class for computing the minimum area enclosing triangle for a given polygon.

This implementation has a linear complexity ( $\theta(n)$ ) with respect to the number of points defining the convex polygon and is based on the algorithm described in the following paper:

J. O'Rourke, A. Aggarwal, S. Maddila, and M. Baldwin, 'An optimal algorithm for finding minimal enclosing triangles', Journal of Algorithms, vol. 7, no. 2, pp. 258–269, Jun. 1986.

Definition at line 19 of file `MinEnclosingTriangleFinder.hpp`.

### 6.97.2 Constructor & Destructor Documentation

#### 6.97.2.1 `MinEnclosingTriangleFinder::MinEnclosingTriangleFinder( )`

Definition at line 13 of file `MinEnclosingTriangleFinder.cpp`.

References `a`, `area`, `b`, `c`, `nrOfPoints`, and `validationFlag`.

#### 6.97.2.2 `MinEnclosingTriangleFinder::~MinEnclosingTriangleFinder( )`

Definition at line 25 of file `MinEnclosingTriangleFinder.cpp`.

### 6.97.3 Member Function Documentation

6.97.3.1 `void MinEnclosingTriangleFinder::advance ( unsigned int & index ) [private]`

Advance the given index with one position.

#### Parameters

<code>index</code>	Index of the point
--------------------	--------------------

Definition at line 444 of file MinEnclosingTriangleFinder.cpp.

References successor().

Referenced by advanceBToRightChain(), moveAIfLowAndBIfHigh(), and searchForBTangency().

6.97.3.2 `void MinEnclosingTriangleFinder::advanceBToRightChain ( ) [private]`

Advance b to the right chain.

See paper for more details

Definition at line 119 of file MinEnclosingTriangleFinder.cpp.

References advance(), b, multiscale::Numeric::greaterOrEqual(), height(), and successor().

Referenced by findMinEnclosingTriangle().

6.97.3.3 `bool MinEnclosingTriangleFinder::almostEqual ( float lhsNumber, float rhsNumber ) [private]`

Check if the provided real numbers are "almost" equal.

The expression for determining if two real numbers x and y are equal is: ( $\text{Abs}(x - y) \leq \text{EPSILON} * \text{Max}(1.0f, \text{Abs}(x), \text{Abs}(y))$ ).

#### Parameters

<code>lhsNumber</code>	The left hand side real number
<code>rhsNumber</code>	The right hand side real number

Definition at line 457 of file MinEnclosingTriangleFinder.cpp.

References EPSILON.

Referenced by intersects(), and isGammaAngleEqualTo().

---

6.97.3.4 **bool MinEnclosingTriangleFinder::areIdenticalLines ( const std::vector< double > & side1Params, const std::vector< double > & side2Params, double sideCExtraParam ) [private]**

Check if the given lines are identical or not.

The lines are specified as:  $ax + by + c = 0$  OR  $ax + by + c (+/-) \text{sideCExtraParam} = 0$

#### Parameters

<i>side1- Params</i>	Vector containing the values of a, b and c for side 1
<i>side2- Params</i>	Vector containing the values of a, b and c for side 2
<i>sideCExtra- Param</i>	Extra parameter for the flush edge C

Definition at line 404 of file MinEnclosingTriangleFinder.cpp.

References multiscale::Geometry2D::areIdenticalLines().

Referenced by findGammaIntersectionPoints().

6.97.3.5 **bool MinEnclosingTriangleFinder::areIntersectingLines ( const std::vector< double > & side1Params, const std::vector< double > & side2Params, double sideCExtraParam, cv::Point2f & intersectionPoint1, cv::Point2f & intersectionPoint2 ) [private]**

Check if the given lines intersect or not. If the lines intersect find their intersection points.

The lines are specified as:  $ax + by + c = 0$  OR  $ax + by + c (+/-) \text{sideCExtraParam} = 0$

#### Parameters

<i>side1- Params</i>	Vector containing the values of a, b and c for side 1
<i>side2- Params</i>	Vector containing the values of a, b and c for side 2
<i>sideCExtra- Param</i>	Extra parameter for the flush edge C
<i>intersection- Point1</i>	The first intersection point, if it exists
<i>intersection- Point2</i>	The second intersection point, if it exists

Definition at line 415 of file MinEnclosingTriangleFinder.cpp.

References multiscale::Geometry2D::lineIntersection().

Referenced by findGammaIntersectionPoints().

**6.97.3.6 double MinEnclosingTriangleFinder::find ( const std::vector< cv::Point2f > & points, std::vector< cv::Point2f > & minEnclosingTriangle )**

Find the minimum area enclosing triangle for the given 2D point set.

Precondition: Number of points in the set is at least 1.

#### Parameters

<i>points</i>	Set of points
<i>min- Enclosing- Triangle</i>	Minimum area triangle enclosing the given polygon

Definition at line 27 of file MinEnclosingTriangleFinder.cpp.

References ERR\_NR\_POINTS, and findMinTriangle().

Referenced by multiscale::analysis::Cluster::isTriangularMeasure(), multiscale::analysis::Region::isTriangularMeasure(), and multiscaletest::MinEnclosingTriangleFinderTest::RunTest().

**6.97.3.7 bool MinEnclosingTriangleFinder::findGammaIntersectionPoints ( unsigned int polygonPointIndex, const cv::Point2f & side1StartVertex, const cv::Point2f & side1EndVertex, const cv::Point2f & side2StartVertex, const cv::Point2f & side2EndVertex, cv::Point2f & intersectionPoint1, cv::Point2f & intersectionPoint2 ) [private]**

Find the intersection points to compute gamma(point)

#### Parameters

<i>polygon- PointIndex</i>	Index of the polygon point for which the distance is known
<i>side1Start- Vertex</i>	Start vertex for side 1
<i>side1End- Vertex</i>	End vertex for side 1
<i>side2Start- Vertex</i>	Start vertex for side 2
<i>side2End- Vertex</i>	End vertex for side 2
<i>intersection- Point1</i>	First intersection point between one pair of lines
<i>intersection- Point2</i>	Second intersection point between another pair of lines

Definition at line 378 of file MinEnclosingTriangleFinder.cpp.

References areIdenticalLines(), areIntersectingLines(), height(), and lineEquationParameters().

Referenced by `findVertexCOnSideB()`, and `gamma()`.

```
6.97.3.8 double MinEnclosingTriangleFinder::findMinEnclosingTriangle (
    const std::vector< cv::Point2f > & polygon, std::vector< cv::Point2f > &
    minEnclosingTriangle ) [private]
```

Find the minimum area enclosing triangle for the given polygon.

#### Parameters

<i>polygon</i>	Polygon of points for which the minimum area enclosing triangle will be found
<i>min- Enclosing- Triangle</i>	Minimum area triangle enclosing the given polygon

Definition at line 62 of file `MinEnclosingTriangleFinder.cpp`.

References `initialiseAlgorithmVariables()`.

Referenced by `findMinTriangle()`.

```
6.97.3.9 void MinEnclosingTriangleFinder::findMinEnclosingTriangle ( std::vector<
    cv::Point2f > & minEnclosingTriangle, double & minEnclosingTriangleArea )
[private]
```

Find the minimum area enclosing triangle for the given polygon.

#### Parameters

<i>min- Enclosing- Triangle</i>	Minimum area triangle enclosing the given polygon
<i>min- Enclosing- TriangleArea</i>	Area of the minimum area enclosing triangle

Definition at line 98 of file `MinEnclosingTriangleFinder.cpp`.

References `advanceBToRightChain()`, `c`, `isLocalMinimalTriangle()`, `isNotBTangency()`, `moveAIfLowAndBIfHigh()`, `nrOfPoints`, `searchForBTangency()`, `updateMinEnclosingTriangle()`, `updateSideB()`, `updateSidesBA()`, and `updateSidesCA()`.

```
6.97.3.10 double MinEnclosingTriangleFinder::findMinTriangle ( const std::vector<
    cv::Point2f > & points, std::vector< cv::Point2f > & minEnclosingTriangle )
[private]
```

Find the minimum area enclosing triangle for the given 2D point set.

**Parameters**

<i>points</i>	Set of points
<i>min- Enclosing- Triangle</i>	Minimum area triangle enclosing the given polygon

Definition at line 39 of file MinEnclosingTriangleFinder.cpp.

References `findMinEnclosingTriangle()`, `initialise()`, `polygon`, and `returnMinEnclosingTriangle()`.

Referenced by `find()`.

#### 6.97.3.11 cv::Point2f MinEnclosingTriangleFinder::findVertexCOnSideB( ) [private]

Find vertex C which lies on side B at a distance =  $2 * \text{height}(a-1)$  from side C.

Considering that line (x y) is a line parallel to (c c-1) and that the distance between the lines is equal to  $2 * \text{height}(a-1)$ , we can have two possible (x y) lines.

Therefore, we will compute two intersection points between the lines (x y) and (b b-1) and take the point which is closest to point `polygon[b]`.

See paper and formula for distance from point to a line for more details

Definition at line 360 of file MinEnclosingTriangleFinder.cpp.

References `a`, `multiscale::Geometry2D::areOnTheSameSideOfLine()`, `c`, `ERR_VERTEX_C_ON_SIDE_B`, `findGammaIntersectionPoints()`, `polygon`, `predecessor()`, `sideBEndVertex`, `sideBStartVertex`, `sideCEndVertex`, `sideCStartVertex`, and `successor()`.

Referenced by `updateSidesBA()`.

#### 6.97.3.12 bool MinEnclosingTriangleFinder::gamma( unsigned int *polygonPointIndex*, cv::Point2f & *gammaPoint* ) [private]

Find gamma for a given point "p" specified by its index.

The function returns true if gamma exists i.e. if lines (a a-1) and (x y) intersect and false otherwise. In case the two lines intersect in point `intersectionPoint`, gamma is computed.

Considering that line (x y) is a line parallel to (c c-1) and that the distance between the lines is equal to  $2 * \text{height}(p)$ , we can have two possible (x y) lines.

Therefore, we will compute two intersection points between the lines (x y) and (a a-1) and take the point which is closest to point `polygon[a]`.

See paper and formula for distance from point to a line for more details

**Parameters**

<i>polygon- PointIndex</i>	Index of the polygon point
<i>gammaPoint</i>	<code>cv::Point2f gamma(polygon[polygonPointIndex])</code>

Definition at line 340 of file MinEnclosingTriangleFinder.cpp.

References a, multiscale::Geometry2D::areOnTheSameSideOfLine(), c, findGammaIntersectionPoints(), polygon, predecessor(), and successor().

Referenced by isNotBTangency(), moveAIfLowAndBIfHigh(), searchForBTangency(), and updateSideB().

**6.97.3.13 double MinEnclosingTriangleFinder::height ( unsigned int *polygonPointIndex* ) [private]**

Compute the height of the point specified by the given index.

See paper for more details

#### Parameters

<i>polygon- PointIndex</i>	Index of the polygon point
--------------------------------	----------------------------

Definition at line 324 of file MinEnclosingTriangleFinder.cpp.

References c, multiscale::Geometry2D::distanceFromPointToLine(), polygon, and predecessor().

Referenced by advanceBToLeftChain(), findGammaIntersectionPoints(), intersectsAboveOrBelow(), isNotBTangency(), moveAIfLowAndBIfHigh(), searchForBTangency(), and updateSidesBA().

**6.97.3.14 double MinEnclosingTriangleFinder::height ( const cv::Point2f & *polygonPoint* ) [private]**

Compute the height of the point.

See paper for more details

#### Parameters

<i>polygonPoint</i>	Polygon point
---------------------	---------------

Definition at line 333 of file MinEnclosingTriangleFinder.cpp.

References c, multiscale::Geometry2D::distanceFromPointToLine(), polygon, and predecessor().

**6.97.3.15 void MinEnclosingTriangleFinder::initialise ( const std::vector< cv::Point2f > & *points*, std::vector< cv::Point2f > & *minEnclosingTriangle* ) [private]**

Initialisation function for the class.

Initialise the polygon and other class' fields.

**Parameters**

<i>points</i>	Set of points
<i>min- Enclosing- Triangle</i>	Minimum area triangle enclosing the given polygon

Definition at line 50 of file MinEnclosingTriangleFinder.cpp.

References initialiseConvexPolygon().

Referenced by findMinTriangle().

#### 6.97.3.16 void MinEnclosingTriangleFinder::initialiseAlgorithmVariables ( ) [private]

Initialisation of the algorithm variables.

Definition at line 90 of file MinEnclosingTriangleFinder.cpp.

References a, b, c, nrOfPoints, and polygon.

Referenced by findMinEnclosingTriangle().

#### 6.97.3.17 void MinEnclosingTriangleFinder::initialiseConvexPolygon ( const std::vector< cv::Point2f > & *points* ) [private]

Initialise polygon as the convex hull of the given set of points.

**Parameters**

<i>points</i>	Set of points
---------------	---------------

Definition at line 58 of file MinEnclosingTriangleFinder.cpp.

References multiscale::Geometry2D::computeConvexHull(), CONVEX\_HULL\_CLOCKWISE, and polygon.

Referenced by initialise().

#### 6.97.3.18 unsigned int MinEnclosingTriangleFinder::intersects ( double *angleOfGammaAndPoint*, unsigned int *polygonPointIndex* ) [private]

Check if/where the line determined by gammaPoint and polygon[polygonPointIndex] intersects the polygon.

**Parameters**

<i>angleOf- GammaAnd- Point</i>	Angle between gammaPoint and polygon[polygonPointIndex]
<i>polygon- PointIndex</i>	Index of the polygon point which is considered when determining the line

Definition at line 260 of file MinEnclosingTriangleFinder.cpp.

References `almostEqual()`, `multiscale::Geometry2D::angleOfLineWrtOxAxis()`, `c`, `INTERSECTS_BELOW`, `INTERSECTS_CRITICAL`, `intersectsAboveOrBelow()`, `isFlushAngleBetweenPredecessorAndSuccessor()`, `isGammaAngleBetween()`, `isGammaAngleEqualTo()`, `polygon`, `predecessor()`, and `successor()`.

Referenced by `intersectsAbove()`, and `intersectsBelow()`.

**6.97.3.19 bool MinEnclosingTriangleFinder::intersectsAbove ( const cv::Point2f & *gammaPoint*, unsigned int *polygonPointIndex* ) [private]**

Check if the line determined by `gammaPoint` and `polygon[polygonPointIndex]` intersects the polygon above.

Check if the line determined by `gammaPoint` and `polygon[polygonPointIndex]` intersects the polygon above the point `polygon[polygonPointIndex]`

**Parameters**

<code>gammaPoint</code>	Gamma(p)
<code>polygon- PointIndex</code>	Index of the polygon point which is considered when determining the line

Definition at line 254 of file MinEnclosingTriangleFinder.cpp.

References `multiscale::Geometry2D::angleOfLineWrtOxAxis()`, `intersects()`, `INTERSECTS_ABOVE`, and `polygon`.

Referenced by `isNotBTangency()`.

**6.97.3.20 unsigned int MinEnclosingTriangleFinder::intersectsAboveOrBelow ( unsigned int *successorOrPredecessorIndex*, unsigned int *pointIndex* ) [private]**

If (`gamma(x) x`) intersects P between `successorOrPredecessorIndex` and `pointIndex` is it above/below?

**Parameters**

<code>successor- Or- Predecessor- Index</code>	Index of the successor or predecessor
<code>pointIndex</code>	Index of the point x in the polygon

Definition at line 293 of file MinEnclosingTriangleFinder.cpp.

References `height()`, `INTERSECTS_ABOVE`, and `INTERSECTS_BELOW`.

Referenced by `intersects()`.

6.97.3.21 `bool MinEnclosingTriangleFinder::intersectsBelow ( const cv::Point2f & gammaPoint, unsigned int polygonPointIndex ) [private]`

Check if the line determined by gammaPoint and polygon[polygonPointIndex] intersects the polygon below.

Check if the line determined by gammaPoint and polygon[polygonPointIndex] intersects the polygon below the point polygon[polygonPointIndex]

#### Parameters

<code>gammaPoint</code>	Gamma(p)
<code>polygon- PointIndex</code>	Index of the polygon point which is considered when determining the line

Definition at line 248 of file MinEnclosingTriangleFinder.cpp.

References multiscale::Geometry2D::angleOfLineWrtOxAxis(), intersects(), INTERSECTS\_BELOW, and polygon.

Referenced by moveAlfLowAndBlfHigh(), and searchForBTangency().

6.97.3.22 `bool MinEnclosingTriangleFinder::isFlushAngleBetweenPredecessor-  
AndSuccessor ( double & angleFlushEdge, double anglePredecessor, double  
angleSuccessor ) [private]`

Check if the flush edge (opposite) angle lies between the predecessor and successor angle.

#### Parameters

<code>angleFlush- Edge</code>	Angle of the flush edge
<code>angle- Predecessor</code>	Angle of the predecessor
<code>angle- Successor</code>	Angle of the successor

Definition at line 302 of file MinEnclosingTriangleFinder.cpp.

References multiscale::Geometry2D::isAngleBetweenNonReflex(), multiscale::-  
Geometry2D::isOppositeAngleBetweenNonReflex(), and multiscale::Geometry2D-  
::oppositeAngle().

Referenced by intersects().

6.97.3.23 `bool MinEnclosingTriangleFinder::isGammaAngleBetween ( double &  
gammaAngle, double angle1, double angle2 ) [private]`

Check if the angle of the line (gamma(p) p) or its opposite angle lie between angle1 and angle2.

**Parameters**

<i>gamma- Angle</i>	Angle of the line ( <code>gamma(p) p</code> )
<i>angle1</i>	One of the boundary angles
<i>angle2</i>	Another boundary angle

Definition at line 316 of file `MinEnclosingTriangleFinder.cpp`.

References `multiscale::Geometry2D::isAngleBetweenNonReflex()`.

Referenced by `intersects()`.

**6.97.3.24 bool MinEnclosingTriangleFinder::isGammaAngleEqualTo ( double & *gammaAngle*, double *angle* ) [private]**

Check if the angle of the line (`gamma(p) p`) or its opposite angle is equal to the given angle.

**Parameters**

<i>gamma- Angle</i>	Angle of the line ( <code>gamma(p) p</code> )
<i>angle</i>	Angle to compare against

Definition at line 320 of file `MinEnclosingTriangleFinder.cpp`.

References `almostEqual()`.

Referenced by `intersects()`.

**6.97.3.25 bool MinEnclosingTriangleFinder::isLocalMinimalTriangle ( ) [private]**

Update the triangle vertices after all sides were set and check if a local minimal triangle was found.

See paper for more details

Definition at line 192 of file `MinEnclosingTriangleFinder.cpp`.

References `isValidMinimalTriangle()`, `multiscale::Geometry2D::lineIntersection()`, `sideAEndVertex`, `sideAStartVertex`, `sideBEndVertex`, `sideBStartVertex`, `sideCEndVertex`, `sideCStartVertex`, `vertexA`, `vertexB`, and `vertexC`.

Referenced by `findMinEnclosingTriangle()`.

**6.97.3.26 bool MinEnclosingTriangleFinder::isNotBTangency ( ) [private]**

Check if tangency for side B was not obtained.

See paper for more details

Definition at line 146 of file MinEnclosingTriangleFinder.cpp.

References a, b, gamma(), height(), intersectsAbove(), and predecessor().

Referenced by findMinEnclosingTriangle().

**6.97.3.27 bool MinEnclosingTriangleFinder::isValidMinimalTriangle ( )**  
 [private]

Check if the found minimal triangle is valid.

This means that all midpoints of the triangle should touch the polygon

See paper for more details

Definition at line 202 of file MinEnclosingTriangleFinder.cpp.

References a, multiscale::Geometry2D::areEqualPoints(), b, multiscale::Geometry2D::isPointOnLineSegment(), multiscale::Geometry2D::middlePoint(), polygon, predecessor(), sideAEndVertex, sideAStartVertex, sideBEndVertex, sideBStartVertex, sideCEndVertex, sideCStartVertex, VALIDATION\_SIDE\_A\_TANGENT, VALIDATION\_SIDE\_B\_TANGENT, validationFlag, vertexA, vertexB, and vertexC.

Referenced by isLocalMinimalTriangle().

**6.97.3.28 std::vector< double > MinEnclosingTriangleFinder::lineEquationParameters ( const cv::Point2f & p, const cv::Point2f & q )**  
 [private]

Get the line equation parameters "a", "b" and "c" for the line determined by points "p" and "q".

The equation of the line is considered in the general form:  $ax + by + c = 0$

**Parameters**

<i>p</i>	One point for defining the equation of the line
<i>q</i>	Second point for defining the equation of the line

Definition at line 430 of file MinEnclosingTriangleFinder.cpp.

References a, b, c, and multiscale::Geometry2D::lineEquationDeterminedByPoints().

Referenced by findGammaIntersectionPoints().

**6.97.3.29 bool MinEnclosingTriangleFinder::middlePointOfSideB ( cv::Point2f & middlePointOfSideB )** [private]

Return the middle point of side B.

Definition at line 235 of file MinEnclosingTriangleFinder.cpp.

References multiscale::Geometry2D::lineIntersection(), multiscale::Geometry2D::middlePoint(), sideAEndVertex, sideAStartVertex, sideBEndVertex, sideBStartVertex,

sideCEndVertex, sideCStartVertex, vertexA, and vertexC.

Referenced by updateSidesBA().

**6.97.3.30 void MinEnclosingTriangleFinder::moveAIfLowAndBIfHigh ( ) [private]**

Move "a" if it is low and "b" if it is high.

See paper for more details

Definition at line 125 of file MinEnclosingTriangleFinder.cpp.

References a, advance(), b, gamma(), height(), and intersectsBelow().

Referenced by findMinEnclosingTriangle().

**6.97.3.31 unsigned int MinEnclosingTriangleFinder::predecessor ( unsigned int *index* ) [private]**

Return the predecessor of the provided point index.

The predecessor of the first polygon point is the last polygon point (circular referencing)

**Parameters**

<i>index</i>	Index of the point
--------------	--------------------

Definition at line 452 of file MinEnclosingTriangleFinder.cpp.

References nrOfPoints.

Referenced by findVertexCOnSideB(), gamma(), height(), intersects(), isNotBTangency(), isValidMinimalTriangle(), searchForBTangency(), updateSidesBA(), and updateSidesCA().

**6.97.3.32 double MinEnclosingTriangleFinder::returnMinEnclosingTriangle ( const std::vector< cv::Point2f > & *polygon*, std::vector< cv::Point2f > & *minEnclosingTriangle* ) [private]**

Return the minimum area enclosing triangle in case the given polygon has at most three points.

**Parameters**

<i>polygon</i>	Polygon of points for which the minimum area enclosing triangle will be found
<i>min- Enclosing- Triangle</i>	Minimum area triangle enclosing the given polygon

Definition at line 73 of file MinEnclosingTriangleFinder.cpp.

References multiscale::Geometry2D::areaOfTriangle().

Referenced by findMinTriangle().

**6.97.3.33 void MinEnclosingTriangleFinder::searchForBTangency ( )**  
 [private]

Search for the tangency of side B.

See paper for more details

Definition at line 137 of file MinEnclosingTriangleFinder.cpp.

References a, advance(), b, gamma(), multiscale::Numeric::greaterOrEqual(), height(), intersectsBelow(), and predecessor().

Referenced by findMinEnclosingTriangle().

**6.97.3.34 unsigned int MinEnclosingTriangleFinder::successor ( unsigned int *index* )**  
 [private]

Return the successor of the provided point index.

The successor of the last polygon point is the first polygon point (circular referencing)

**Parameters**

<i>index</i>	Index of the point
--------------	--------------------

Definition at line 448 of file MinEnclosingTriangleFinder.cpp.

References nrOfPoints.

Referenced by advance(), advanceBToRightChain(), findVertexCOnSideB(), gamma(), and intersects().

**6.97.3.35 void MinEnclosingTriangleFinder::updateMinEnclosingTriangle  
 ( std::vector< cv::Point2f > & *minEnclosingTriangle*, double &  
*minEnclosingTriangleArea* ) [private]**

Update the current minimum area enclosing triangle if the newly obtained one has a smaller area.

**Parameters**

<i>min- Enclosing- Triangle</i>	Minimum area triangle enclosing the given polygon
<i>min- Enclosing- TriangleArea</i>	Area of the minimum area triangle enclosing the given polygon

Definition at line 220 of file MinEnclosingTriangleFinder.cpp.

References area, multiscale::Geometry2D::areaOfTriangle(), vertexA, vertexB, and vertexC.

Referenced by findMinEnclosingTriangle().

#### 6.97.3.36 void MinEnclosingTriangleFinder::updateSideB( ) [private]

Set side B if tangency for side B was obtained.

See paper for more details

Definition at line 182 of file MinEnclosingTriangleFinder.cpp.

References b, ERR\_SIDE\_B\_GAMMA, gamma(), polygon, sideBEndVertex, sideBStartVertex, VALIDATION\_SIDE\_B\_TANGENT, and validationFlag.

Referenced by findMinEnclosingTriangle().

#### 6.97.3.37 void MinEnclosingTriangleFinder::updateSidesBA( ) [private]

Update sides B and possibly A if tangency for side B was not obtained.

See paper for more details

Definition at line 164 of file MinEnclosingTriangleFinder.cpp.

References a, b, findVertexCOnSideB(), height(), middlePointOfSideB(), polygon, predecessor(), sideAEndVertex, sideAStartVertex, sideBEndVertex, sideBStartVertex, VALIDATION\_SIDE\_A\_TANGENT, VALIDATION\_SIDES\_FLUSH, and validationFlag.

Referenced by findMinEnclosingTriangle().

#### 6.97.3.38 void MinEnclosingTriangleFinder::updateSidesCA( ) [private]

Update sides A and C.

Side C will have as start and end vertices the polygon points "c" and "c-1" Side A will have as start and end vertices the polygon points "a" and "a-1"

Definition at line 156 of file MinEnclosingTriangleFinder.cpp.

References a, c, polygon, predecessor(), sideAEndVertex, sideAStartVertex, sideCEndVertex, and sideCStartVertex.

Referenced by findMinEnclosingTriangle().

### 6.97.4 Member Data Documentation

#### 6.97.4.1 unsigned int multiscale::MinEnclosingTriangleFinder::a [private]

Index of point "a"; see paper for more details

Definition at line 44 of file MinEnclosingTriangleFinder.hpp.

Referenced by findVertexCOnSideB(), gamma(), initialiseAlgorithmVariables(), isNotBTangency(), isValidMinimalTriangle(), lineEquationParameters(), MinEnclosingTriangleFinder(), moveAIfLowAndBIfHigh(), searchForBTangency(), updateSidesBA(), and updateSidesCA().

#### 6.97.4.2 double multiscale::MinEnclosingTriangleFinder::area [private]

Area of the current considered enclosing triangle

Definition at line 42 of file MinEnclosingTriangleFinder.hpp.

Referenced by MinEnclosingTriangleFinder(), and updateMinEnclosingTriangle().

#### 6.97.4.3 unsigned int multiscale::MinEnclosingTriangleFinder::b [private]

Index of point "b"; see paper for more details

Definition at line 45 of file MinEnclosingTriangleFinder.hpp.

Referenced by advanceBToRightChain(), initialiseAlgorithmVariables(), isNotBTangency(), isValidMinimalTriangle(), lineEquationParameters(), MinEnclosingTriangleFinder(), moveAIfLowAndBIfHigh(), searchForBTangency(), updateSideB(), and updateSidesBA().

#### 6.97.4.4 unsigned int multiscale::MinEnclosingTriangleFinder::c [private]

Index of point "c"; see paper for more details

Definition at line 46 of file MinEnclosingTriangleFinder.hpp.

Referenced by findMinEnclosingTriangle(), findVertexCOnSideB(), gamma(), height(), initialiseAlgorithmVariables(), intersects(), lineEquationParameters(), MinEnclosingTriangleFinder(), and updateSidesCA().

#### 6.97.4.5 const bool MinEnclosingTriangleFinder::CONVEX\_HULL\_CLOCKWISE = true [static, private]

Definition at line 396 of file MinEnclosingTriangleFinder.hpp.

Referenced by initialiseConvexPolygon().

#### 6.97.4.6 const float MinEnclosingTriangleFinder::EPSILON = 1E-5 [static, private]

Definition at line 394 of file MinEnclosingTriangleFinder.hpp.

Referenced by almostEqual().

6.97.4.7 `const std::string MinEnclosingTriangleFinder::ERR_MIDPOINT_SIDE_B = "The position of the middle point of side B could not be determined." [static, private]`

Definition at line 404 of file MinEnclosingTriangleFinder.hpp.

6.97.4.8 `const std::string MinEnclosingTriangleFinder::ERR_NR_POINTS = "The number of 2D points in the input std::vector should be greater than 0." [static, private]`

Definition at line 403 of file MinEnclosingTriangleFinder.hpp.

Referenced by `find()`.

6.97.4.9 `const std::string MinEnclosingTriangleFinder::ERR_SIDE_B_GAMMA = "The position of side B could not be determined, because gamma(b) could not be computed." [static, private]`

Definition at line 405 of file MinEnclosingTriangleFinder.hpp.

Referenced by `updateSideB()`.

6.97.4.10 `const std::string MinEnclosingTriangleFinder::ERR_TRIANGLE_VERTICES = "The position of the triangle vertices could not be determined, because the sides of the triangle do not intersect." [static, private]`

Definition at line 407 of file MinEnclosingTriangleFinder.hpp.

6.97.4.11 `const std::string MinEnclosingTriangleFinder::ERR_VERTEX_C_ON_SIDE_B = "The position of the vertex C on side B could not be determined, because the considered lines do not intersect." [static, private]`

Definition at line 406 of file MinEnclosingTriangleFinder.hpp.

Referenced by `findVertexCOnSideB()`.

6.97.4.12 `const unsigned int MinEnclosingTriangleFinder::INTERSECTS_ABOVE = 2 [static, private]`

Definition at line 399 of file MinEnclosingTriangleFinder.hpp.

Referenced by `intersectsAbove()`, and `intersectsAboveOrBelow()`.

6.97.4.13 `const unsigned int MinEnclosingTriangleFinder::INTERSECTS_BELOW = 1 [static, private]`

Definition at line 398 of file MinEnclosingTriangleFinder.hpp.

Referenced by intersects(), intersectsAboveOrBelow(), and intersectsBelow().

**6.97.4.14 const unsigned int MinEnclosingTriangleFinder::INTERSECTS\_CRITICAL = 3 [static, private]**

Definition at line 400 of file MinEnclosingTriangleFinder.hpp.

Referenced by intersects().

**6.97.4.15 const unsigned int MinEnclosingTriangleFinder::INTERSECTS\_LIMIT = 4 [static, private]**

Definition at line 401 of file MinEnclosingTriangleFinder.hpp.

**6.97.4.16 unsigned int multiscale::MinEnclosingTriangleFinder::nrOfPoints [private]**

Number of points defining the polygon

Definition at line 48 of file MinEnclosingTriangleFinder.hpp.

Referenced by findMinEnclosingTriangle(), initialiseAlgorithmVariables(), MinEnclosingTriangleFinder(), predecessor(), and successor().

**6.97.4.17 std::vector<cv::Point2f> multiscale::MinEnclosingTriangleFinder::polygon [private]**

Polygon for which the minimum area enclosing triangle is computed

Definition at line 50 of file MinEnclosingTriangleFinder.hpp.

Referenced by findMinTriangle(), findVertexCOnSideB(), gamma(), height(), initialiseAlgorithmVariables(), initialiseConvexPolygon(), intersects(), intersectsAbove(), intersectsBelow(), isValidMinimalTriangle(), updateSideB(), updateSidesBA(), and updateSidesCA().

**6.97.4.18 cv::Point2f multiscale::MinEnclosingTriangleFinder::sideAEndVertex [private]**

Ending vertex for side A of triangle

Definition at line 34 of file MinEnclosingTriangleFinder.hpp.

Referenced by isLocalMinimalTriangle(), isValidMinimalTriangle(), middlePointOfSideB(), updateSidesBA(), and updateSidesCA().

**6.97.4.19 cv::Point2f multiscale::MinEnclosingTriangleFinder::sideAStartVertex  
[private]**

Starting vertex for side A of triangle

Definition at line 33 of file MinEnclosingTriangleFinder.hpp.

Referenced by isLocalMinimalTriangle(), isValidMinimalTriangle(), middlePointOfSideB(), updateSidesBA(), and updateSidesCA().

**6.97.4.20 cv::Point2f multiscale::MinEnclosingTriangleFinder::sideBEndVertex  
[private]**

Ending vertex for side B of triangle

Definition at line 37 of file MinEnclosingTriangleFinder.hpp.

Referenced by findVertexCOnSideB(), isLocalMinimalTriangle(), isValidMinimalTriangle(), middlePointOfSideB(), updateSideB(), and updateSidesBA().

**6.97.4.21 cv::Point2f multiscale::MinEnclosingTriangleFinder::sideBStartVertex  
[private]**

Starting vertex for side B of triangle

Definition at line 36 of file MinEnclosingTriangleFinder.hpp.

Referenced by findVertexCOnSideB(), isLocalMinimalTriangle(), isValidMinimalTriangle(), middlePointOfSideB(), updateSideB(), and updateSidesBA().

**6.97.4.22 cv::Point2f multiscale::MinEnclosingTriangleFinder::sideCEndVertex  
[private]**

Ending vertex for side C of triangle

Definition at line 40 of file MinEnclosingTriangleFinder.hpp.

Referenced by findVertexCOnSideB(), isLocalMinimalTriangle(), isValidMinimalTriangle(), middlePointOfSideB(), and updateSidesCA().

**6.97.4.23 cv::Point2f multiscale::MinEnclosingTriangleFinder::sideCStartVertex  
[private]**

Starting vertex for side C of triangle

Definition at line 39 of file MinEnclosingTriangleFinder.hpp.

Referenced by findVertexCOnSideB(), isLocalMinimalTriangle(), isValidMinimalTriangle(), middlePointOfSideB(), and updateSidesCA().

```
6.97.4.24 const unsigned int MinEnclosingTriangleFinder::VALIDATION_SIDE_A_TANGENT = 0 [static, private]
```

Definition at line 409 of file MinEnclosingTriangleFinder.hpp.

Referenced by isValidMinimalTriangle(), and updateSidesBA().

```
6.97.4.25 const unsigned int MinEnclosingTriangleFinder::VALIDATION_SIDE_B_TANGENT = 1 [static, private]
```

Definition at line 410 of file MinEnclosingTriangleFinder.hpp.

Referenced by isValidMinimalTriangle(), and updateSideB().

```
6.97.4.26 const unsigned int MinEnclosingTriangleFinder::VALIDATION_SIDES_FLUSH = 2 [static, private]
```

Definition at line 411 of file MinEnclosingTriangleFinder.hpp.

Referenced by updateSidesBA().

```
6.97.4.27 unsigned int multiscale::MinEnclosingTriangleFinder::validationFlag [private]
```

Validation flag can take the following values:

- VALIDATION\_SIDE\_A\_TANGENT;
- VALIDATION\_SIDE\_B\_TANGENT;
- VALIDATION\_SIDES\_FLUSH.

Definition at line 23 of file MinEnclosingTriangleFinder.hpp.

Referenced by isValidMinimalTriangle(), MinEnclosingTriangleFinder(), updateSideB(), and updateSidesBA().

```
6.97.4.28 cv::Point2f multiscale::MinEnclosingTriangleFinder::vertexA [private]
```

Vertex A of the current considered enclosing triangle

Definition at line 29 of file MinEnclosingTriangleFinder.hpp.

Referenced by isLocalMinimalTriangle(), isValidMinimalTriangle(), middlePointOfSideB(), and updateMinEnclosingTriangle().

**6.97.4.29 cv::Point2f multiscale::MinEnclosingTriangleFinder::vertexB  
[private]**

Vertex B of the current considered enclosing triangle

Definition at line 30 of file MinEnclosingTriangleFinder.hpp.

Referenced by isLocalMinimalTriangle(), isValidMinimalTriangle(), and updateMinEnclosingTriangle().

**6.97.4.30 cv::Point2f multiscale::MinEnclosingTriangleFinder::vertexC  
[private]**

Vertex C of the current considered enclosing triangle

Definition at line 31 of file MinEnclosingTriangleFinder.hpp.

Referenced by isLocalMinimalTriangle(), isValidMinimalTriangle(), middlePointOfSideB(), and updateMinEnclosingTriangle().

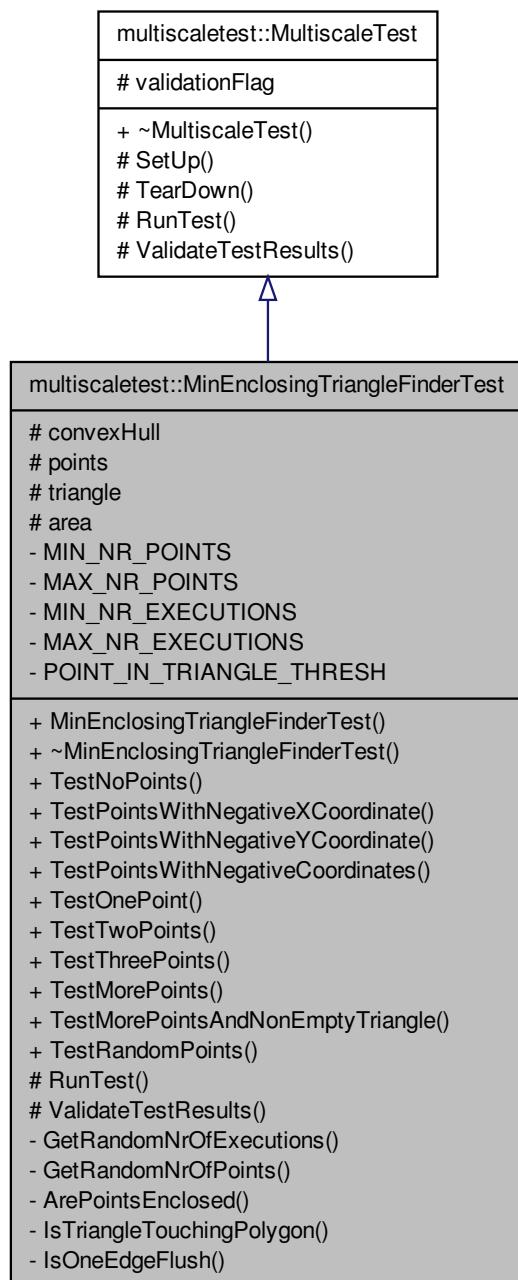
The documentation for this class was generated from the following files:

- MinEnclosingTriangleFinder.hpp
- MinEnclosingTriangleFinder.cpp

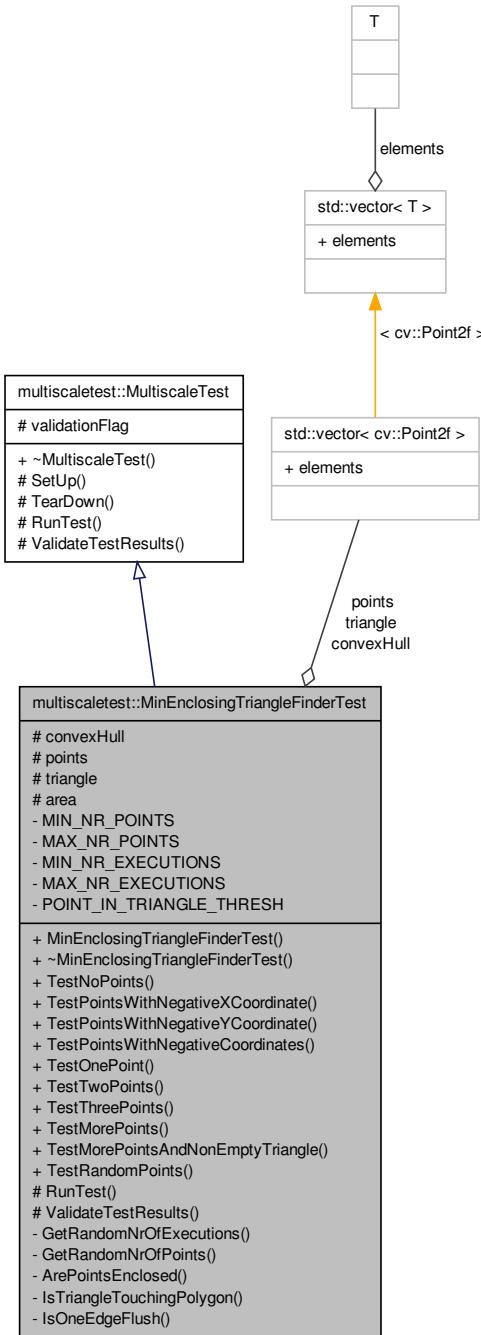
**6.98 multiscaletest::MinEnclosingTriangleFinderTest Class - Reference**

Class for testing the minimum enclosing triangle algorithm.

Inheritance diagram for multiscaletest::MinEnclosingTriangleFinderTest:



Collaboration diagram for multiscaletest::MinEnclosingTriangleFinderTest:



## Public Member Functions

- [MinEnclosingTriangleFinderTest \(\)](#)  
*Test the scenario when an empty vector of points is provided.*
- [~MinEnclosingTriangleFinderTest \(\)](#)
- [bool TestNoPoints \(\)](#)  
*Test the scenario when there exists at least one point with negative x coordinate.*
- [bool TestPointsWithNegativeYCoordinate \(\)](#)  
*Test the scenario when there exists at least one point with negative y coordinate.*
- [bool TestPointsWithNegativeCoordinates \(\)](#)  
*Test the scenario when there exists at least one point with negative coordinates.*
- [bool TestOnePoint \(\)](#)  
*Test the scenario when only one input point is provided.*
- [bool TestTwoPoints \(\)](#)  
*Test the scenario when only two input points are provided.*
- [bool TestThreePoints \(\)](#)  
*Test the scenario when only three input points are provided.*
- [bool TestMorePoints \(\)](#)  
*Test the scenario when more than three input points are provided.*
- [bool TestMorePointsAndNonEmptyTriangle \(\)](#)  
*Test the scenario when the output vector is not empty.*
- [bool TestRandomPoints \(\)](#)  
*Test the scenario when randomly initialised vectors of input points are provided.*

## Protected Member Functions

- [void RunTest \(\) override](#)  
*Run the test for the given set of points.*
- [void ValidateTestResults \(\) override](#)  
*Check if the obtained results are valid.*

## Protected Attributes

- [std::vector< cv::Point2f > convexHull](#)
- [std::vector< cv::Point2f > points](#)
- [std::vector< cv::Point2f > triangle](#)
- [double area](#)

## Private Member Functions

- int `GetRandomNrOfExecutions ()`  
*Get a random number of executions.*
- int `GetRandomNrOfPoints ()`  
*Get a random number of points.*
- bool `ArePointsEnclosed ()`  
*Check if all the points are enclosed by the polygon.*
- bool `IsTriangleTouchingPolygon ()`  
*Check if the triangle's middle points are touching the polygon.*
- bool `IsOneEdgeFlush ()`  
*Check if at least one of the triangle sides is flush with a polygon edge.*

## Static Private Attributes

- static const int `MIN_NR_POINTS` = 1
- static const int `MAX_NR_POINTS` = 10000
- static const int `MIN_NR_EXECUTIONS` = 5000
- static const int `MAX_NR_EXECUTIONS` = 10000
- static const double `POINT_IN_TRIANGLE_THRESH` = 1E-4

### 6.98.1 Detailed Description

Class for testing the minimum enclosing triangle algorithm.

Definition at line 14 of file MinEnclosingTriangleFinderTest.cpp.

### 6.98.2 Constructor & Destructor Documentation

#### 6.98.2.1 `multiscaletest::MinEnclosingTriangleFinderTest::MinEnclosingTriangleFinderTest( )`

Definition at line 96 of file MinEnclosingTriangleFinderTest.cpp.

#### 6.98.2.2 `multiscaletest::MinEnclosingTriangleFinderTest::~MinEnclosingTriangleFinderTest( )`

Definition at line 104 of file MinEnclosingTriangleFinderTest.cpp.

### 6.98.3 Member Function Documentation

#### 6.98.3.1 `bool multiscaletest::MinEnclosingTriangleFinderTest::ArePointsEnclosed( ) [private]`

Check if all the points are enclosed by the polygon.

Definition at line 244 of file MinEnclosingTriangleFinderTest.cpp.

**6.98.3.2 int multiscaletest::MinEnclosingTriangleFinderTest::GetRandomNrOfExecutions( ) [private]**

Get a random number of executions.

Definition at line 234 of file MinEnclosingTriangleFinderTest.cpp.

**6.98.3.3 int multiscaletest::MinEnclosingTriangleFinderTest::GetRandomNrOfPoints( ) [private]**

Get a random number of points.

Definition at line 239 of file MinEnclosingTriangleFinderTest.cpp.

**6.98.3.4 bool multiscaletest::MinEnclosingTriangleFinderTest::IsOneEdgeFlush( ) [private]**

Check if at least one of the triangle sides is flush with a polygon edge.

Definition at line 280 of file MinEnclosingTriangleFinderTest.cpp.

References multiscale::Geometry2D::isPointOnLineSegment().

**6.98.3.5 bool multiscaletest::MinEnclosingTriangleFinderTest::IsTriangleTouchingPolygon( ) [private]**

Check if the triangle's middle points are touching the polygon.

Definition at line 258 of file MinEnclosingTriangleFinderTest.cpp.

References multiscale::Geometry2D::isPointOnLineSegment(), and multiscale::Geometry2D::middlePoint().

**6.98.3.6 void multiscaletest::MinEnclosingTriangleFinderTest::RunTest( ) [override, protected, virtual]**

Run the test for the given set of points.

Implements [multiscaletest::MultiscaleTest](#).

Definition at line 220 of file MinEnclosingTriangleFinderTest.cpp.

References multiscale::Geometry2D::computeConvexHull(), and multiscale::MinEnclosingTriangleFinder::find().

**6.98.3.7 bool multiscaletest::MinEnclosingTriangleFinderTest::TestMorePoints( )**

Test the scenario when more than three input points are provided.

Definition at line 173 of file MinEnclosingTriangleFinderTest.cpp.

**6.98.3.8 bool multiscaletest::MinEnclosingTriangleFinderTest::TestMorePoints-AndNonEmptyTriangle( )**

Test the scenario when the output vector is not empty.

Definition at line 183 of file MinEnclosingTriangleFinderTest.cpp.

**6.98.3.9 bool multiscaletest::MinEnclosingTriangleFinderTest::TestNoPoints( )**

Test the scenario when an empty vector of points is provided.

Definition at line 112 of file MinEnclosingTriangleFinderTest.cpp.

**6.98.3.10 bool multiscaletest::MinEnclosingTriangleFinderTest::TestOnePoint( )**

Test the scenario when only one input point is provided.

Definition at line 146 of file MinEnclosingTriangleFinderTest.cpp.

**6.98.3.11 bool multiscaletest::MinEnclosingTriangleFinderTest::TestPointsWith-NegativeCoordinates( )**

Test the scenario when there exists at least one point with negative coordinates.

Definition at line 137 of file MinEnclosingTriangleFinderTest.cpp.

**6.98.3.12 bool multiscaletest::MinEnclosingTriangleFinderTest::TestPointsWith-NegativeXCoordinate( )**

Test the scenario when there exists at least one point with negative x coordinate.

Definition at line 119 of file MinEnclosingTriangleFinderTest.cpp.

**6.98.3.13 bool multiscaletest::MinEnclosingTriangleFinderTest::TestPointsWith-NegativeYCoordinate( )**

Test the scenario when there exists at least one point with negative y coordinate.

Definition at line 128 of file MinEnclosingTriangleFinderTest.cpp.

**6.98.3.14 bool multiscaletest::MinEnclosingTriangleFinderTest::TestRandom-Points( )**

Test the scenario when randomly initialised vectors of input points are provided.

Definition at line 195 of file MinEnclosingTriangleFinderTest.cpp.

6.98.3.15 `bool multiscaletest::MinEnclosingTriangleFinderTest::TestThreePoints( )`

Test the scenario when only three input points are provided.

Definition at line 164 of file MinEnclosingTriangleFinderTest.cpp.

6.98.3.16 `bool multiscaletest::MinEnclosingTriangleFinderTest::TestTwoPoints( )`

Test the scenario when only two input points are provided.

Definition at line 155 of file MinEnclosingTriangleFinderTest.cpp.

6.98.3.17 `void multiscaletest::MinEnclosingTriangleFinderTest::ValidateTestResults( ) [override, protected, virtual]`

Check if the obtained results are valid.

Implements [multiscaletest::MultiscaleTest](#).

Definition at line 226 of file MinEnclosingTriangleFinderTest.cpp.

## 6.98.4 Member Data Documentation

6.98.4.1 `double multiscaletest::MinEnclosingTriangleFinderTest::area [protected]`

Area of the minimum enclosing triangle

Definition at line 22 of file MinEnclosingTriangleFinderTest.cpp.

6.98.4.2 `std::vector<cv::Point2f> multiscaletest::MinEnclosingTriangleFinderTest::convexHull [protected]`

Convex hull of the 2D point set

Definition at line 18 of file MinEnclosingTriangleFinderTest.cpp.

6.98.4.3 `const int multiscaletest::MinEnclosingTriangleFinderTest::MAX_NR_EXECUTIONS = 10000 [static, private]`

Definition at line 90 of file MinEnclosingTriangleFinderTest.cpp.

---

6.98.4.4 **const int multiscaletest::MinEnclosingTriangleFinderTest::MAX\_NR\_POINTS = 10000** [static, private]

Definition at line 88 of file MinEnclosingTriangleFinderTest.cpp.

6.98.4.5 **const int multiscaletest::MinEnclosingTriangleFinderTest::MIN\_NR\_EXECUTIONS = 5000** [static, private]

Definition at line 89 of file MinEnclosingTriangleFinderTest.cpp.

6.98.4.6 **const int multiscaletest::MinEnclosingTriangleFinderTest::MIN\_NR\_POINTS = 1** [static, private]

Definition at line 87 of file MinEnclosingTriangleFinderTest.cpp.

6.98.4.7 **const double multiscaletest::MinEnclosingTriangleFinderTest::POINT\_IN\_TRIANGLE\_THRESH = 1E-4** [static, private]

Definition at line 92 of file MinEnclosingTriangleFinderTest.cpp.

6.98.4.8 **std::vector<cv::Point2f> multiscaletest::MinEnclosingTriangleFinderTest::points** [protected]

Collection of 2D points

Definition at line 20 of file MinEnclosingTriangleFinderTest.cpp.

6.98.4.9 **std::vector<cv::Point2f> multiscaletest::MinEnclosingTriangleFinderTest::triangle** [protected]

Minimum enclosing triangle

Definition at line 21 of file MinEnclosingTriangleFinderTest.cpp.

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

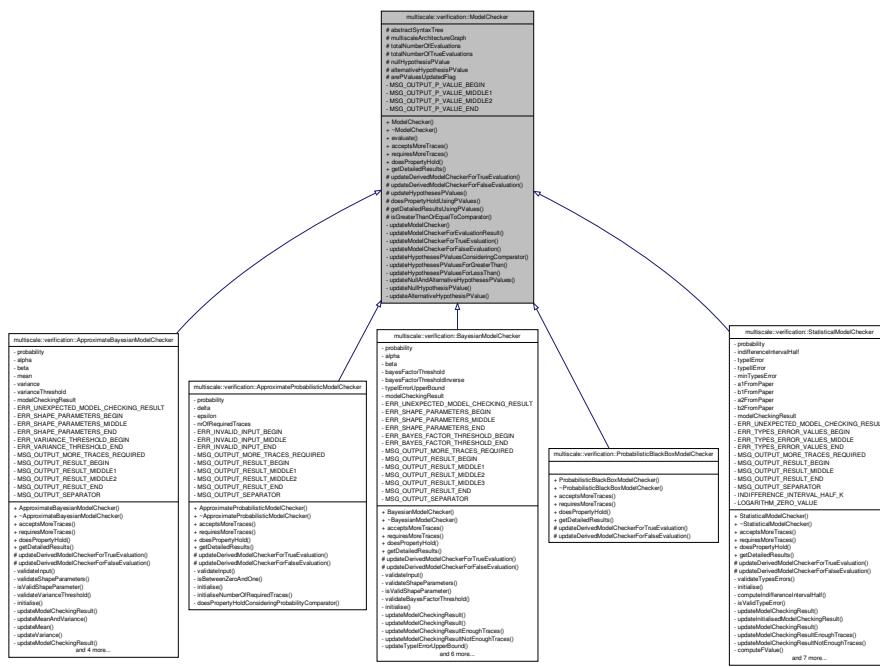
- MinEnclosingTriangleFinderTest.cpp

## 6.99 multiscale::verification::ModelChecker Class Reference

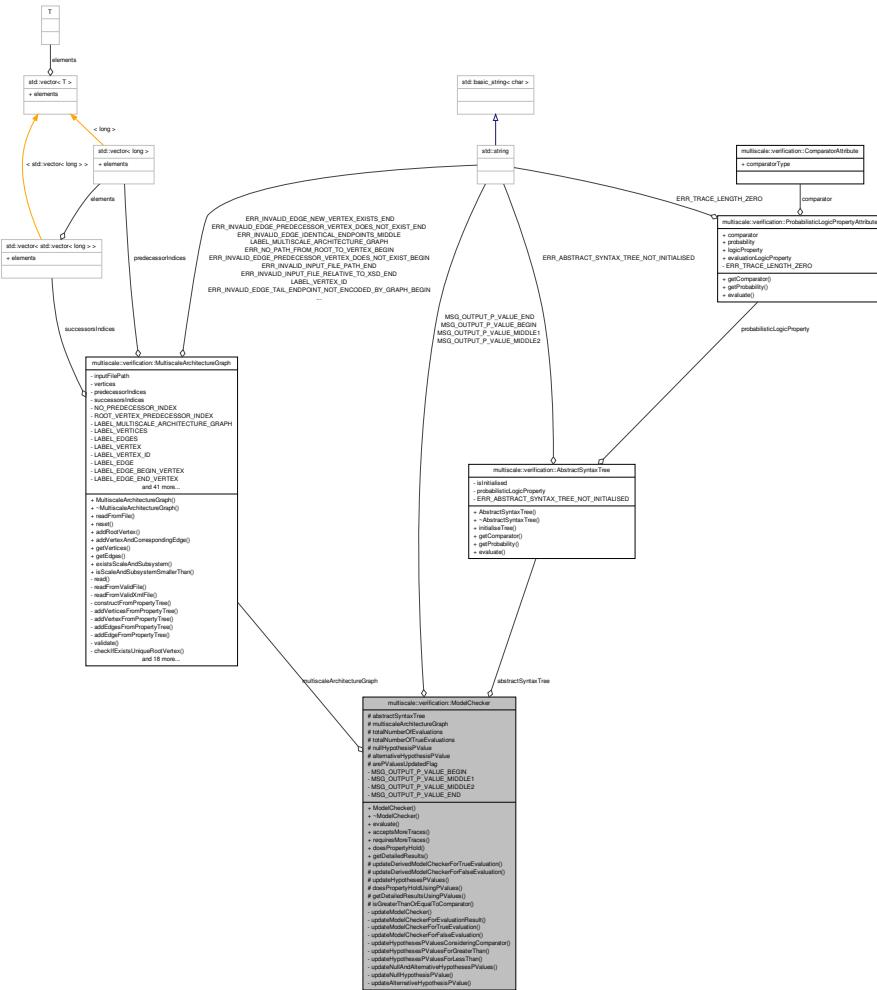
Abstract class representing a generic model checker.

```
#include <ModelChecker.hpp>
```

## Inheritance diagram for multiscale::verification::ModelChecker:



Collaboration diagram for multiscale::verification::ModelChecker:



## Public Member Functions

- `ModelChecker (const AbstractSyntaxTree &abstractSyntaxTree, const MultiscaleArchitectureGraph &multiscaleArchitectureGraph)`
- `virtual ~ModelChecker ()`
- `bool evaluate (const SpatialTemporalTrace &trace)`

*Evaluate the abstract syntax tree for the given trace and return the result.*

- `virtual bool acceptsMoreTraces ()=0`

*Check if more traces are accepted for evaluating the logic property.*

- `virtual bool requiresMoreTraces ()=0`

*Check if more traces are required for evaluating the logic property.*

- virtual bool `doesPropertyHold ()=0`  
*Check if the given property holds.*
- virtual std::string `getDetailedResults ()=0`  
*Get a detailed report of the results.*

## Protected Member Functions

- virtual void `updateDerivedModelCheckerForTrueEvaluation ()=0`  
*Update the results of the derived model checker type considering that the logic property was evaluated to true for the last trace.*
- virtual void `updateDerivedModelCheckerForFalseEvaluation ()=0`  
*Update the results of the derived model checker type considering that the logic property was evaluated to false for the last trace.*
- void `updateHypothesesPValues ()`  
*Update the p-values for the null and alternative hypothesis.*
- bool `doesPropertyHoldUsingPValues ()`  
*Check if the property holds considering the given p-values.*
- std::string `getDetailedResultsUsingPValues ()`  
*Get the detailed results when deciding if the property holds based on p-values.*
- bool `isGreaterThanOrEqualToComparator ()`  
*Check if the comparator used by the probabilistic logic property is greater than or equal to.*

## Protected Attributes

- `AbstractSyntaxTree abstractSyntaxTree`
- `MultiscaleArchitectureGraph multiscaleArchitectureGraph`
- unsigned int `totalNumberOfEvaluations`
- unsigned int `totalNumberOfTrueEvaluations`
- double `nullHypothesisPValue`
- double `alternativeHypothesisPValue`
- bool `arePValuesUpdatedFlag`

## Private Member Functions

- void `updateModelChecker (bool evaluationResult)`  
*Update the model checker results considering that the logic property was evaluated to evaluationResult for the last trace.*
- void `updateModelCheckerForEvaluationResult (bool evaluationResult)`  
*Update the model checker results considering that the logic property was evaluated to evaluationResult for the last trace.*
- void `updateModelCheckerForTrueEvaluation ()`  
*Update the results of the model checker considering that the logic property was evaluated to true for the last trace.*

- void [updateModelCheckerForFalseEvaluation \(\)](#)  
*Update the results of the model checker considering that the logic property was evaluated to false for the last trace.*
- void [updateHypothesesPValuesConsideringComparator \(\)](#)  
*Update the p-values for the null and alternative hypothesis considering the comparator contained by the probabilistic logic property.*
- void [updateHypothesesPValuesForGreaterThan \(\)](#)  
*Update the p-values considering that the probabilistic logic property is of the form  $P > [=]\theta[\phi]$ .*
- void [updateHypothesesPValuesForLessThan \(\)](#)  
*Update the p-values considering that the probabilistic logic property is of the form  $P < [=]\theta[\phi]$ .*
- void [updateNullAndAlternativeHypothesesPValues \(unsigned int nrOfEvaluations, unsigned int nrOfSuccesses, double probability\)](#)  
*Update the null and alternative hypotheses p-values.*
- void [updateNullHypothesisPValue \(unsigned int nrOfEvaluations, unsigned int nrOfSuccesses, double probability\)](#)  
*Update the null hypothesis p-value.*
- void [updateAlternativeHypothesisPValue \(unsigned int nrOfEvaluations, unsigned int nrOfSuccesses, double probability\)](#)  
*Update the alternative hypothesis p-value.*

## Static Private Attributes

- static const std::string [MSG\\_OUTPUT\\_P\\_VALUE\\_BEGIN](#) = "The confidence level of the answer expressed as a p-value (lower is better): "
- static const std::string [MSG\\_OUTPUT\\_P\\_VALUE\\_MIDDLE1](#) = " (p-value H0: "
- static const std::string [MSG\\_OUTPUT\\_P\\_VALUE\\_MIDDLE2](#) = ", p-value H1: "
- static const std::string [MSG\\_OUTPUT\\_P\\_VALUE\\_END](#) = ")"

### 6.99.1 Detailed Description

Abstract class representing a generic model checker.

Definition at line 14 of file ModelChecker.hpp.

### 6.99.2 Constructor & Destructor Documentation

- #### 6.99.2.1 multiscale::verification::ModelChecker::ModelChecker
- ```
( const AbstractSyntaxTree & abstractSyntaxTree, const
      MultiscaleArchitectureGraph & multiscaleArchitectureGraph ) [inline]
```

Definition at line 37 of file ModelChecker.hpp.

6.99.2.2 **virtual multiscale::verification::ModelChecker::~ModelChecker( )**  
[inline, virtual]

Definition at line 46 of file ModelChecker.hpp.

### 6.99.3 Member Function Documentation

6.99.3.1 **virtual bool multiscale::verification::ModelChecker::acceptsMoreTraces( )** [pure virtual]

Check if more traces are accepted for evaluating the logic property.

Implemented in [multiscale::verification::StatisticalModelChecker](#), [multiscale::verification::ApproximateBayesianModelChecker](#), [multiscale::verification::BayesianModelChecker](#), [multiscale::verification::ApproximateProbabilisticModelChecker](#), and [multiscale::verification::ProbabilisticBlackBoxModelChecker](#).

6.99.3.2 **virtual bool multiscale::verification::ModelChecker::doesPropertyHold( )**  
[pure virtual]

Check if the given property holds.

Implemented in [multiscale::verification::StatisticalModelChecker](#), [multiscale::verification::ApproximateBayesianModelChecker](#), [multiscale::verification::BayesianModelChecker](#), [multiscale::verification::ApproximateProbabilisticModelChecker](#), and [multiscale::verification::ProbabilisticBlackBoxModelChecker](#).

6.99.3.3 **bool ModelChecker::doesPropertyHoldUsingPValues( )**  
[protected]

Check if the property holds considering the given p-values.

Definition at line 24 of file ModelChecker.cpp.

References [alternativeHypothesisPValue](#), [nullHypothesisPValue](#), and [updateHypothesesPValues\(\)](#).

Referenced by [multiscale::verification::ProbabilisticBlackBoxModelChecker::doesPropertyHold\(\)](#), [multiscale::verification::ApproximateProbabilisticModelChecker::doesPropertyHold\(\)](#), [multiscale::verification::ApproximateBayesianModelChecker::doesPropertyHoldConsideringResult\(\)](#), [multiscale::verification::BayesianModelChecker::doesPropertyHoldConsideringResult\(\)](#), and [multiscale::verification::StatisticalModelChecker::doesPropertyHoldConsideringResult\(\)](#).

6.99.3.4 **bool ModelChecker::evaluate( const SpatialTemporalTrace & trace )**

Evaluate the abstract syntax tree for the given trace and return the result.

**Parameters**

|              |                                  |
|--------------|----------------------------------|
| <i>trace</i> | The given spatial temporal trace |
|--------------|----------------------------------|

Definition at line 8 of file ModelChecker.cpp.

References abstractSyntaxTree, multiscale::verification::AbstractSyntaxTree::evaluate(), multiscaleArchitectureGraph, and updateModelChecker().

**6.99.3.5 virtual std::string multiscale::verification::ModelChecker::getDetailedResults( ) [pure virtual]**

Get a detailed report of the results.

Implemented in [multiscale::verification::StatisticalModelChecker](#), [multiscale::verification::ApproximateBayesianModelChecker](#), [multiscale::verification::BayesianModelChecker](#), [multiscale::verification::ApproximateProbabilisticModelChecker](#), and [multiscale::verification::ProbabilisticBlackBoxModelChecker](#).

**6.99.3.6 std::string ModelChecker::getDetailedResultsUsingPValues( ) [protected]**

Get the detailed results when deciding if the property holds based on p-values.

Definition at line 30 of file ModelChecker.cpp.

References alternativeHypothesisPValue, MSG\_OUTPUT\_P\_VALUE\_BEGIN, MSG\_OUTPUT\_P\_VALUE\_END, MSG\_OUTPUT\_P\_VALUE\_MIDDLE1, MSG\_OUTPUT\_P\_VALUE\_MIDDLE2, nullHypothesisPValue, multiscale::StringManipulator::toString(), and updateHypothesesPValues().

Referenced by [multiscale::verification::ProbabilisticBlackBoxModelChecker::getDetailedResults\(\)](#), [multiscale::verification::ApproximateProbabilisticModelChecker::getDetailedResults\(\)](#), [multiscale::verification::ApproximateBayesianModelChecker::getDetailedUpdatedResults\(\)](#), [multiscale::verification::BayesianModelChecker::getDetailedUpdatedResults\(\)](#), and [multiscale::verification::StatisticalModelChecker::getDetailedUpdatedResults\(\)](#).

**6.99.3.7 bool ModelChecker::isGreaterThanOrEqualToComparator( ) [protected]**

Check if the comparator used by the probabilistic logic property is greater than or equal to.

Definition at line 42 of file ModelChecker.cpp.

References abstractSyntaxTree, and [multiscale::verification::AbstractSyntaxTree::getComparator\(\)](#).

Referenced by [multiscale::verification::ApproximateProbabilisticModelChecker::doesPropertyHoldConsideringProbabilityComparator\(\)](#), [multiscale::verification::BayesianModelChecker::doesPropertyHoldConsideringProbabilityComparator\(\)](#), [multiscale::verification::ApproximateBayesianModelChecker::doesPropertyHoldConsideringProbabilityComparator\(\)](#), and [multiscale::verification::BayesianModelChecker::doesPropertyHoldConsideringProbabilityComparator\(\)](#).

::verification::StatisticalModelChecker::doesPropertyHoldConsideringProbabilityComparator(), multiscale::verification::ApproximateBayesianModelChecker::isModelCheckingResultTrueConsideringComparator(), and updateHypothesesPValuesConsideringComparator().

#### 6.99.3.8 virtual bool multiscale::verification::ModelChecker::requiresMoreTraces ( ) [pure virtual]

Check if more traces are required for evaluating the logic property.

Implemented in [multiscale::verification::StatisticalModelChecker](#), [multiscale::verification::ApproximateBayesianModelChecker](#), [multiscale::verification::BayesianModelChecker](#), [multiscale::verification::ApproximateProbabilisticModelChecker](#), and [multiscale::verification::ProbabilisticBlackBoxModelChecker](#).

#### 6.99.3.9 void ModelChecker::updateAlternativeHypothesisPValue ( unsigned int nrOfEvaluations, unsigned int nrOfSuccesses, double probability ) [private]

Update the alternative hypothesis p-value.

##### Parameters

|                        |                                                 |
|------------------------|-------------------------------------------------|
| <i>nrOfEvaluations</i> | The number of evaluations                       |
| <i>nrOfSuccesses</i>   | The number of true evaluations                  |
| <i>probability</i>     | The probability specified in the logic property |

Definition at line 115 of file ModelChecker.cpp.

References [alternativeHypothesisPValue](#), and [multiscale::BinomialDistribution::cdf\(\)](#).

Referenced by [updateNullAndAlternativeHypothesesPValues\(\)](#).

#### 6.99.3.10 virtual void multiscale::verification::ModelChecker::updateDerivedModelCheckerForFalseEvaluation ( ) [protected, pure virtual]

Update the results of the derived model checker type considering that the logic property was evaluated to false for the last trace.

Implemented in [multiscale::verification::StatisticalModelChecker](#), [multiscale::verification::ApproximateBayesianModelChecker](#), [multiscale::verification::BayesianModelChecker](#), [multiscale::verification::ApproximateProbabilisticModelChecker](#), and [multiscale::verification::ProbabilisticBlackBoxModelChecker](#).

**6.99.3.11 virtual void multiscale::verification::ModelChecker::updateDerivedModelCheckerForTrueEvaluation( ) [protected, pure virtual]**

Update the results of the derived model checker type considering that the logic property was evaluated to true for the last trace.

Implemented in [multiscale::verification::StatisticalModelChecker](#), [multiscale::verification::ApproximateBayesianModelChecker](#), [multiscale::verification::BayesianModelChecker](#), [multiscale::verification::ApproximateProbabilisticModelChecker](#), and [multiscale::verification::ProbabilisticBlackBoxModelChecker](#).

**6.99.3.12 void ModelChecker::updateHypothesesPValues( ) [protected]**

Update the p-values for the null and alternative hypothesis.

The method for updating the p-values is based on considering that each trace is represented by a Bernoulli variable which can be either true or false with respect to the given logic property.

The probability distribution of a sum of n Bernoulli variables (where n = number of traces) is a binomial distribution. Using the cumulative distribution function the p-values of the hypotheses can be computed.

More details are given in the following paper: H. L. S. Younes, ‘Probabilistic Verification for “Black-Box” Systems’, in Computer Aided Verification, K. Etessami and S. K. - Rajamani, Eds. Springer Berlin Heidelberg, 2005, pp. 253–265.

Definition at line 16 of file ModelChecker.cpp.

References `arePValuesUpdatedFlag`, and `updateHypothesesPValuesConsideringComparator()`.

Referenced by `doesPropertyHoldUsingPValues()`, and `getDetailedResultsUsingPValues()`.

**6.99.3.13 void ModelChecker::updateHypothesesPValuesConsideringComparator( ) [private]**

Update the p-values for the null and alternative hypothesis considering the comparator contained by the probabilistic logic property.

Definition at line 74 of file ModelChecker.cpp.

References `isGreaterThanOrEqualToComparator()`, `updateHypothesesPValuesForGreaterThanOrEqual()`, and `updateHypothesesPValuesForLessThan()`.

Referenced by `updateHypothesesPValues()`.

**6.99.3.14 void ModelChecker::updateHypothesesPValuesForGreaterThan( )**  
 [private]

Update the p-values considering that the probabilistic logic property is of the form  $P > [=]\theta[\phi]$ .

$p-value_{H_0} = 1 - F(d - 1; n, \theta)$   $p-value_{H_1} = F(d; n, \theta)$  where d = number of true evaluations, n = number of evaluations and  $\theta$  = probability specified in the logic property

Definition at line 82 of file ModelChecker.cpp.

References abstractSyntaxTree, multiscale::verification::AbstractSyntaxTree::getProbability(), totalNumberOfEvaluations, totalNumberOfTrueEvaluations, and updateNullAndAlternativeHypothesesPValues().

Referenced by updateHypothesesPValuesConsideringComparator().

**6.99.3.15 void ModelChecker::updateHypothesesPValuesForLessThan( )**  
 [private]

Update the p-values considering that the probabilistic logic property is of the form  $P < [=]\theta[\phi]$ .

$p-value_{H_0} = 1 - F(d' - 1; n, \theta)$   $p-value_{H_1} = F(d'; n, \theta)$  where  $d' = n - d$ , d = number of true evaluations, n = number of evaluations and  $\theta$  = probability specified in the logic property

Definition at line 90 of file ModelChecker.cpp.

References abstractSyntaxTree, multiscale::verification::AbstractSyntaxTree::getProbability(), totalNumberOfEvaluations, totalNumberOfTrueEvaluations, and updateNullAndAlternativeHypothesesPValues().

Referenced by updateHypothesesPValuesConsideringComparator().

**6.99.3.16 void ModelChecker::updateModelChecker( bool evaluationResult )**  
 [private]

Update the model checker results considering that the logic property was evaluated to evaluationResult for the last trace.

**Parameters**

|                          |                                                                        |
|--------------------------|------------------------------------------------------------------------|
| <i>evaluation-Result</i> | The result of evaluating the logic property considering the last trace |
|--------------------------|------------------------------------------------------------------------|

Definition at line 51 of file ModelChecker.cpp.

References arePValuesUpdatedFlag, and updateModelCheckerForEvaluationResult().

Referenced by evaluate().

---

**6.99.3.17 void ModelChecker::updateModelCheckerForEvaluationResult ( bool evaluationResult ) [private]**

Update the model checker results considering that the logic property was evaluated to evaluationResult for the last trace.

**Parameters**

|                          |                                                                        |
|--------------------------|------------------------------------------------------------------------|
| <i>evaluation-Result</i> | The result of evaluating the logic property considering the last trace |
|--------------------------|------------------------------------------------------------------------|

Definition at line 57 of file ModelChecker.cpp.

References updateModelCheckerForFalseEvaluation(), and updateModelCheckerForTrueEvaluation().

Referenced by updateModelChecker().

**6.99.3.18 void ModelChecker::updateModelCheckerForFalseEvaluation ( ) [private]**

Update the results of the model checker considering that the logic property was evaluated to false for the last trace.

Definition at line 70 of file ModelChecker.cpp.

References totalNumberOfEvaluations.

Referenced by updateModelCheckerForEvaluationResult().

**6.99.3.19 void ModelChecker::updateModelCheckerForTrueEvaluation ( ) [private]**

Update the results of the model checker considering that the logic property was evaluated to true for the last trace.

Definition at line 65 of file ModelChecker.cpp.

References totalNumberOfEvaluations, and totalNumberOfTrueEvaluations.

Referenced by updateModelCheckerForEvaluationResult().

**6.99.3.20 void ModelChecker::updateNullAndAlternativeHypothesesPValues ( unsigned int nrOfEvaluations, unsigned int nrOfSuccesses, double probability ) [private]**

Update the null and alternative hypotheses p-values.

**Parameters**

|                         |                           |
|-------------------------|---------------------------|
| <i>nrOf-Evaluations</i> | The number of evaluations |
|-------------------------|---------------------------|

|                      |                                                 |
|----------------------|-------------------------------------------------|
| <i>nrOfSuccesses</i> | The number of true evaluations                  |
| <i>probability</i>   | The probability specified in the logic property |

Definition at line 98 of file ModelChecker.cpp.

References updateAlternativeHypothesisPValue(), and updateNullHypothesisPValue().

Referenced by updateHypothesesPValuesForGreaterThan(), and updateHypothesesPValuesForLessThan().

#### 6.99.3.21 void ModelChecker::updateNullHypothesisPValue ( unsigned int *nrOfEvaluations*, unsigned int *nrOfSuccesses*, double *probability* ) [private]

Update the null hypothesis p-value.

##### Parameters

|                        |                                                 |
|------------------------|-------------------------------------------------|
| <i>nrOfEvaluations</i> | The number of evaluations                       |
| <i>nrOfSuccesses</i>   | The number of true evaluations                  |
| <i>probability</i>     | The probability specified in the logic property |

Definition at line 105 of file ModelChecker.cpp.

References multiscale::BinomialDistribution::cdf(), and nullHypothesisPValue().

Referenced by updateNullAndAlternativeHypothesesPValues().

#### 6.99.4 Member Data Documentation

##### 6.99.4.1 AbstractSyntaxTree multiscale::verification::ModelChecker::abstractSyntaxTree [protected]

The abstract syntax tree representing the logic property which this model checker instance evaluates

Definition at line 18 of file ModelChecker.hpp.

Referenced by evaluate(), multiscale::verification::ApproximateProbabilisticModelChecker::initialise(), multiscale::verification::StatisticalModelChecker::initialise(), multiscale::verification::BayesianModelChecker::initialise(), multiscale::verification::ApproximateBayesianModelChecker::initialise(), isGreaterThanOrEqualToComparator(), updateHypothesesPValuesForGreaterThan(), and updateHypothesesPValuesForLessThan().

6.99.4.2 **double multiscale::verification::ModelChecker::alternativeHypothesisP-Value** [protected]

The p-value for the alternative hypothesis to hold

Definition at line 31 of file ModelChecker.hpp.

Referenced by doesPropertyHoldUsingPValues(), getDetailedResultsUsingPValues(), and updateAlternativeHypothesisPValue().

6.99.4.3 **bool multiscale::verification::ModelChecker::arePValuesUpdatedFlag** [protected]

Flag indicating if the p-values were updated

Definition at line 33 of file ModelChecker.hpp.

Referenced by updateHypothesesPValues(), and updateModelChecker().

6.99.4.4 **const std::string ModelChecker::MSG\_OUTPUT\_P\_VALUE\_BEGIN = "The confidence level of the answer expressed as a p-value (lower is better): "** [static, private]

Definition at line 170 of file ModelChecker.hpp.

Referenced by getDetailedResultsUsingPValues().

6.99.4.5 **const std::string ModelChecker::MSG\_OUTPUT\_P\_VALUE\_END = ")"** [static, private]

Definition at line 173 of file ModelChecker.hpp.

Referenced by getDetailedResultsUsingPValues().

6.99.4.6 **const std::string ModelChecker::MSG\_OUTPUT\_P\_VALUE\_MIDDLE1 = "(p-value H0: "** [static, private]

Definition at line 171 of file ModelChecker.hpp.

Referenced by getDetailedResultsUsingPValues().

6.99.4.7 **const std::string ModelChecker::MSG\_OUTPUT\_P\_VALUE\_MIDDLE2 = ", p-value H1: "** [static, private]

Definition at line 172 of file ModelChecker.hpp.

Referenced by getDetailedResultsUsingPValues().

**6.99.4.8 MultiscaleArchitectureGraph multiscale::verification-  
::ModelChecker::multiscaleArchitectureGraph  
[protected]**

The multiscale architecture graph encoding the hierarchical organization of scales and subsystems

Definition at line 23 of file ModelChecker.hpp.

Referenced by evaluate().

**6.99.4.9 double multiscale::verification::ModelChecker::nullHypothesisPValue  
[protected]**

The p-value for the null hypothesis to hold

Definition at line 30 of file ModelChecker.hpp.

Referenced by doesPropertyHoldUsingPValues(), getDetailedResultsUsingPValues(), and updateNullHypothesisPValue().

**6.99.4.10 unsigned int multiscale::verification::ModelChecker::totalNumberOf-  
Evaluations [protected]**

The total number of evaluations

Definition at line 26 of file ModelChecker.hpp.

Referenced by multiscale::verification::ApproximateProbabilisticModelChecker::acceptsMoreTraces(), multiscale::verification::BayesianModelChecker::computeBinomialPDF(), multiscale::verification::StatisticalModelChecker::computeFPrimeValue(), multiscale::verification::StatisticalModelChecker::computeFValue(), multiscale::verification::BayesianModelChecker::computeMaximumBinomialPDF(), multiscale::verification::ApproximateProbabilisticModelChecker::doesPropertyHoldConsideringProbabilityComparator(), multiscale::verification::BayesianModelChecker::indicatorFunction(), updateHypothesesPValuesForGreaterThan(), updateHypothesesPValuesForLessThan(), multiscale::verification::ApproximateBayesianModelChecker::updateMean(), updateModelCheckerForFalseEvaluation(), updateModelCheckerForTrueEvaluation(), multiscale::verification::BayesianModelChecker::updateModelCheckingResult(), multiscale::verification::BayesianModelChecker::updateTypeIErrorUpperBound(), and multiscale::verification::ApproximateBayesianModelChecker::updateVariance().

**6.99.4.11 unsigned int multiscale::verification::ModelChecker::totalNumberOfTrue-  
Evaluations [protected]**

The total number of times the abstract syntax tree was evaluated to true

Definition at line 27 of file ModelChecker.hpp.

Referenced by multiscale::verification::StatisticalModelChecker::computeFPrime-

`Value()`, `multiscale::verification::StatisticalModelChecker::computeFValue()`, `multiscale::verification::ApproximateProbabilisticModelChecker::doesPropertyHoldConsideringProbabilityComparator()`, `updateHypothesesPValuesForGreaterThan()`, `updateHypothesesPValuesForLessThan()`, `multiscale::verification::ApproximateBayesianModelChecker::updateMean()`, `updateModelCheckerForTrueEvaluation()`, `multiscale::verification::BayesianModelChecker::updateModelCheckingResult()`, and `multiscale::verification::ApproximateBayesianModelChecker::updateVariance()`.

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

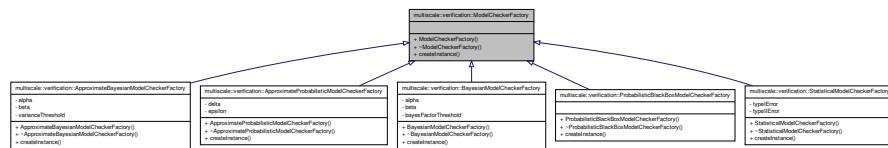
- `ModelChecker.hpp`
- `ModelChecker.cpp`

## 6.100 multiscale::verification::ModelCheckerFactory Class - Reference

Interface for different model checker factories.

```
#include <ModelCheckerFactory.hpp>
```

Inheritance diagram for `multiscale::verification::ModelCheckerFactory`:



### Public Member Functions

- `ModelCheckerFactory ()`
- `virtual ~ModelCheckerFactory ()`
- `virtual std::shared_ptr < ModelChecker > createInstance (const AbstractSyntaxTree &abstractSyntaxTree, const MultiscaleArchitectureGraph &multiscaleArchitectureGraph)=0`

*Create an instance of the model checker.*

#### 6.100.1 Detailed Description

Interface for different model checker factories.

Definition at line 15 of file `ModelCheckerFactory.hpp`.

### 6.100.2 Constructor & Destructor Documentation

6.100.2.1 `multiscale::verification::ModelCheckerFactory::ModelCheckerFactory( ) [inline]`

Definition at line 19 of file ModelCheckerFactory.hpp.

6.100.2.2 `virtual multiscale::verification::ModelCheckerFactory::~ModelCheckerFactory( ) [inline, virtual]`

Definition at line 20 of file ModelCheckerFactory.hpp.

### 6.100.3 Member Function Documentation

6.100.3.1 `virtual std::shared_ptr<ModelChecker> multiscale::verification::ModelCheckerFactory::createInstance( const AbstractSyntaxTree & abstractSyntaxTree, const MultiscaleArchitectureGraph & multiscaleArchitectureGraph ) [pure virtual]`

Create an instance of the model checker.

#### Parameters

|                                            |                                                                                                   |
|--------------------------------------------|---------------------------------------------------------------------------------------------------|
| <code>abstract-SyntaxTree</code>           | The abstract syntax tree representing the logic property to be checked                            |
| <code>multiscale-Architecture-Graph</code> | The multiscale architecture graph encoding the hierarchical organization of scales and subsystems |

Implemented in `multiscale::verification::ApproximateBayesianModelCheckerFactory`, `multiscale::verification::BayesianModelCheckerFactory`, `multiscale::verification::ApproximateProbabilisticModelCheckerFactory`, `multiscale::verification::StatisticalModelCheckerFactory`, and `multiscale::verification::ProbabilisticBlackBoxModelCheckerFactory`.

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

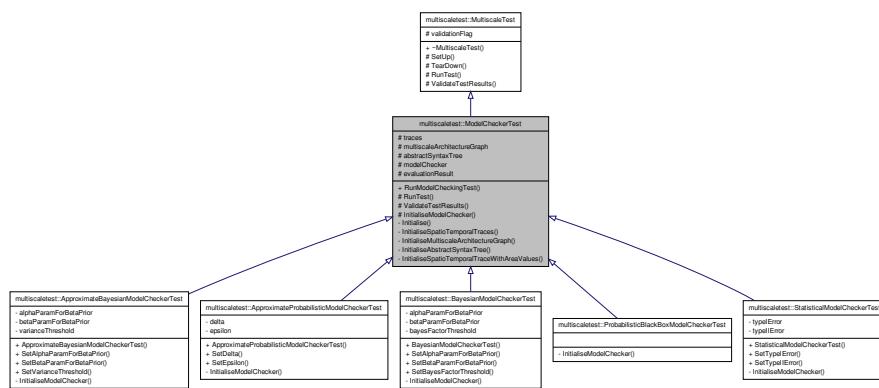
- `ModelCheckerFactory.hpp`

## 6.101 multiscaletest::ModelCheckerTest Class Reference

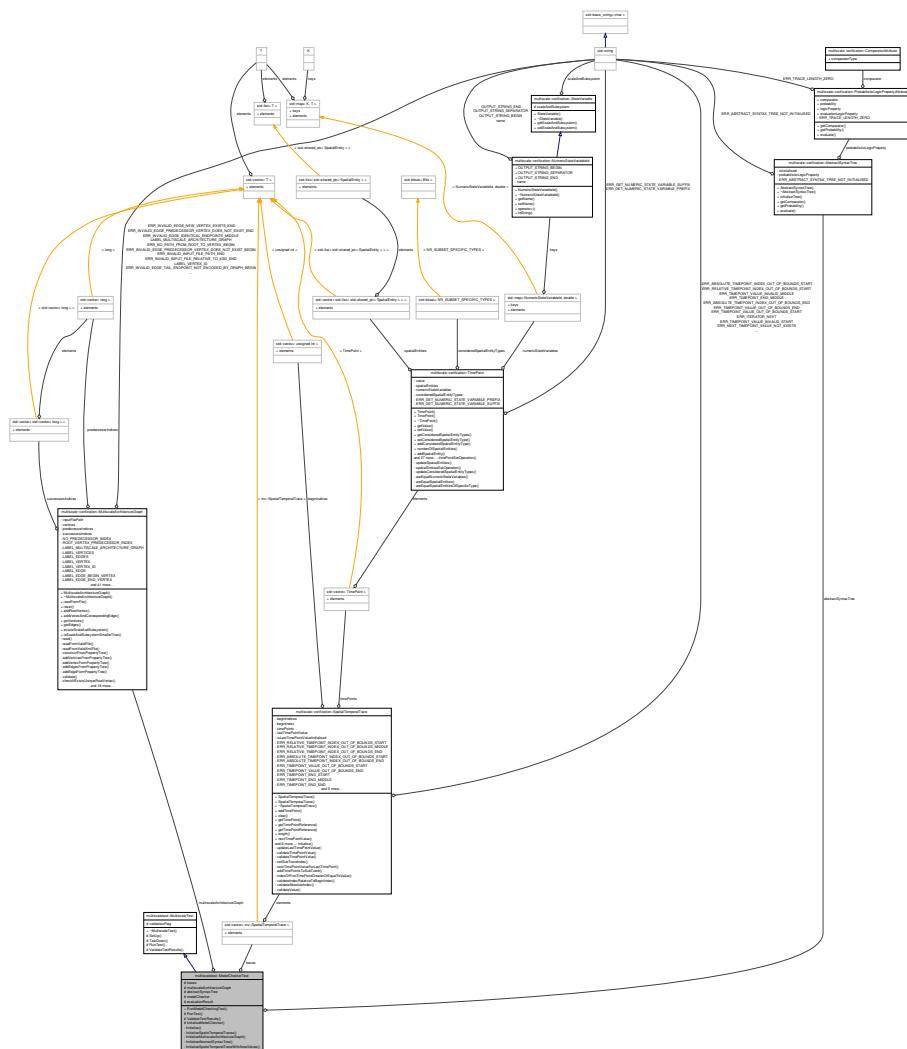
Class for testing model checkers.

```
#include <ModelCheckerTest.hpp>
```

Inheritance diagram for multiscaletest::ModelCheckerTest:



## Collaboration diagram for multiscaletest::ModelCheckerTest:



## Public Member Functions

- bool RunModelCheckingTest ()  
*Run the test for the given logic property.*

## Protected Member Functions

- virtual void `RunTest ()` override  
*Run the test.*
  - virtual void `ValidateTestResults ()` override

*Validate the results of the test.*

- virtual void `InitialiseModelChecker ()=0`

*Initialise the model checker.*

## Protected Attributes

- std::vector < `mv::SpatialTemporalTrace` > `traces`
- `MultiscaleArchitectureGraph` `multiscaleArchitectureGraph`
- `mv::AbstractSyntaxTree` `abstractSyntaxTree`
- std::shared\_ptr< `mv::ModelChecker` > `modelChecker`
- bool `evaluationResult`

## Private Member Functions

- void `Initialise ()`  
*Initialisation function.*
- void `InitialiseSpatioTemporalTraces ()`  
*Initialise the collection of spatio-temporal traces.*
- void `InitialiseMultiscaleArchitectureGraph ()`  
*Initialise the multiscale architecture graph.*
- void `InitialiseAbstractSyntaxTree ()`  
*Initialise the abstract syntax tree.*
- void `InitialiseSpatioTemporalTraceWithAreaValues (const std::vector< double > areaValues)`  
*Initialise the collection of spatio-temporal traces with the given spatial entity area values.*

### 6.101.1 Detailed Description

Class for testing model checkers.

Definition at line 30 of file ModelCheckerTest.hpp.

### 6.101.2 Member Function Documentation

#### 6.101.2.1 void multiscaletest::ModelCheckerTest::Initialise ( ) [private]

Initialisation function.

Definition at line 111 of file ModelCheckerTest.hpp.

References `InitialiseAbstractSyntaxTree()`, `InitialiseModelChecker()`, `InitialiseMultiscaleArchitectureGraph()`, and `InitialiseSpatioTemporalTraces()`.

Referenced by `RunModelCheckingTest()`.

6.101.2.2 void multiscaletest::ModelCheckerTest::InitialiseAbstractSyntaxTree( )  
[private]

Initialise the abstract syntax tree.

Definition at line 139 of file ModelCheckerTest.hpp.

References abstractSyntaxTree, and multiscale::verification::Parser::parse().

Referenced by Initialise().

6.101.2.3 virtual void multiscaletest::ModelCheckerTest::InitialiseModelChecker( )  
[protected, pure virtual]

Initialise the model checker.

Implemented in multiscaletest::ApproximateBayesianModelCheckerTest, multiscaletest::BayesianModelCheckerTest, multiscaletest::ApproximateProbabilisticModelCheckerTest, multiscaletest::StatisticalModelCheckerTest, and multiscaletest::ProbabilisticBlackBoxModelCheckerTest.

Referenced by Initialise().

6.101.2.4 void multiscaletest::ModelCheckerTest::InitialiseMultiscaleArchitecture-  
Graph( ) [private]

Initialise the multiscale architecture graph.

Definition at line 135 of file ModelCheckerTest.hpp.

Referenced by Initialise().

6.101.2.5 void multiscaletest::ModelCheckerTest::InitialiseSpatioTemporalTraces( )  
[private]

Initialise the collection of spatio-temporal traces.

Definition at line 118 of file ModelCheckerTest.hpp.

References InitialiseSpatioTemporalTraceWithAreaValues(), and traces.

Referenced by Initialise().

6.101.2.6 void multiscaletest::ModelCheckerTest::InitialiseSpatioTemporal-  
TraceWithAreaValues( const std::vector< double > areaValues )  
[private]

Initialise the collection of spatio-temporal traces with the given spatial entity area values.

The assumption is that each timepoint contains only one spatial entity of the same type.  
Therefore each area value corresponds to a different timepoint and spatial entity.

**Parameters**

|                         |                               |
|-------------------------|-------------------------------|
| <code>areaValues</code> | The collection of area values |
|-------------------------|-------------------------------|

Definition at line 145 of file ModelCheckerTest.hpp.

References multiscale::verification::TimePoint::addSpatialEntityAndType(), multiscale::verification::SpatialTemporalTrace::addTimePoint(), multiscale::verification::TimePoint::getSpatialEntitiesBeginIterator(), multiscale::verification::TimePoint::removeSpatialEntity(), multiscale::verification::TimePoint::setConsideredSpatialEntityType(), and traces.

Referenced by InitialiseSpatioTemporalTraces().

**6.101.2.7 bool multiscaletest::ModelCheckerTest::RunModelCheckingTest( )**

Run the test for the given logic property.

Definition at line 90 of file ModelCheckerTest.hpp.

References evaluationResult, Initialise(), RunTest(), and ValidateTestResults().

**6.101.2.8 void multiscaletest::ModelCheckerTest::RunTest( ) [override, protected, virtual]**

Run the test.

Implements [multiscaletest::MultiscaleTest](#).

Definition at line 99 of file ModelCheckerTest.hpp.

References evaluationResult, modelChecker, and traces.

Referenced by RunModelCheckingTest().

**6.101.2.9 void multiscaletest::ModelCheckerTest::ValidateTestResults( ) [override, protected, virtual]**

Validate the results of the test.

Implements [multiscaletest::MultiscaleTest](#).

Definition at line 109 of file ModelCheckerTest.hpp.

Referenced by RunModelCheckingTest().

**6.101.3 Member Data Documentation****6.101.3.1 mv::AbstractSyntaxTree multiscaletest::ModelCheckerTest::abstractSyntaxTree [protected]**

The abstract syntax tree corresponding to the logic property

Definition at line 42 of file ModelCheckerTest.hpp.

Referenced by InitialiseAbstractSyntaxTree().

**6.101.3.2 bool multiscaletest::ModelCheckerTest::evaluationResult  
[protected]**

The result of the model checking evaluation

Definition at line 47 of file ModelCheckerTest.hpp.

Referenced by RunModelCheckingTest(), and RunTest().

**6.101.3.3 std::shared\_ptr<mv::ModelChecker> multiscaletest::ModelCheckerTest-  
::modelChecker [protected]**

The specific type of model checker employed

Definition at line 44 of file ModelCheckerTest.hpp.

Referenced by RunTest().

**6.101.3.4 MultiscaleArchitectureGraph multiscaletest::-  
ModelCheckerTest::multiscaleArchitectureGraph  
[protected]**

The multiscale architecture graph encoding the hierarchical organization of the scales and subsystems

Definition at line 38 of file ModelCheckerTest.hpp.

**6.101.3.5 std::vector<mv::SpatialTemporalTrace> multiscaletest::ModelChecker-  
Test::traces [protected]**

The collection of spatio-temporal traces

Definition at line 35 of file ModelCheckerTest.hpp.

Referenced by InitialiseSpatioTemporalTraces(), InitialiseSpatioTemporalTraceWithAreaValues(), and RunTest().

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

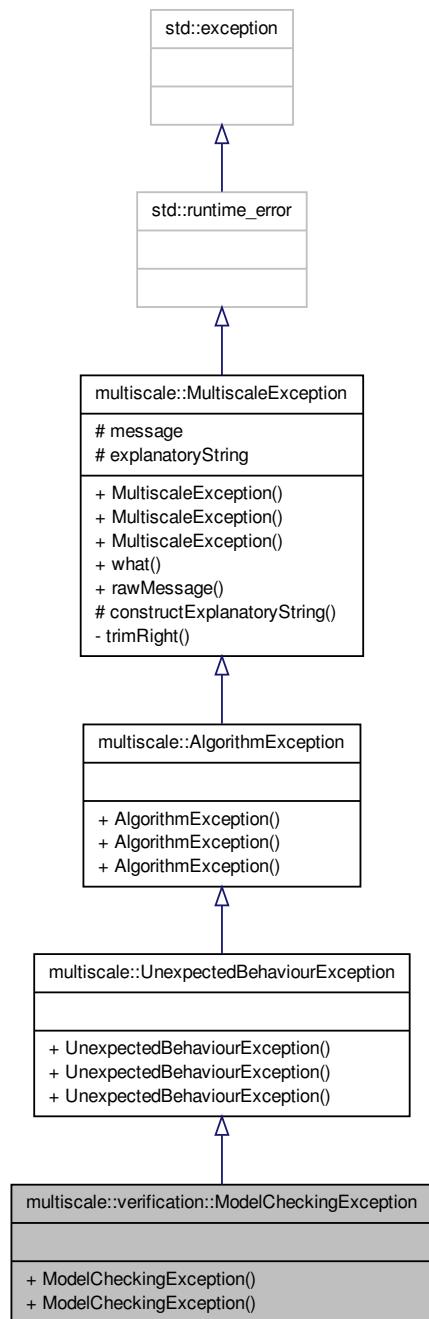
- ModelCheckerTest.hpp

**6.102 multiscale::verification::ModelCheckingException Class -  
Reference**

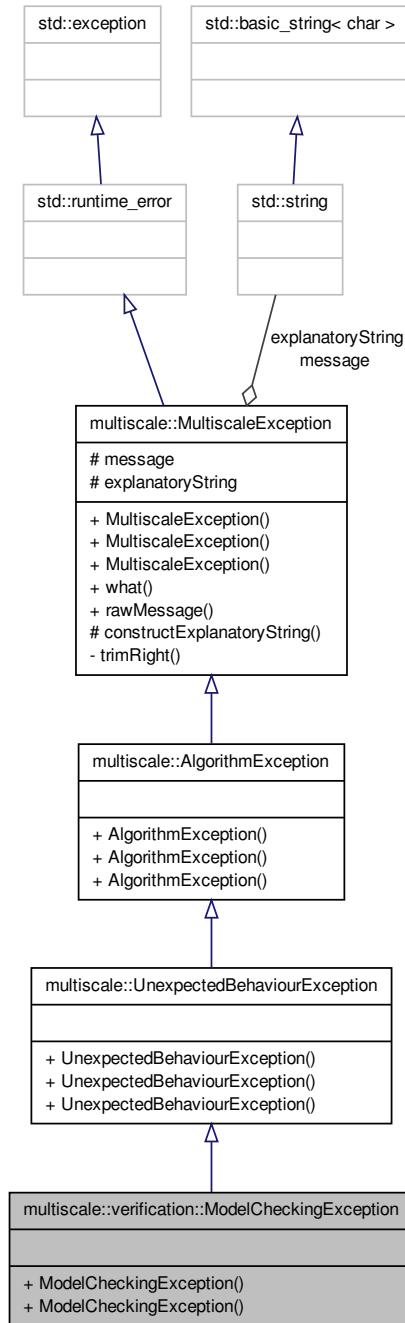
Class for representing a model checking exception.

```
#include <ModelCheckingException.hpp>
```

Inheritance diagram for multiscale::verification::ModelCheckingException:



Collaboration diagram for multiscale::verification::ModelCheckingException:



## Public Member Functions

- [ModelCheckingException](#) (const std::string &file, int line, const std::string &msg)
- [ModelCheckingException](#) (const std::string &file, int line, const char \*msg)

### 6.102.1 Detailed Description

Class for representing a model checking exception.

Definition at line 12 of file ModelCheckingException.hpp.

### 6.102.2 Constructor & Destructor Documentation

#### 6.102.2.1 multiscale::verification::ModelCheckingException::ModelCheckingException ( const std::string & file, int line, const std::string & msg ) [inline, explicit]

Definition at line 16 of file ModelCheckingException.hpp.

#### 6.102.2.2 multiscale::verification::ModelCheckingException::ModelCheckingException ( const std::string & file, int line, const char \* msg ) [inline, explicit]

Definition at line 21 of file ModelCheckingException.hpp.

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

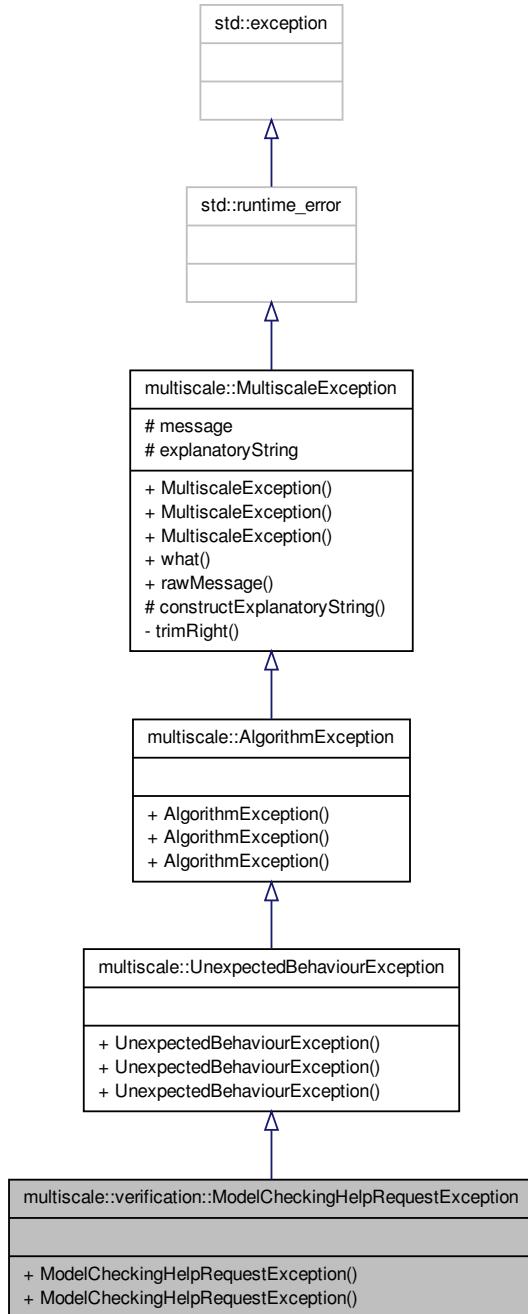
- ModelCheckingException.hpp

## 6.103 multiscale::verification::ModelCheckingHelpRequestException Class Reference

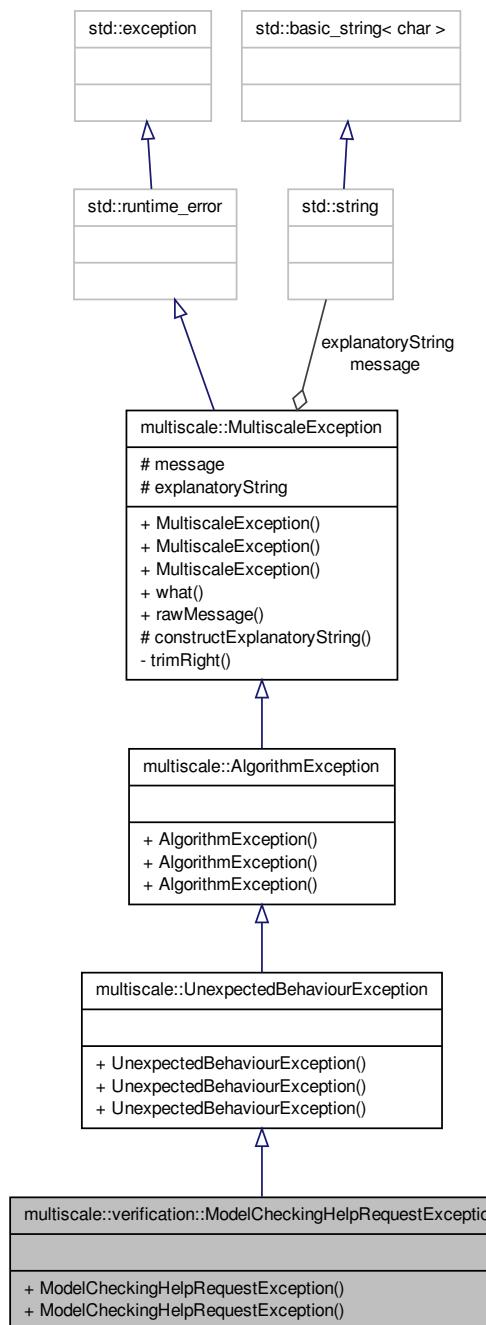
Class for representing a model checking help request exception.

```
#include <ModelCheckingHelpRequestException.hpp>
```

Inheritance diagram for multiscale::verification::ModelCheckingHelpRequestException:



Collaboration diagram for multiscale::verification::ModelCheckingHelpRequestException:



## Public Member Functions

- [ModelCheckingHelpRequestException](#) (const std::string &file, int line, const std::string &msg)
- [ModelCheckingHelpRequestException](#) (const std::string &file, int line, const char \*msg)

### 6.103.1 Detailed Description

Class for representing a model checking help request exception.

Definition at line 12 of file ModelCheckingHelpRequestException.hpp.

### 6.103.2 Constructor & Destructor Documentation

#### 6.103.2.1 multiscale::verification::ModelCheckingHelpRequestException::ModelCheckingHelpRequestException ( const std::string & file, int line, const std::string & msg ) [inline, explicit]

Definition at line 16 of file ModelCheckingHelpRequestException.hpp.

#### 6.103.2.2 multiscale::verification::ModelCheckingHelpRequestException::ModelCheckingHelpRequestException ( const std::string & file, int line, const char \* msg ) [inline, explicit]

Definition at line 21 of file ModelCheckingHelpRequestException.hpp.

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

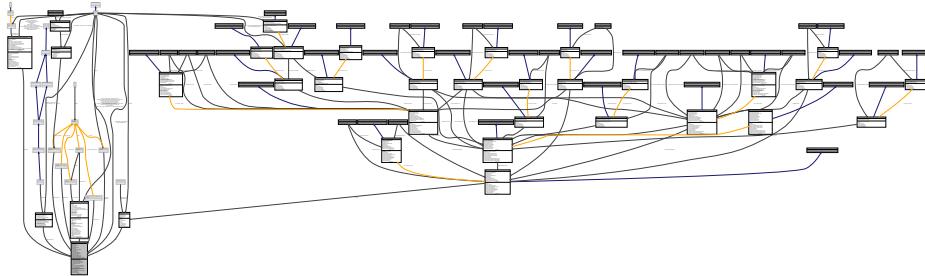
- [ModelCheckingHelpRequestException.hpp](#)

## 6.104 multiscale::verification::ModelCheckingManager Class - Reference

Class for managing the model checking processes.

```
#include <ModelCheckingManager.hpp>
```

Collaboration diagram for multiscale::verification::ModelCheckingManager:



## Public Member Functions

- `ModelCheckingManager` (const std::string &logicPropertiesFilepath, const std::string &tracesFolderPath, unsigned long extraEvaluationTime, const std::string &multiscaleArchitectureGraphfilepath)
- `~ModelCheckingManager` ()
- void `setExtraEvaluationProgramPath` (const std::string &`extraEvaluationProgramPath`)  
*Set the path of the program which should be executed whenever extra evaluation is required.*
- void `setShouldPrintDetailedEvaluation` (bool `shouldPrintDetailedEvaluation`)  
*Set the flag indicating if the detailed evaluation should be printed.*
- void `runModelCheckingTasks` (const std::shared\_ptr< `ModelCheckerFactory` > &modelCheckerFactory)  
*Run the model checking tasks.*

## Private Member Functions

- void `initialise` (const std::string &logicPropertiesFilepath, unsigned long extraEvaluationTime, const std::string &multiscaleArchitectureGraphfilepath)  
*Initialise the model checking manager.*
- void `initialiseExtraEvaluationTimeCounters` (unsigned long extraEvaluationTime)  
*Initialise the extra evaluation time counters.*
- void `initialiseMultiscaleArchitectureGraph` (const std::string &multiscaleArchitectureGraphfilepath)  
*Initialise the multiscale architecture graph.*
- void `initialiseLogicProperties` (const std::string &logicPropertiesFilepath)  
*Initialise the logic properties using the provided input file.*
- void `runModelCheckingAndOutputResults` (const std::shared\_ptr< `ModelCheckerFactory` > &modelCheckerFactory)  
*Run the model checking tasks and output the results.*
- void `parseLogicPropertiesAndPrintMessage` ()

- `void parseLogicProperties ()`  
*Parse the logic properties and print message informing the user about this.*
- `bool parseLogicPropertyAndPrintMessages (const std::string &logicProperty)`  
*Parse the logic property and inform the user if the logic property was syntactically correct.*
- `bool parseLogicProperty (const std::string &logicProperty)`  
*Parse the given logic property and return true if parsing was successful and false otherwise.*
- `bool isValidLogicProperty (const std::string &logicProperty)`  
*Parse the given logic property and return true if parsing was successful and false otherwise.*
- `void printParsingMessage (bool isParsingSuccessful)`  
*Print a message stating if the logic property was parsed successfully.*
- `void createModelCheckers (const std::shared_ptr< ModelCheckerFactory > &modelCheckerFactory)`  
*Create the model checker instances using the provided model checker factory.*
- `void runModelCheckersAndPrintMessage ()`  
*Run the model checkers and print a message informing the user about it.*
- `void runModelCheckers ()`  
*Run the model checkers and verify the logic properties.*
- `void runModelCheckersForCurrentlyExistingTraces ()`  
*Run the model checkers and verify the logic properties for the currently existing traces.*
- `SpatialTemporalTrace getNextSpatialTemporalTrace ()`  
*Get the next spatial temporal trace and store its path.*
- `void storeNewSpatialTemporalTracePath (const std::string &tracePath)`  
*Store new trace path if the shouldPrintDetailedEvaluation flag is set to true.*
- `void createNewEvaluationResults ()`  
*Create a new vector for storing the evaluation results for the (logic property, new trace) pairs.*
- `void runModelCheckersForTrace (const SpatialTemporalTrace &trace, bool &continueEvaluation)`  
*Run the model checkers and verify the logic properties considering the given trace.*
- `void runModelCheckerForTrace (const std::size_t &modelCheckerIndex, const - SpatialTemporalTrace &trace)`  
*Run the model checker for the given trace.*
- `void updateEvaluationResults (const std::size_t &modelCheckerIndex, bool evaluationResult)`  
*Update the evaluation results for the given model checker index and result.*
- `void runModelCheckersAndRequestAdditionalTraces ()`  
*Run the model checkers and request additional traces.*
- `void updateExtraEvaluationStartTime ()`  
*Set the extra evaluation start time equal to current time.*
- `bool isEvaluationTimeRemaining ()`

- *Check if there is evaluation time remaining.*  
 • bool [areUnfinishedModelCheckingTasks \(\)](#)
- *Check if there exist model checkers which require extra traces.*  
 • void [executeExtraEvaluationProgram \(\)](#)
- *Execute the extra evaluation program for generating potential new traces.*  
 • void [executeExtraEvaluationProgramAndPrintMessage \(\)](#)
- *Execute the extra evaluation program for generating potential new traces.*  
 • void [waitBeforeRetry \(\)](#)
- *Wait TRACE\_INPUT\_REFRESH\_TIMEOUT minutes before updating the trace reader.*  
 • void [updateTraceReader \(\)](#)
- *Update trace reader.*  
 • void [outputModelCheckersResultsAndPrintMessage \(\)](#)
- *Output the model checking results and print the message informing the user about this.*  
 • void [outputModelCheckersResults \(\)](#)
- *Output the model checking results.*  
 • void [outputModelCheckerResults \(const std::shared\\_ptr< ModelChecker > &modelChecker, const std::string &logicProperty\)](#)
- *Output the model checking results for the given model checker.*  
 • void [outputDetailedEvaluationResults \(\)](#)
- *Output the logic properties detailed evaluation results.*

### Private Attributes

- Parser parser
- std::vector< std::string > logicProperties
- std::vector< AbstractSyntaxTree > abstractSyntaxTrees
- std::vector< std::string > tracesPaths
- LogicPropertyDataReader logicPropertyReader
- SpatialTemporalDataReader traceReader
- MultiscaleArchitectureGraph multiscaleArchitectureGraph
- std::vector< std::vector< bool > > evaluationResults
- std::vector< std::shared\_ptr < ModelChecker > > modelCheckers
- std::chrono::time\_point < std::chrono::system\_clock > extraEvaluationStartTime
- double extraEvaluationElapsedTime
- unsigned long extraEvaluationTime
- std::string extraEvaluationProgramPath
- bool shouldPrintDetailedEvaluation

### Static Private Attributes

- static const unsigned long TRACE\_INPUT\_REFRESH\_TIMEOUT = 30
- static const std::string PARSER\_EMPTY\_LOGIC\_PROPERTY = ""

### 6.104.1 Detailed Description

Class for managing the model checking processes.

Definition at line 25 of file ModelCheckingManager.hpp.

### 6.104.2 Constructor & Destructor Documentation

**6.104.2.1 ModelCheckingManager::ModelCheckingManager ( const std::string & *logicPropertiesFilepath*, const std::string & *tracesFolderPath*, unsigned long *extraEvaluationTime*, const std::string & *multiscaleArchitectureGraphFilepath* )**

Definition at line 12 of file ModelCheckingManager.cpp.

References initialise().

**6.104.2.2 ModelCheckingManager::~ModelCheckingManager ( )**

Definition at line 25 of file ModelCheckingManager.cpp.

References abstractSyntaxTrees, logicProperties, modelCheckers, and tracesPaths.

### 6.104.3 Member Function Documentation

**6.104.3.1 bool ModelCheckingManager::areUnfinishedModelCheckingTasks ( ) [private]**

Check if there exist model checkers which require extra traces.

Definition at line 270 of file ModelCheckingManager.cpp.

References modelCheckers.

Referenced by runModelCheckers(), and runModelCheckersAndRequestAdditionalTraces().

**6.104.3.2 void ModelCheckingManager::createModelCheckers ( const std::shared\_ptr< ModelCheckerFactory > & *modelCheckerFactory* ) [private]**

Create the model checker instances using the provided model checker factory.

Each model checker instance verifies one logic property

#### Parameters

|                                        |                                           |
|----------------------------------------|-------------------------------------------|
| <i>model-<br/>Checker-<br/>Factory</i> | The factory used to create model checkers |
|----------------------------------------|-------------------------------------------|

Definition at line 143 of file ModelCheckingManager.cpp.

References abstractSyntaxTrees, modelCheckers, and multiscaleArchitectureGraph.

Referenced by runModelCheckingAndOutputResults().

**6.104.3.3 void ModelCheckingManager::createNewEvaluationResults( ) [private]**

Create a new vector for storing the evaluation results for the (logic property, new trace) pairs.

The vector is created only if the shouldPrintDetailedEvaluation flag is set to true

Definition at line 200 of file ModelCheckingManager.cpp.

References evaluationResults, modelCheckers, and shouldPrintDetailedEvaluation.

Referenced by runModelCheckersForCurrentlyExistingTraces().

**6.104.3.4 void ModelCheckingManager::executeExtraEvaluationProgram( ) [private]**

Execute the extra evaluation program for generating potential new traces.

Definition at line 280 of file ModelCheckingManager.cpp.

References executeExtraEvaluationProgramAndPrintMessage(), and extraEvaluationProgramPath.

Referenced by runModelCheckersAndRequestAdditionalTraces().

**6.104.3.5 void ModelCheckingManager::executeExtraEvaluationProgramAndPrintMessage( ) [private]**

Execute the extra evaluation program for generating potential new traces.

Execute the extra evaluation program for generating potential new traces and print a message informing the user about this

Definition at line 286 of file ModelCheckingManager.cpp.

References multiscale::OperatingSystem::executeProgram(), extraEvaluationProgramPath, and multiscale::verification::ModelCheckingOutputWriter::printExecuteExtraEvaluationProgramMessage().

Referenced by executeExtraEvaluationProgram().

**6.104.3.6 SpatialTemporalTrace ModelCheckingManager::getNextSpatialTemporalTrace( ) [private]**

Get the next spatial temporal trace and store its path.

Definition at line 179 of file ModelCheckingManager.cpp.

---

References      multiscale::verification::SpatialTemporalDataReader::getNextSpatialTemporalTrace(), multiscale::verification::ModelCheckingOutputWriter::printStartTraceEvaluationMessage(), storeNewSpatialTemporalTracePath(), and traceReader.

Referenced by runModelCheckersForCurrentlyExistingTraces().

**6.104.3.7 void ModelCheckingManager::initialise ( const std::string & *logicPropertiesFilepath*, unsigned long *extraEvaluationTime*, const std::string & *multiscaleArchitectureGraphFilepath* ) [private]**

Initialise the model checking manager.

Initialise the model checking manager considering the given logic properties input file and extra evaluation time, and print the introduction message

#### Parameters

|                                               |                                                                       |
|-----------------------------------------------|-----------------------------------------------------------------------|
| <i>logic-Properties-Filepath</i>              | The path to the logic properties input file                           |
| <i>extra-Evaluation-Time</i>                  | The number of extra minutes allocated for evaluating logic properties |
| <i>multiscale-Architecture-Graph-Filepath</i> | The path to the multiscale architecture graph                         |

Definition at line 44 of file ModelCheckingManager.cpp.

References      initialiseExtraEvaluationTimeCounters(), initialiseLogicProperties(), initialiseMultiscaleArchitectureGraph(), and shouldPrintDetailedEvaluation.

Referenced by ModelCheckingManager().

**6.104.3.8 void ModelCheckingManager::initialiseExtraEvaluationTimeCounters ( unsigned long *extraEvaluationTime* ) [private]**

Initialise the extra evaluation time counters.

#### Parameters

|                              |                                                                       |
|------------------------------|-----------------------------------------------------------------------|
| <i>extra-Evaluation-Time</i> | The number of extra minutes allocated for evaluating logic properties |
|------------------------------|-----------------------------------------------------------------------|

Definition at line 54 of file ModelCheckingManager.cpp.

References      extraEvaluationElapsedTime, extraEvaluationStartTime, and extraEvaluationTime.

Referenced by initialise().

**6.104.3.9 void ModelCheckingManager::initialiseLogicProperties ( const std::string & *logicPropertiesFilepath* ) [private]**

Initialise the logic properties using the provided input file.

**Parameters**

|                                  |                                             |
|----------------------------------|---------------------------------------------|
| <i>logic-Properties-Filepath</i> | The path to the logic properties input file |
|----------------------------------|---------------------------------------------|

Definition at line 68 of file ModelCheckingManager.cpp.

References *logicProperties*, *logicPropertyReader*, and *multiscale::verification::LogicPropertyDataReader::readLogicPropertiesFromFile()*.

Referenced by *initialise()*.

**6.104.3.10 void ModelCheckingManager::initialiseMultiscaleArchitectureGraph ( const std::string & *multiscaleArchitectureGraphFilepath* ) [private]**

Initialise the multiscale architecture graph.

**Parameters**

|                                               |                                                          |
|-----------------------------------------------|----------------------------------------------------------|
| <i>multiscale-Architecture-Graph-Filepath</i> | The path to the multiscale architecture graph input file |
|-----------------------------------------------|----------------------------------------------------------|

Definition at line 60 of file ModelCheckingManager.cpp.

References *multiscaleArchitectureGraph*, and *multiscale::verification::MultiscaleArchitectureGraph::readFromFile()*.

Referenced by *initialise()*.

**6.104.3.11 bool ModelCheckingManager::isEvaluationTimeRemaining ( ) [private]**

Check if there is evaluation time remaining.

Definition at line 259 of file ModelCheckingManager.cpp.

References *extraEvaluationElapsedTime*, *extraEvaluationStartTime*, and *extraEvaluationTime*.

Referenced by *runModelCheckersAndRequestAdditionalTraces()*.

---

**6.104.3.12 bool ModelCheckingManager::isValidLogicProperty ( const std::string & *logicProperty* ) [private]**

Parse the given logic property and return true if parsing was successful and false otherwise.

Exceptions are not catched in this method

**Parameters**

|                            |                          |
|----------------------------|--------------------------|
| <i>logic-<br/>Property</i> | The given logic property |
|----------------------------|--------------------------|

Definition at line 121 of file ModelCheckingManager.cpp.

References abstractSyntaxTrees, multiscale::verification::Parser::parse(), parser, and multiscale::verification::Parser::setLogicalQuery().

Referenced by parseLogicProperty().

**6.104.3.13 void ModelCheckingManager::outputDetailedEvaluationResults ( ) [private]**

Output the logic properties detailed evaluation results.

Definition at line 325 of file ModelCheckingManager.cpp.

References evaluationResults, logicProperties, multiscale::verification::ModelCheckingOutputWriter::printDetailedEvaluationResults(), shouldPrintDetailedEvaluation, and tracesPaths.

Referenced by runModelCheckingAndOutputResults().

**6.104.3.14 void ModelCheckingManager::outputModelCheckerResults ( const std::shared\_ptr< ModelChecker > & *modelChecker*, const std::string & *logicProperty* ) [private]**

Output the model checking results for the given model checker.

**Parameters**

|                            |                                                        |
|----------------------------|--------------------------------------------------------|
| <i>model-<br/>Checker</i>  | The given model checker                                |
| <i>logic-<br/>Property</i> | The logic property verified by the given model checker |

Definition at line 316 of file ModelCheckingManager.cpp.

References multiscale::verification::ModelCheckingOutputWriter::printModelCheckingResultMessage().

Referenced by outputModelCheckersResults().

**6.104.3.15 void ModelCheckingManager::outputModelCheckersResults( )**  
[private]

Output the model checking results.

Definition at line 308 of file ModelCheckingManager.cpp.

References logicProperties, modelCheckers, and outputModelCheckerResults().

Referenced by outputModelCheckersResultsAndPrintMessage().

**6.104.3.16 void ModelCheckingManager::outputModelCheckersResultsAndPrint-  
Message( ) [private]**

Output the model checking results and print the message informing the user about this.

Definition at line 302 of file ModelCheckingManager.cpp.

References outputModelCheckersResults(), and multiscale::verification::Model-  
CheckingOutputWriter::printModelCheckingResultsIntroductionMessage().

Referenced by runModelCheckingAndOutputResults().

**6.104.3.17 void ModelCheckingManager::parseLogicProperties( ) [private]**

Parse the logic properties and create abstract syntax trees.

Parse the logic properties and create abstract syntax trees whenever a logic property  
was successfully parsed

Definition at line 89 of file ModelCheckingManager.cpp.

References logicProperties, and parseLogicPropertyAndPrintMessages().

Referenced by parseLogicPropertiesAndPrintMessage().

**6.104.3.18 void ModelCheckingManager::parseLogicPropertiesAndPrintMessage( ) [private]**

Parse the logic properties and print message informing the user about this.

Definition at line 81 of file ModelCheckingManager.cpp.

References parseLogicProperties(), multiscale::verification::ModelCheckingOutput-  
Writer::printParsingLogicPropertiesBeginMessage(), and multiscale::verification::-  
ModelCheckingOutputWriter::printParsingLogicPropertiesEndMessage().

Referenced by runModelCheckingAndOutputResults().

---

**6.104.3.19 bool ModelCheckingManager::parseLogicProperty ( const std::string & *logicProperty* ) [private]**

Parse the given logic property and return true if parsing was successful and false otherwise.

Exceptions are catched in this method

**Parameters**

|                            |                          |
|----------------------------|--------------------------|
| <i>logic-<br/>Property</i> | The given logic property |
|----------------------------|--------------------------|

Definition at line 111 of file ModelCheckingManager.cpp.

References isValidLogicProperty(), and multiscale::ExceptionHandler::printDetailedErrorMessage().

Referenced by parseLogicPropertyAndPrintMessages().

**6.104.3.20 bool ModelCheckingManager::parseLogicPropertyAndPrintMessages ( const std::string & *logicProperty* ) [private]**

Parse the logic property and inform the user if the logic property was syntactically correct.

**Parameters**

|                            |                          |
|----------------------------|--------------------------|
| <i>logic-<br/>Property</i> | The given logic property |
|----------------------------|--------------------------|

Definition at line 101 of file ModelCheckingManager.cpp.

References parseLogicProperty(), multiscale::verification::ModelCheckingOutputWriter::printParsingLogicPropertyMessage(), and printParsingMessage().

Referenced by parseLogicProperties().

**6.104.3.21 void ModelCheckingManager::printParsingMessage ( bool *isParsingsSuccessful* ) [private]**

Print a message stating if the logic property was parsed successfully.

**Parameters**

|                                   |                                               |
|-----------------------------------|-----------------------------------------------|
| <i>isParsings-<br/>Successful</i> | Flag indicating if the parsing was successful |
|-----------------------------------|-----------------------------------------------|

Definition at line 135 of file ModelCheckingManager.cpp.

References multiscale::verification::ModelCheckingOutputWriter::printFailedMessage(), and multiscale::verification::ModelCheckingOutputWriter::printSuccessMessage().

Referenced by parseLogicPropertyAndPrintMessages().

```
6.104.3.22 void ModelCheckingManager::runModelCheckerForTrace ( const
    std::size_t & modelCheckerIndex, const SpatialTemporalTrace & trace )
    [private]
```

Run the model checker for the given trace.

#### Parameters

|                                      |                                                                        |
|--------------------------------------|------------------------------------------------------------------------|
| <i>model-<br/>Checker-<br/>Index</i> | The index of the model checker inside the collection of model checkers |
| <i>trace</i>                         | The given spatial-temporal trace                                       |

Definition at line 225 of file ModelCheckingManager.cpp.

References modelCheckers, shouldPrintDetailedEvaluation, and updateEvaluationResults().

Referenced by runModelCheckersForTrace().

```
6.104.3.23 void ModelCheckingManager::runModelCheckers ( ) [private]
```

Run the model checkers and verify the logic properties.

Definition at line 160 of file ModelCheckingManager.cpp.

References areUnfinishedModelCheckingTasks(), runModelCheckersAndRequestAdditionalTraces(), and runModelCheckersForCurrentlyExistingTraces().

Referenced by runModelCheckersAndPrintMessage().

```
6.104.3.24 void ModelCheckingManager::runModelCheckersAndPrintMessage ( ) [private]
```

Run the model checkers and print a message informing the user about it.

Definition at line 154 of file ModelCheckingManager.cpp.

References multiscale::verification::ModelCheckingOutputWriter::printStartModelCheckingExecutionMessage(), and runModelCheckers().

Referenced by runModelCheckingAndOutputResults().

```
6.104.3.25 void ModelCheckingManager::runModelCheckersAndRequest-
    AdditionalTraces ( ) [private]
```

Run the model checkers and request additional traces.

Definition at line 243 of file ModelCheckingManager.cpp.

References areUnfinishedModelCheckingTasks(), executeExtraEvaluationProgram(), isEvaluationTimeRemaining(), runModelCheckersForCurrentlyExistingTraces(), updateExtraEvaluationStartTime(), updateTraceReader(), and waitBeforeRetry().

Referenced by runModelCheckers().

#### 6.104.3.26 void ModelCheckingManager::runModelCheckersForCurrentlyExistingTraces( ) [private]

Run the model checkers and verify the logic properties for the currently existing traces.

Definition at line 168 of file ModelCheckingManager.cpp.

References createNewEvaluationResults(), getNextSpatialTemporalTrace(), multiscale::verification::SpatialTemporalDataReader::hasNext(), runModelCheckersForTrace(), and traceReader.

Referenced by runModelCheckers(), and runModelCheckersAndRequestAdditionalTraces().

#### 6.104.3.27 void ModelCheckingManager::runModelCheckersForTrace ( const SpatialTemporalTrace & trace, bool & continueEvaluation ) [private]

Run the model checkers and verify the logic properties considering the given trace.

If none of the model checkers need additional traces then the continueEvaluation flag will be set to false.

##### Parameters

|                           |                                                                                                                                                                         |
|---------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>trace</i>              | The spatial temporal trace used for the logic properties evaluation                                                                                                     |
| <i>continueEvaluation</i> | The flag indicating if there is at least one logic property whose truth value was not determined yet and needs to be evaluated considering more spatial temporal traces |

Definition at line 211 of file ModelCheckingManager.cpp.

References modelCheckers, and runModelCheckerForTrace().

Referenced by runModelCheckersForCurrentlyExistingTraces().

#### 6.104.3.28 void ModelCheckingManager::runModelCheckingAndOutputResults ( const std::shared\_ptr< ModelCheckerFactory > & modelCheckerFactory ) [private]

Run the model checking tasks and output the results.

##### Parameters

|                              |                                           |
|------------------------------|-------------------------------------------|
| <i>model-Checker-Factory</i> | The factory used to create model checkers |
|------------------------------|-------------------------------------------|

Definition at line 72 of file ModelCheckingManager.cpp.

References createModelCheckers(), outputDetailedEvaluationResults(), outputModelCheckersResultsAndPrintMessage(), parseLogicPropertiesAndPrintMessage(), and runModelCheckersAndPrintMessage().

Referenced by runModelCheckingTasks().

**6.104.3.29 void ModelCheckingManager::runModelCheckingTasks ( const std::shared\_ptr< ModelCheckerFactory > & *modelCheckerFactory* )**

Run the model checking tasks.

**Parameters**

|                                        |                                           |
|----------------------------------------|-------------------------------------------|
| <i>model-<br/>Checker-<br/>Factory</i> | The factory used to create model checkers |
|----------------------------------------|-------------------------------------------|

Definition at line 40 of file ModelCheckingManager.cpp.

References runModelCheckingAndOutputResults().

**6.104.3.30 void ModelCheckingManager::setExtraEvaluationProgramPath ( const std::string & *extraEvaluationProgramPath* )**

Set the path of the program which should be executed whenever extra evaluation is required.

**Parameters**

|                                                     |                                                                                  |
|-----------------------------------------------------|----------------------------------------------------------------------------------|
| <i>extra-<br/>Evaluation-<br/>Program-<br/>Path</i> | The path to the program which will be executed when extra evaluation is required |
|-----------------------------------------------------|----------------------------------------------------------------------------------|

Definition at line 32 of file ModelCheckingManager.cpp.

References extraEvaluationProgramPath.

**6.104.3.31 void ModelCheckingManager::setShouldPrintDetailedEvaluation ( bool *shouldPrintDetailedEvaluation* )**

Set the flag indicating if the detailed evaluation should be printed.

**Parameters**

|                                                  |          |
|--------------------------------------------------|----------|
| <i>shouldPrint-<br/>Detailed-<br/>Evaluation</i> | The flag |
|--------------------------------------------------|----------|

Definition at line 36 of file ModelCheckingManager.cpp.

References shouldPrintDetailedEvaluation.

**6.104.3.32 void ModelCheckingManager::storeNewSpatialTemporalTracePath ( const std::string & tracePath ) [private]**

Store new trace path if the shouldPrintDetailedEvaluation flag is set to true.

**Parameters**

|                  |                       |
|------------------|-----------------------|
| <i>tracePath</i> | The path to the trace |
|------------------|-----------------------|

Definition at line 194 of file ModelCheckingManager.cpp.

References shouldPrintDetailedEvaluation, and tracesPaths.

Referenced by getNextSpatialTemporalTrace().

**6.104.3.33 void ModelCheckingManager::updateEvaluationResults ( const std::size\_t & modelCheckerIndex, bool evaluationResult ) [private]**

Update the evaluation results for the given model checker index and result.

**Parameters**

|                                      |                                                                        |
|--------------------------------------|------------------------------------------------------------------------|
| <i>model-<br/>Checker-<br/>Index</i> | The index of the model checker inside the collection of model checkers |
| <i>evaluation-<br/>Result</i>        | The result of evaluating the model checker for the last trace          |

Definition at line 235 of file ModelCheckingManager.cpp.

References evaluationResults.

Referenced by runModelCheckerForTrace().

**6.104.3.34 void ModelCheckingManager::updateExtraEvaluationStartTime ( ) [private]**

Set the extra evaluation start time equal to current time.

Definition at line 255 of file ModelCheckingManager.cpp.

References extraEvaluationStartTime.

Referenced by runModelCheckersAndRequestAdditionalTraces().

**6.104.3.35 void ModelCheckingManager::updateTraceReader( ) [private]**

Update trace reader.

Definition at line 298 of file ModelCheckingManager.cpp.

References multiscale::verification::SpatialTemporalDataReader::refresh(), and trace-Reader.

Referenced by runModelCheckersAndRequestAdditionalTraces().

**6.104.3.36 void ModelCheckingManager::waitBeforeRetry( ) [private]**

Wait TRACE\_INPUT\_REFRESH\_TIMEOUT minutes before updating the trace reader.

Definition at line 292 of file ModelCheckingManager.cpp.

References multiscale::verification::ModelCheckingOutputWriter::printTimeoutMessage(), and TRACE\_INPUT\_REFRESH\_TIMEOUT.

Referenced by runModelCheckersAndRequestAdditionalTraces().

#### 6.104.4 Member Data Documentation

**6.104.4.1 std::vector<AbstractSyntaxTree> multiscale::verification-  
::ModelCheckingManager::abstractSyntaxTrees  
[private]**

The collection of abstract syntax tree obtained after parsing the logic properties

Definition at line 36 of file ModelCheckingManager.hpp.

Referenced by createModelCheckers(), isValidLogicProperty(), and ~ModelCheckingManager().

**6.104.4.2 std::vector<std::vector<bool>> multiscale::verification-  
::ModelCheckingManager::evaluationResults  
[private]**

The two-dimensional array storing the evaluation result for each (logic property, trace) pair. A pair of boolean values (isEvaluated, evaluationResult) is associated to each (logic property, trace) pair

Definition at line 50 of file ModelCheckingManager.hpp.

Referenced by createNewEvaluationResults(), outputDetailedEvaluationResults(), and updateEvaluationResults().

**6.104.4.3 double multiscale::verification::ModelCheckingManager::extra-  
EvaluationElapsedTime [private]**

The elapsed time for the extra evaluation process expressed in seconds

Definition at line 61 of file ModelCheckingManager.hpp.

Referenced by initialiseExtraEvaluationTimeCounters(), and isEvaluationTimeRemaining().

#### **6.104.4.4 std::string multiscale::verification::ModelCheckingManager::extraEvaluationProgramPath [private]**

The path to the program which should be executed when extra evaluation is required

Definition at line 67 of file ModelCheckingManager.hpp.

Referenced by executeExtraEvaluationProgram(), executeExtraEvaluationProgramAndPrintMessage(), and setExtraEvaluationProgramPath().

#### **6.104.4.5 std::chrono::time\_point<std::chrono::system\_clock> multiscale::verification::ModelCheckingManager::extraEvaluationStartTime [private]**

The start time for the current evaluation process

Definition at line 59 of file ModelCheckingManager.hpp.

Referenced by initialiseExtraEvaluationTimeCounters(), isEvaluationTimeRemaining(), and updateExtraEvaluationStartTime().

#### **6.104.4.6 unsigned long multiscale::verification::ModelCheckingManager::extraEvaluationTime [private]**

The number of minutes for which the program waits for new traces to be added to the trace folder

Definition at line 64 of file ModelCheckingManager.hpp.

Referenced by initialiseExtraEvaluationTimeCounters(), and isEvaluationTimeRemaining().

#### **6.104.4.7 std::vector<std::string> multiscale::verification::ModelCheckingManager::logicProperties [private]**

The collection of logic properties

Definition at line 34 of file ModelCheckingManager.hpp.

Referenced by initialiseLogicProperties(), outputDetailedEvaluationResults(), outputModelCheckersResults(), parseLogicProperties(), and ~ModelCheckingManager().

**6.104.4.8 LogicPropertyDataReader multiscale::verification::ModelCheckingManager::logicPropertyReader [private]**

The logic property reader

Definition at line 42 of file ModelCheckingManager.hpp.

Referenced by initialiseLogicProperties().

**6.104.4.9 std::vector<std::shared\_ptr<ModelChecker>> multiscale::verification::ModelCheckingManager::modelCheckers [private]**

The collection of model checkers

Definition at line 56 of file ModelCheckingManager.hpp.

Referenced by areUnfinishedModelCheckingTasks(), createModelCheckers(), createNewEvaluationResults(), outputModelCheckersResults(), runModelCheckerForTrace(), runModelCheckersForTrace(), and ~ModelCheckingManager().

**6.104.4.10 MultiscaleArchitectureGraph multiscale::verification::ModelCheckingManager::multiscaleArchitectureGraph [private]**

The multiscale architecture graph

Definition at line 47 of file ModelCheckingManager.hpp.

Referenced by createModelCheckers(), and initialiseMultiscaleArchitectureGraph().

**6.104.4.11 Parser multiscale::verification::ModelCheckingManager::parser [private]**

The parser used to verify if logical properties are syntactically correct

Definition at line 30 of file ModelCheckingManager.hpp.

Referenced by isValidLogicProperty().

**6.104.4.12 const std::string ModelCheckingManager::PARSE\_R\_EMPTY\_LOGIC\_PROPERTY = "" [static, private]**

An empty logic property

Definition at line 285 of file ModelCheckingManager.hpp.

---

**6.104.4.13 bool multiscale::verification::ModelCheckingManager::shouldPrintDetailedEvaluation [private]**

Flag indicating if detailed evaluation results should be printed

Definition at line 71 of file ModelCheckingManager.hpp.

Referenced by `createNewEvaluationResults()`, `initialise()`, `outputDetailedEvaluationResults()`, `runModelCheckerForTrace()`, `setShouldPrintDetailedEvaluation()`, and `storeNewSpatialTemporalTracePath()`.

**6.104.4.14 const unsigned long ModelCheckingManager::TRACE\_INPUT\_REFRESH\_TIMEOUT = 30 [static, private]**

The number of seconds for which the manager waits before updating the trace reader

Definition at line 281 of file ModelCheckingManager.hpp.

Referenced by `waitBeforeRetry()`.

**6.104.4.15 SpatialTemporalDataReader multiscale::verification::ModelCheckingManager::traceReader [private]**

The behaviour/trace reader

Definition at line 44 of file ModelCheckingManager.hpp.

Referenced by `getNextSpatialTemporalTrace()`, `runModelCheckersForCurrentlyExistingTraces()`, and `updateTraceReader()`.

**6.104.4.16 std::vector<std::string> multiscale::verification::ModelCheckingManager::tracesPaths [private]**

The collection of traces paths

Definition at line 39 of file ModelCheckingManager.hpp.

Referenced by `outputDetailedEvaluationResults()`, `storeNewSpatialTemporalTracePath()`, and `~ModelCheckingManager()`.

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

- `ModelCheckingManager.hpp`
- `ModelCheckingManager.cpp`

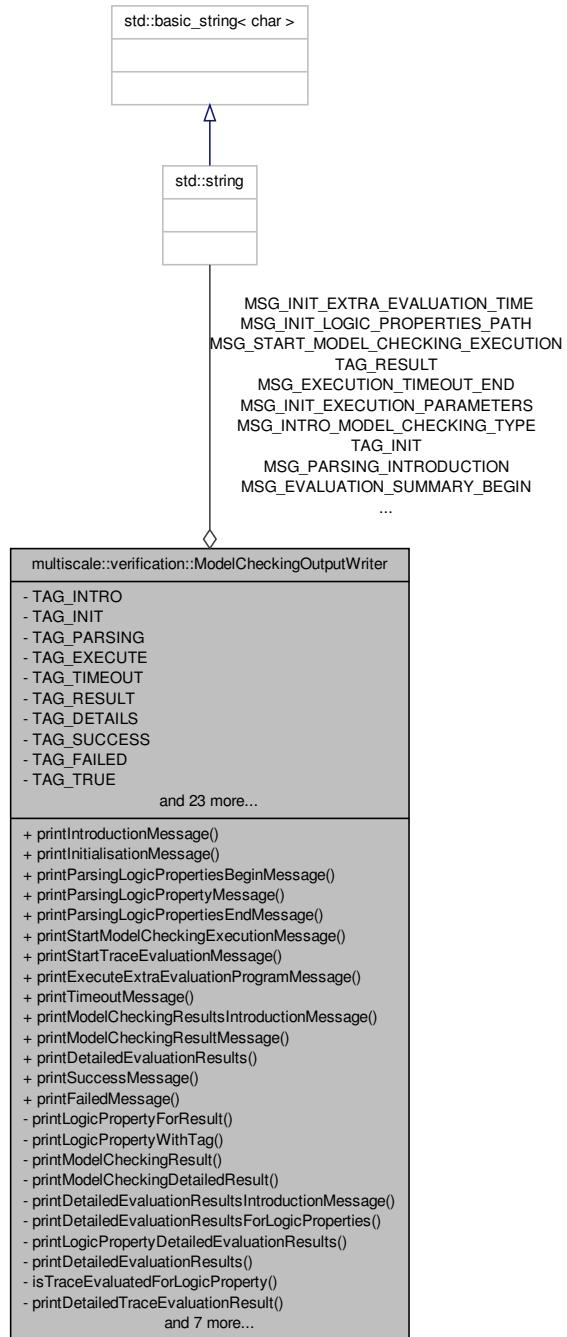
## 6.105 multiscale::verification::ModelCheckingOutputWriter Class - Reference

Class used to output the model checkers progress.

## **6.105 multiscale::verification::ModelCheckingOutputWriter Class Reference 587**

```
#include <ModelCheckingOutputWriter.hpp>
```

Collaboration diagram for multiscale::verification::ModelCheckingOutputWriter:



### Static Public Member Functions

- static void [printIntroductionMessage](#) (const std::string &modelCheckerType, const std::string &modelCheckerParameters)  
*Print the model checker introduction message considering the given model checker details.*
- static void [printInitialisationMessage](#) (const std::string &logicProperty, const std::string &tracesFolderPath, unsigned long extraEvaluationTime)  
*Print the model checker initialisation message.*
- static void [printParsingLogicPropertiesBeginMessage](#) ()  
*Print an introduction message informing the user that the logic properties will be parsed.*
- static void [printParsingLogicPropertyMessage](#) (const std::string &logicProperty)  
*Print a message informing the user which logic property will be parsed.*
- static void [printParsingLogicPropertiesEndMessage](#) ()  
*Print a closing message after the logic properties were parsed.*
- static void [printStartModelCheckingExecutionMessage](#) ()  
*Print a message informing the user that the model checking execution has started.*
- static void [printStartTraceEvaluationMessage](#) (const std::string &tracePath)  
*Print a message informing the user which trace will be evaluated next by the model checkers.*
- static void [printExecuteExtraEvaluationProgramMessage](#) (const std::string &programPath)  
*Print a message informing the user that the extra evaluation program located at the given path will be executed.*
- static void [printTimeoutMessage](#) (unsigned long timeOut)  
*Print a message informing the user that the model checking execution is suspended for timeOut seconds.*
- static void [printModelCheckingResultsIntroductionMessage](#) ()  
*Print an introduction message informing the user that the model checking results will be displayed.*
- static void [printModelCheckingResultMessage](#) (bool doesPropertyHold, const std::string &detailedResult, const std::string &logicProperty)  
*Print a message with the results of checking if the given property holds.*
- static void [printDetailedEvaluationResults](#) (const std::vector< std::string > &logicProperties, const std::vector< std::string > &tracesPaths, const std::vector< std::vector< bool > > &evaluationResults)  
*Print for each logic property the traces for which the evaluation result was true/false.*
- static void [printSuccessMessage](#) ()  
*Print a success message.*
- static void [printFailedMessage](#) ()  
*Print a fail message.*

## Static Private Member Functions

- static void `printLogicPropertyForResult` (const std::string &logicProperty)
 

*Print the given logic property in the context of a result message.*
- static void `printLogicPropertyWithTag` (const std::string &logicProperty, const std::string &tag)
 

*Print the given logic property in the context of the provided tag.*
- static void `printModelCheckingResult` (bool doesPropertyHold)
 

*Print if the logic property verified by the model checker holds in the context of a result message.*
- static void `printModelCheckingDetailedResult` (bool doesPropertyHold, const std::string &detailedResult)
 

*Print the detailed result of the model checking procedure.*
- static void `printDetailedEvaluationResultsIntroductionMessage` ()
 

*Print an introduction message informing the user that the detailed evaluation results will be printed.*
- static void `printDetailedEvaluationResultsForLogicProperties` (const std::vector<std::string> &logicProperties, const std::vector<std::string> &tracesPaths, const std::vector<std::vector<bool>> &evaluationResults)
 

*Print the detailed evaluation results for the given logic properties and traces.*
- static void `printLogicPropertyDetailedEvaluationResults` (const std::size\_t &logicPropertyIndex, const std::vector<std::string> &tracesPaths, const std::vector<std::vector<bool>> &evaluationResults)
 

*Print the detailed evaluation results for the given logic property.*
- static void `printDetailedEvaluationResults` (const std::size\_t &logicPropertyIndex, const std::vector<std::string> &tracesPaths, const std::vector<std::vector<bool>> &evaluationResults)
 

*Print the detailed evaluation results for the given logic property.*
- static bool `isTraceEvaluatedForLogicProperty` (const std::size\_t &logicPropertyIndex, const std::size\_t &tracePathIndex, const std::vector<std::vector<bool>> &evaluationResults)
 

*Check if the trace was evaluated for the given logic property.*
- static void `printDetailedTraceEvaluationResult` (const std::size\_t &logicPropertyIndex, const std::string &tracePath, const std::size\_t &tracePathIndex, const std::vector<std::vector<bool>> &evaluationResults)
 

*Print the detailed evaluation result for the given logic property and trace.*
- static void `printTraceEvaluationResult` (const std::string &tracePath, bool evaluationResult)
 

*Print the trace path with the associated evaluation result.*
- static void `printEvaluationResultsSummary` (const std::size\_t &logicPropertyIndex, const std::vector<std::string> &tracesPaths, const std::vector<std::vector<bool>> &evaluationResults)
 

*Print the summary of the evaluation results for the given logic property.*
- static void `updateSummaryEvaluationResults` (const std::size\_t &logicPropertyIndex, const std::size\_t &tracePathIndex, const std::vector<std::vector<bool>> &evaluationResults, size\_t &nrOfEvaluatedTraces, size\_t &nrOfTracesEvaluatedTrue)
 

*Update the summary of the evaluation results for the given logic property.*

*Update the summary evaluation results considering the logic property, trace and evaluation results.*

- static bool `isTraceEvaluatedTrueForLogicProperty` (const std::size\_t &logicPropertyIndex, const std::size\_t &tracePathIndex, const std::vector< std::vector< bool >> &evaluationResults)

*Check if the trace was evaluated to true for the given logic property.*

- static void `printEvaluationResultsSummary` (std::size\_t nrOfTraces, std::size\_t nrOfCorrectTraces)

*Print the summary of the evaluation results for the given logic property.*

- static void `printTruthValueDependentMessage` (const std::string &message, const std::string &tag, bool truthValue)

*Print a message with the given tag and colour depending on the truth value.*

- static void `printResultTag` ()

*Print a line containing a result tag and no content.*

- static void `printSeparatorTag` ()

*Print a line containing a separator tag.*

## Static Private Attributes

- static const std::string `TAG_INTRO` = "[ INTRO ]"
- static const std::string `TAG_INIT` = "[ INIT ]"
- static const std::string `TAG_PARSING` = "[ PARSING ]"
- static const std::string `TAG_EXECUTE` = "[ EXECUTE ]"
- static const std::string `TAG_TIMEOUT` = "[ TIMEOUT ]"
- static const std::string `TAG_RESULT` = "[ RESULT ]"
- static const std::string `TAG_DETAILS` = "[ DETAILS ]"
- static const std::string `TAG_SUCCESS` = "[ SUCCESS ]"
- static const std::string `TAG_FAILED` = "[ FAILED ]"
- static const std::string `TAG_TRUE` = "[ TRUE ]"
- static const std::string `TAG_FALSE` = "[ FALSE ]"
- static const std::string `TAG_SEPARATOR` = "=====
- static const std::string `MSG_INTRO_NAME` = "Mule 1.0.513 (Multidimensional multiscale model checker)"
- static const std::string `MSG_INTRO_COPYRIGHT` = "Copyright Ovidiu Pârvu 2014"
- static const std::string `MSG_INTRO_MODEL_CHECKING_TYPE` = "Model checker type: "
- static const std::string `MSG_INTRO_MODEL_CHECKING_PARAMETERS` = "- Parameters: "
- static const std::string `MSG_INTRO_CONTACT` = "For more details, recommendations or suggestions feel free to contact me at <ovidiu.parvu[AT]gmail.com>."
- static const std::string `MSG_INIT_EXECUTION_PARAMETERS` = "Multidimensional multiscale model checking input parameters"
- static const std::string `MSG_INIT_LOGIC_PROPERTIES_PATH` = "Logic properties input file: "

- static const std::string `MSG_INIT_TRACES_FOLDER_PATH` = "Spatio-temporal traces input folder: "
- static const std::string `MSG_INIT_EXTRA_EVALUATION_TIME` = "Extra evaluation time (minutes): "
- static const std::string `MSG_PARSING_INTRODUCTION` = "I am starting to parse logic properties..."
- static const std::string `MSG_START_MODEL_CHECKING_EXECUTION` = "I am starting the execution of the model checkers..."
- static const std::string `MSG_START_TRACE_EVALUATION` = "Evaluating the spatio-temporal trace: "
- static const std::string `MSG_START_EXTRA_EVALUATION_PROGRAM_EXECUTION` = "I am starting the execution of the extra evaluation program located at the following path: "
- static const std::string `MSG_EXECUTION_TIMEOUT_BEGIN` = "The model checker execution was suspended for "
- static const std::string `MSG_EXECUTION_TIMEOUT_END` = " seconds during which new traces can be provided in the traces input folder."
- static const std::string `MSG_RESULTS_INTRODUCTION` = "I have finished evaluating the logic properties and will display the results..."
- static const std::string `MSG_EVALUATION_RESULTS_INTRODUCTION` = "I will display for each logic property which traces evaluated to `TRUE` and which evaluated to `FALSE`..."
- static const std::string `MSG_EVALUATION_SUMMARY_BEGIN` = "/"
- static const std::string `MSG_EVALUATION_SUMMARY_END` = " spatio-temporal traces evaluated to `TRUE`"
- static const std::string `MSG_LOGIC_PROPERTY HOLDS` = "The logic property holds: "
- static const std::string `MSG_LOGIC_PROPERTY HOLDS TRUE` = "`TRUE`"
- static const std::string `MSG_LOGIC_PROPERTY HOLDS FALSE` = "`FALSE`"

### 6.105.1 Detailed Description

Class used to output the model checkers progress.

Definition at line 12 of file ModelCheckingOutputWriter.hpp.

### 6.105.2 Member Function Documentation

```
6.105.2.1 bool ModelCheckingOutputWriter::isTraceEvaluatedForLogicProperty
( const std::size_t & logicPropertyIndex, const std::size_t & tracePathIndex,
  const std::vector< std::vector< bool >> & evaluationResults ) [static,
  private]
```

Check if the trace was evaluated for the given logic property.

## **6.105 multiscale::verification::ModelCheckingOutputWriter Class Reference 593**

### Parameters

|                             |                                                                                                                                                                                                                                                                                                         |
|-----------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>logic-Property-Index</i> | The index of the logic property in the collection of logic properties                                                                                                                                                                                                                                   |
| <i>tracePath-Index</i>      | The index of the trace path in the collection of trace paths                                                                                                                                                                                                                                            |
| <i>evaluation-Results</i>   | The evaluation results (i.e. a two-dimensional array of size $ \text{logicProperties}  \times  2 *  \text{traces}  $ where the first boolean value associated to a (logicProperty, trace) pair states if the logic property was evaluated for that trace and the second one stores the evaluation value |

Definition at line 211 of file ModelCheckingOutputWriter.cpp.

Referenced by `isTraceEvaluatedTrueForLogicProperty()`, `printDetailedEvaluationResults()`, and `updateSummaryEvaluationResults()`.

```
6.105.2.2 bool ModelCheckingOutputWriter::isTraceEvaluatedTrueForLogic-  
Property ( const std::size_t & logicPropertyIndex, const std::size_t & tracePathIndex,  
const std::vector< std::vector< bool >> & evaluationResults ) [static,  
private]
```

Check if the trace was evaluated to true for the given logic property.

### Parameters

|                             |                                                                                                                                                                                                                                                                                                         |
|-----------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>logic-Property-Index</i> | The index of the logic property in the collection of logic properties                                                                                                                                                                                                                                   |
| <i>tracePath-Index</i>      | The index of the trace path in the collection of trace paths                                                                                                                                                                                                                                            |
| <i>evaluation-Results</i>   | The evaluation results (i.e. a two-dimensional array of size $ \text{logicProperties}  \times  2 *  \text{traces}  $ where the first boolean value associated to a (logicProperty, trace) pair states if the logic property was evaluated for that trace and the second one stores the evaluation value |

Definition at line 250 of file ModelCheckingOutputWriter.cpp.

References `isTraceEvaluatedForLogicProperty()`.

Referenced by `updateSummaryEvaluationResults()`.

```
6.105.2.3 void ModelCheckingOutputWriter::printDetailedEvaluationResults (  
const std::vector< std::string > & logicProperties, const std::vector< std::string >  
& tracesPaths, const std::vector< std::vector< bool >> & evaluationResults )  
[static]
```

Print for each logic property the traces for which the evaluation result was true/false.

**Parameters**

|                           |                                                                                                                                                                                                                                                                                                                          |
|---------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>logic-Properties</i>   | The collection of logic properties                                                                                                                                                                                                                                                                                       |
| <i>tracesPaths</i>        | The collection of trace paths                                                                                                                                                                                                                                                                                            |
| <i>evaluation-Results</i> | The evaluation results (i.e. a two-dimensional array of size $ \text{logic-Properties}  \times  2 *  \text{traces}  $ where the first boolean value associated to a $(\text{logicProperty}, \text{trace})$ pair states if the logic property was evaluated for that trace and the second one stores the evaluation value |

Definition at line 99 of file ModelCheckingOutputWriter.cpp.

References `printDetailedEvaluationResultsForLogicProperties()`, and `printDetailedEvaluationResultsIntroductionMessage()`.

Referenced by `multiscale::verification::ModelCheckingManager::outputDetailedEvaluationResults()`, and `printLogicPropertyDetailedEvaluationResults()`.

```
6.105.2.4 void ModelCheckingOutputWriter::printDetailedEvaluationResults
( const std::size_t & logicPropertyIndex, const std::vector< std::string > &
tracesPaths, const std::vector< std::vector< bool >> & evaluationResults )
[static, private]
```

Print the detailed evaluation results for the given logic property.

**Parameters**

|                             |                                                                                                                                                                                                                                                                                                                          |
|-----------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>logic-Property-Index</i> | The index of the logic property in the collection of logic properties                                                                                                                                                                                                                                                    |
| <i>tracesPaths</i>          | The collection of trace paths                                                                                                                                                                                                                                                                                            |
| <i>evaluation-Results</i>   | The evaluation results (i.e. a two-dimensional array of size $ \text{logic-Properties}  \times  2 *  \text{traces}  $ where the first boolean value associated to a $(\text{logicProperty}, \text{trace})$ pair states if the logic property was evaluated for that trace and the second one stores the evaluation value |

Definition at line 170 of file ModelCheckingOutputWriter.cpp.

References `isTraceEvaluatedForLogicProperty()`, and `printDetailedTraceEvaluationResult()`.

```
6.105.2.5 void ModelCheckingOutputWriter::printDetailedEvaluationResultsFor-
LogicProperties ( const std::vector< std::string > & logicProperties, const
std::vector< std::string > & tracesPaths, const std::vector< std::vector< bool >> &
evaluationResults ) [static, private]
```

Print the detailed evaluation results for the given logic properties and traces.

## **6.105 multiscale::verification::ModelCheckingOutputWriter Class Reference 595**

### Parameters

|                           |                                                                                                                                                                                                                                                                                                          |
|---------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>logic-Properties</i>   | The collection of logic properties                                                                                                                                                                                                                                                                       |
| <i>tracesPaths</i>        | The collection of trace paths                                                                                                                                                                                                                                                                            |
| <i>evaluation-Results</i> | The evaluation results (i.e. a two-dimensional array of size $ \text{logic-Properties}  \times  2 *  \text{traces}  $ where the first boolean value associated to a (logicProperty, trace) pair states if the logic property was evaluated for that trace and the second one stores the evaluation value |

Definition at line 149 of file ModelCheckingOutputWriter.cpp.

References multiscale::ConsolePrinter::printColouredMessage(), printLogicProperty-DetailedEvaluationResults(), printLogicPropertyWithTag(), printSeparatorTag(), and TAG\_DETAILS.

Referenced by printDetailedEvaluationResults().

**6.105.2.6 void ModelCheckingOutputWriter::printDetailedEvaluationResultsIntroductionMessage ( ) [static, private]**

Print an introduction message informing the user that the detailed evaluation results will be printed.

Definition at line 140 of file ModelCheckingOutputWriter.cpp.

References MSG\_EVALUATION\_RESULTS\_INTRODUCTION, multiscale::ConsolePrinter::printColouredMessage(), multiscale::ConsolePrinter::printEmptyLine(), multiscale::ConsolePrinter::printMessageWithColouredTag(), printSeparatorTag(), and TAG\_DETAILS.

Referenced by printDetailedEvaluationResults().

**6.105.2.7 void ModelCheckingOutputWriter::printDetailedTraceEvaluationResult ( const std::size\_t & *logicPropertyIndex*, const std::string & *tracePath*, const std::size\_t & *tracePathIndex*, const std::vector< std::vector< bool >> & *evaluationResults* ) [static, private]**

Print the detailed evaluation result for the given logic property and trace.

### Parameters

|                             |                                                                                                                                                                                                                                                                                                          |
|-----------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>logic-Property-Index</i> | The index of the logic property in the collection of logic properties                                                                                                                                                                                                                                    |
| <i>tracePath</i>            | The path to the spatial temporal trace                                                                                                                                                                                                                                                                   |
| <i>tracePath-Index</i>      | The index of the trace path in the collection of trace paths                                                                                                                                                                                                                                             |
| <i>evaluation-Results</i>   | The evaluation results (i.e. a two-dimensional array of size $ \text{logic-Properties}  \times  2 *  \text{traces}  $ where the first boolean value associated to a (logicProperty, trace) pair states if the logic property was evaluated for that trace and the second one stores the evaluation value |

Definition at line 231 of file ModelCheckingOutputWriter.cpp.

References printTraceEvaluationResult().

Referenced by printDetailedEvaluationResults().

```
6.105.2.8 void ModelCheckingOutputWriter::printEvaluationResultsSummary
( const std::size_t & logicPropertyIndex, const std::vector< std::string > &
tracesPaths, const std::vector< std::vector< bool >> & evaluationResults )
[static, private]
```

Print the summary of the evaluation results for the given logic property.

Print a message informing the user how many traces out of the total number of traces evaluated to true for the given logic property.

#### Parameters

|                                       |                                                                                                                                                                                                                                                                                                         |
|---------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>logic-<br/>Property-<br/>Index</i> | The index of the logic property in the collection of logic properties                                                                                                                                                                                                                                   |
| <i>tracesPaths</i>                    | The collection of trace paths                                                                                                                                                                                                                                                                           |
| <i>evaluation-<br/>Results</i>        | The evaluation results (i.e. a two-dimensional array of size $ \text{logicProperties}  \times  2 *  \text{traces}  $ where the first boolean value associated to a (logicProperty, trace) pair states if the logic property was evaluated for that trace and the second one stores the evaluation value |

Definition at line 182 of file ModelCheckingOutputWriter.cpp.

References updateSummaryEvaluationResults().

Referenced by printLogicPropertyDetailedEvaluationResults().

```
6.105.2.9 void ModelCheckingOutputWriter::printEvaluationResultsSummary (
std::size_t nrOfTraces, std::size_t nrOfCorrectTraces ) [static, private]
```

Print the summary of the evaluation results for the given logic property.

Print a message informing the user how many traces out of the total number of traces evaluated to true for the given logic property.

#### Parameters

|                                |                                                                                     |
|--------------------------------|-------------------------------------------------------------------------------------|
| <i>nrOfTraces</i>              | The total number of traces                                                          |
| <i>nrOfCorrect-<br/>Traces</i> | The number of traces out of the total number of traces which were evaluated to true |

Definition at line 217 of file ModelCheckingOutputWriter.cpp.

References MSG\_EVALUATION\_SUMMARY\_BEGIN, MSG\_EVALUATION\_SUMMARY\_END, multiscale::ConsolePrinter::printColouredMessage(), multiscale::ConsolePrinter::printMessageWithColouredTag(), TAG\_DETAILS, and multiscale::String-

Manipulator::toString().

**6.105.2.10 void ModelCheckingOutputWriter::printExecuteExtra-EvaluationProgramMessage ( const std::string & *programPath* ) [static]**

Print a message informing the user that the extra evaluation program located at the given path will be executed.

**Parameters**

|                    |                                          |
|--------------------|------------------------------------------|
| <i>programPath</i> | The path to the extra evaluation program |
|--------------------|------------------------------------------|

Definition at line 67 of file ModelCheckingOutputWriter.cpp.

References MSG\_START\_EXTRA\_EVALUATION\_PROGRAM\_EXECUTION, multiscale-::ConsolePrinter::printMessageWithColouredTag(), and TAG\_EXECUTE.

Referenced by multiscale::verification::ModelCheckingManager::executeExtra-EvaluationProgramAndPrintMessage().

**6.105.2.11 void ModelCheckingOutputWriter::printFailedMessage ( ) [static]**

Print a fail message.

Definition at line 111 of file ModelCheckingOutputWriter.cpp.

References multiscale::ConsolePrinter::printColouredMessage(), printSeparatorTag(), and TAG\_FAILED.

Referenced by multiscale::verification::ModelCheckingManager::printParsingMessage().

**6.105.2.12 void ModelCheckingOutputWriter::printInitialisationMessage ( const std::string & *logicProperty*, const std::string & *tracesFolderPath*, unsigned long *extraEvaluationTime* ) [static]**

Print the model checker initialisation message.

**Parameters**

|                              |                                                                                                         |
|------------------------------|---------------------------------------------------------------------------------------------------------|
| <i>logic-Property</i>        | The path to the input file containing logic properties                                                  |
| <i>traces-FolderPath</i>     | The path to the folder containing the traces                                                            |
| <i>extra-Evaluation-Time</i> | The number of extra minutes which the application will wait for new traces to be provided and evaluated |

Definition at line 24 of file ModelCheckingOutputWriter.cpp.

---

References MSG\_INIT\_EXECUTION\_PARAMETERS, MSG\_INIT\_EXTRA\_EVALUATION\_TIME, MSG\_INIT\_LOGIC\_PROPERTIES\_PATH, MSG\_INIT\_TRACES\_FOLDER\_PATH, multiscale::ConsolePrinter::printColouredMessage(), multiscale::ConsolePrinter::printEmptyLine(), multiscale::ConsolePrinter::printMessageWithColouredTag(), TAG\_INIT, and multiscale::StringManipulator::toString().

Referenced by multiscale::verification::CommandLineModelChecking::printModelCheckingInitialisationMessage().

**6.105.2.13 void ModelCheckingOutputWriter::printIntroductionMessage ( const std::string & *modelCheckerType*, const std::string & *modelCheckerParameters* ) [static]**

Print the model checker introduction message considering the given model checker details.

#### Parameters

|                                           |                               |
|-------------------------------------------|-------------------------------|
| <i>model-<br/>Checker-<br/>Type</i>       | The type of the model checker |
| <i>model-<br/>Checker-<br/>Parameters</i> | The model checking parameters |

Definition at line 8 of file ModelCheckingOutputWriter.cpp.

References MSG\_INTRO\_CONTACT, MSG\_INTRO\_COPYRIGHT, MSG\_INTRO\_MODEL\_CHECKING\_PARAMETERS, MSG\_INTRO\_MODEL\_CHECKING\_TYPE, MSG\_INTRO\_NAME, multiscale::ConsolePrinter::printColouredMessage(), multiscale::ConsolePrinter::printEmptyLine(), multiscale::ConsolePrinter::printMessageWithColouredTag(), and TAG\_INTRO.

Referenced by multiscale::verification::CommandLineModelChecking::printModelCheckingInitialisationMessage().

**6.105.2.14 void ModelCheckingOutputWriter::printLogicPropertyDetailedEvaluationResults ( const std::size\_t & *logicPropertyIndex*, const std::vector< std::string > & *tracesPaths*, const std::vector< std::vector< bool >> & *evaluationResults* ) [static, private]**

Print the detailed evaluation results for the given logic property.

#### Parameters

|                                       |                                                                       |
|---------------------------------------|-----------------------------------------------------------------------|
| <i>logic-<br/>Property-<br/>Index</i> | The index of the logic property in the collection of logic properties |
| <i>tracesPaths</i>                    | The collection of trace paths                                         |

## **6.105 multiscale::verification::ModelCheckingOutputWriter Class Reference 599**

|                                |                                                                                                                                                                                                                                                                                                         |
|--------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>evaluation-<br/>Results</i> | The evaluation results (i.e. a two-dimensional array of size $ \text{logicProperties}  \times  2 *  \text{traces}  $ where the first boolean value associated to a (logicProperty, trace) pair states if the logic property was evaluated for that trace and the second one stores the evaluation value |
|--------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

Definition at line 163 of file ModelCheckingOutputWriter.cpp.

References printDetailedEvaluationResults(), and printEvaluationResultsSummary().

Referenced by printDetailedEvaluationResultsForLogicProperties().

**6.105.2.15 void ModelCheckingOutputWriter::printLogicPropertyForResult ( const std::string & *logicProperty* ) [static, private]**

Print the given logic property in the context of a result message.

### Parameters

|                            |                    |
|----------------------------|--------------------|
| <i>logic-<br/>Property</i> | The logic property |
|----------------------------|--------------------|

Definition at line 116 of file ModelCheckingOutputWriter.cpp.

References printLogicPropertyWithTag(), and TAG\_RESULT.

Referenced by printModelCheckingResultMessage().

**6.105.2.16 void ModelCheckingOutputWriter::printLogicPropertyWithTag ( const std::string & *logicProperty*, const std::string & *tag* ) [static, private]**

Print the given logic property in the context of the provided tag.

### Parameters

|                            |                          |
|----------------------------|--------------------------|
| <i>logic-<br/>Property</i> | The given logic property |
| <i>tag</i>                 | The given tag            |

Definition at line 120 of file ModelCheckingOutputWriter.cpp.

References multiscale::ConsolePrinter::printMessageWithColouredTag(), and multiscale::StringManipulator::trimRight().

Referenced by printDetailedEvaluationResultsForLogicProperties(), and printLogicPropertyForResult().

---

**6.105.2.17 void ModelCheckingOutputWriter::printModelCheckingDetailedResult ( bool *doesPropertyHold*, const std::string & *detailedResult* ) [static, private]**

Print the detailed result of the model checking procedure.

Definition at line 135 of file ModelCheckingOutputWriter.cpp.

References printTruthValueDependentMessage(), and TAG\_RESULT.

Referenced by printModelCheckingResultMessage().

**6.105.2.18 void ModelCheckingOutputWriter::printModelCheckingResult ( bool *doesPropertyHold* ) [static, private]**

Print if the logic property verified by the model checker holds in the context of a result message.

#### Parameters

|                                     |                                             |
|-------------------------------------|---------------------------------------------|
| <i>does-<br/>Property-<br/>Hold</i> | Flag indicating if the logic property holds |
|-------------------------------------|---------------------------------------------|

Definition at line 126 of file ModelCheckingOutputWriter.cpp.

References MSG\_LOGIC\_PROPERTY HOLDS, MSG\_LOGIC\_PROPERTY HOLDS FALSE, MSG\_LOGIC\_PROPERTY HOLDS TRUE, printTruthValueDependentMessage(), and TAG\_RESULT.

Referenced by printModelCheckingResultMessage().

**6.105.2.19 void ModelCheckingOutputWriter::printModelCheckingResultMessage ( bool *doesPropertyHold*, const std::string & *detailedResult*, const std::string & *logicProperty* ) [static]**

Print a message with the results of checking if the given property holds.

#### Parameters

|                                     |                                                                                                                |
|-------------------------------------|----------------------------------------------------------------------------------------------------------------|
| <i>does-<br/>Property-<br/>Hold</i> | The flag indicating if the logic property holds (with a given probability and/or confidence)                   |
| <i>detailed-<br/>Result</i>         | The detailed result report indicating if the logic property holds (with a given probability and/or confidence) |
| <i>logic-<br/>Property</i>          | The logic property to be verified                                                                              |

Definition at line 88 of file ModelCheckingOutputWriter.cpp.

References printLogicPropertyForResult(), printModelCheckingDetailedResult(), printModelCheckingResult(), printResultTag(), and printSeparatorTag().

## **6.105 multiscale::verification::ModelCheckingOutputWriter Class Reference 601**

---

Referenced by multiscale::verification::ModelCheckingManager::outputModelCheckerResults().

### **6.105.2.20 void ModelCheckingOutputWriter::printModelCheckingResults-IntroductionMessage( ) [static]**

Print an introduction message informing the user that the model checking results will be displayed.

Definition at line 79 of file ModelCheckingOutputWriter.cpp.

References MSG\_RESULTS\_INTRODUCTION, multiscale::ConsolePrinter::printColouredMessage(), multiscale::ConsolePrinter::printEmptyLine(), multiscale::ConsolePrinter::printMessageWithColouredTag(), printSeparatorTag(), and TAG\_RESULT.

Referenced by multiscale::verification::ModelCheckingManager::outputModelCheckersResultsAndPrintMessage().

### **6.105.2.21 void ModelCheckingOutputWriter::printParsingLogicPropertiesBegin-Message( ) [static]**

Print an introduction message informing the user that the logic properties will be parsed.

Definition at line 41 of file ModelCheckingOutputWriter.cpp.

References MSG\_PARSING\_INTRODUCTION, multiscale::ConsolePrinter::printColouredMessage(), multiscale::ConsolePrinter::printMessageWithColouredTag(), printSeparatorTag(), and TAG\_PARSING.

Referenced by multiscale::verification::ModelCheckingManager::parseLogicPropertiesAndPrintMessage().

### **6.105.2.22 void ModelCheckingOutputWriter::printParsingLogicPropertiesEnd-Message( ) [static]**

Print a closing message after the logic properties were parsed.

Definition at line 52 of file ModelCheckingOutputWriter.cpp.

References multiscale::ConsolePrinter::printEmptyLine().

Referenced by multiscale::verification::ModelCheckingManager::parseLogicPropertiesAndPrintMessage().

### **6.105.2.23 void ModelCheckingOutputWriter::printParsingLogicPropertyMessage ( const std::string & *logicProperty* ) [static]**

Print a message informing the user which logic property will be parsed.

**Parameters**

|                            |                          |
|----------------------------|--------------------------|
| <i>logic-<br/>Property</i> | The given logic property |
|----------------------------|--------------------------|

Definition at line 48 of file ModelCheckingOutputWriter.cpp.

References multiscale::ConsolePrinter::printMessageWithColouredTag(), TAG\_PARSI-NG, and multiscale::StringManipulator::trimRight().

Referenced by multiscale::verification::ModelCheckingManager::parseLogicProperty-AndPrintMessages().

**6.105.2.24 void ModelCheckingOutputWriter::printResultTag( ) [static,  
private]**

Print a line containing a result tag and no content.

Definition at line 270 of file ModelCheckingOutputWriter.cpp.

References multiscale::ConsolePrinter::printColouredMessage(), and TAG\_RESULT.

Referenced by printModelCheckingResultMessage().

**6.105.2.25 void ModelCheckingOutputWriter::printSeparatorTag( ) [static,  
private]**

Print a line containing a separator tag.

Definition at line 274 of file ModelCheckingOutputWriter.cpp.

References multiscale::ConsolePrinter::printColouredMessage(), and TAG\_SEPARAT-OR.

Referenced by printDetailedEvaluationResultsForLogicProperties(), printDetailed-EvaluationResultsIntroductionMessage(), printFailedMessage(), printModelChecking-ResultMessage(), printModelCheckingResultsIntroductionMessage(), printParsings-LogicPropertiesBeginMessage(), printStartModelCheckingExecutionMessage(), and printSuccessMessage().

**6.105.2.26 void ModelCheckingOutputWriter::printStartModelCheckingExecution-  
Message( ) [static]**

Print a message informing the user that the model checking execution has started.

Definition at line 57 of file ModelCheckingOutputWriter.cpp.

References MSG\_START\_MODEL\_CHECKING\_EXECUTION, multiscale::Console-Printer::printColouredMessage(), multiscale::ConsolePrinter::printMessageWith-ColouredTag(), printSeparatorTag(), and TAG\_EXECUTE.

Referenced by multiscale::verification::ModelCheckingManager::runModelCheckers-AndPrintMessage().

**6.105.2.27 void ModelCheckingOutputWriter::printStartTraceEvaluationMessage ( const std::string & *tracePath* ) [static]**

Print a message informing the user which trace will be evaluated next by the model checkers.

**Parameters**

|                  |                                        |
|------------------|----------------------------------------|
| <i>tracePath</i> | The path to the spatial-temporal trace |
|------------------|----------------------------------------|

Definition at line 63 of file ModelCheckingOutputWriter.cpp.

References MSG\_START\_TRACE\_EVALUATION, multiscale::ConsolePrinter::printMessageWithColouredTag(), and TAG\_EXECUTE.

Referenced by multiscale::verification::ModelCheckingManager::getNextSpatialTemporalTrace().

**6.105.2.28 void ModelCheckingOutputWriter::printSuccessMessage ( ) [static]**

Print a success message.

Definition at line 106 of file ModelCheckingOutputWriter.cpp.

References multiscale::ConsolePrinter::printColouredMessage(), printSeparatorTag(), and TAG\_SUCCESS.

Referenced by multiscale::verification::ModelCheckingManager::printParsingMessage().

**6.105.2.29 void ModelCheckingOutputWriter::printTimeoutMessage ( unsigned long *timeOut* ) [static]**

Print a message informing the user that the model checking execution is suspended for timeOut seconds.

Additionally let the user know that the list of traces is updated after the timeout

**Parameters**

|                |                   |
|----------------|-------------------|
| <i>timeOut</i> | The timeout value |
|----------------|-------------------|

Definition at line 72 of file ModelCheckingOutputWriter.cpp.

References MSG\_EXECUTION\_TIMEOUT\_BEGIN, MSG\_EXECUTION\_TIMEOUT-END, multiscale::ConsolePrinter::printMessageWithColouredTag(), TAG\_TIMEOUT, and multiscale::StringManipulator::toString().

Referenced by multiscale::verification::ModelCheckingManager::waitBeforeRetry().

---

**6.105.2.30 void ModelCheckingOutputWriter::printTraceEvaluationResult ( const std::string & *tracePath*, bool *evaluationResult* ) [static, private]**

Print the trace path with the associated evaluation result.

**Parameters**

|                         |                                        |
|-------------------------|----------------------------------------|
| <i>tracePath</i>        | The path to the spatial temporal trace |
| <i>evaluationResult</i> | The evaluation result                  |

Definition at line 242 of file ModelCheckingOutputWriter.cpp.

References printTruthValueDependentMessage(), TAG\_FALSE, and TAG\_TRUE.

Referenced by printDetailedTraceEvaluationResult().

**6.105.2.31 void ModelCheckingOutputWriter::printTruthValueDependentMessage ( const std::string & *message*, const std::string & *tag*, bool *truthValue* ) [static, private]**

Print a message with the given tag and colour depending on the truth value.

If the truthValue is true then the tag colour is green, otherwise red

**Parameters**

|                   |                                                       |
|-------------------|-------------------------------------------------------|
| <i>message</i>    | The given message                                     |
| <i>tag</i>        | The given tag                                         |
| <i>truthValue</i> | Boolean flag depending on which the tag colour is set |

Definition at line 260 of file ModelCheckingOutputWriter.cpp.

References multiscale::ConsolePrinter::printMessageWithColouredTag().

Referenced by printModelCheckingDetailedResult(), printModelCheckingResult(), and printTraceEvaluationResult().

**6.105.2.32 void ModelCheckingOutputWriter::updateSummaryEvaluationResults ( const std::size\_t & *logicPropertyIndex*, const std::size\_t & *tracePathIndex*, const std::vector< std::vector< bool >> & *evaluationResults*, size\_t & *nrOfEvaluatedTraces*, size\_t & *nrOfTracesEvaluatedTrue* ) [static, private]**

Update the summary evaluation results considering the logic property, trace and evaluation results.

**Parameters**

|                           |                                                                       |
|---------------------------|-----------------------------------------------------------------------|
| <i>logicPropertyIndex</i> | The index of the logic property in the collection of logic properties |
|---------------------------|-----------------------------------------------------------------------|

|                                  |                                                                                                                                                                                                                                                                                                                         |
|----------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>tracePath-Index</i>           | The index of the trace path in the collection of trace paths                                                                                                                                                                                                                                                            |
| <i>evaluation-Results</i>        | The evaluation results (i.e. a two-dimensional array of size $ \text{logicProperties}  \times  2 *  \text{traces}  $ where the first boolean value associated to a $(\text{logicProperty}, \text{trace})$ pair states if the logic property was evaluated for that trace and the second one stores the evaluation value |
| <i>nrOf-Evaluated-Traces</i>     | The number of evaluated traces                                                                                                                                                                                                                                                                                          |
| <i>nrOfTraces-Evaluated-True</i> | The number of traces evaluated true                                                                                                                                                                                                                                                                                     |

Definition at line 197 of file ModelCheckingOutputWriter.cpp.

References `isTraceEvaluatedForLogicProperty()`, and `isTraceEvaluatedTrueForLogicProperty()`.

Referenced by `printEvaluationResultsSummary()`.

### 6.105.3 Member Data Documentation

**6.105.3.1 const std::string ModelCheckingOutputWriter::MSG\_EVALUATION\_RESULTS\_INTRODUCTION = "I will display for each logic property which traces evaluated to TRUE and which evaluated to FALSE..." [static, private]**

Definition at line 290 of file ModelCheckingOutputWriter.hpp.

Referenced by `printDetailedEvaluationResultsIntroductionMessage()`.

**6.105.3.2 const std::string ModelCheckingOutputWriter::MSG\_EVALUATION\_SUMMARY\_BEGIN = "/" [static, private]**

Definition at line 291 of file ModelCheckingOutputWriter.hpp.

Referenced by `printEvaluationResultsSummary()`.

**6.105.3.3 const std::string ModelCheckingOutputWriter::MSG\_EVALUATION\_SUMMARY\_END = " spatio-temporal traces evaluated to TRUE" [static, private]**

Definition at line 292 of file ModelCheckingOutputWriter.hpp.

Referenced by `printEvaluationResultsSummary()`.

```
6.105.3.4 const std::string ModelCheckingOutputWriter::MSG_EXECUTION_TIMEO-
UT_BEGIN = "The model checker execution was suspended for " [static,
private]
```

Definition at line 285 of file ModelCheckingOutputWriter.hpp.

Referenced by printTimeoutMessage().

```
6.105.3.5 const std::string ModelCheckingOutputWriter::MSG_EXECUTION_TIMEO-
UT_END = " seconds during which new traces can be provided in the traces input
folder." [static, private]
```

Definition at line 286 of file ModelCheckingOutputWriter.hpp.

Referenced by printTimeoutMessage().

```
6.105.3.6 const std::string ModelCheckingOutputWriter::MSG_INIT_EXECUTION_P-
ARAMETERS = "Multidimensional multiscale model checking input parameters"
[static, private]
```

Definition at line 275 of file ModelCheckingOutputWriter.hpp.

Referenced by printInitialisationMessage().

```
6.105.3.7 const std::string ModelCheckingOutputWriter::MSG_INIT_EXTRA_-
EVALUATION_TIME = "Extra evaluation time (minutes): " [static,
private]
```

Definition at line 278 of file ModelCheckingOutputWriter.hpp.

Referenced by printInitialisationMessage().

```
6.105.3.8 const std::string ModelCheckingOutputWriter::MSG_INIT_LOGIC_-
PROPERTIES_PATH = "Logic properties input file: " [static,
private]
```

Definition at line 276 of file ModelCheckingOutputWriter.hpp.

Referenced by printInitialisationMessage().

```
6.105.3.9 const std::string ModelCheckingOutputWriter::MSG_INIT_TRACES-
_FOLDER_PATH = "Spatio-temporal traces input folder: " [static,
private]
```

Definition at line 277 of file ModelCheckingOutputWriter.hpp.

Referenced by printInitialisationMessage().

## **6.105 multiscale::verification::ModelCheckingOutputWriter Class Reference 607**

---

```
6.105.3.10 const std::string ModelCheckingOutputWriter::MSG_INTRO_CONTACT  
= "For more details, recommendations or suggestions feel free to contact me at  
<ovidiu.parvu[AT]gmail.com>." [static, private]
```

Definition at line 273 of file ModelCheckingOutputWriter.hpp.

Referenced by printIntroductionMessage().

```
6.105.3.11 const std::string ModelCheckingOutputWriter::MSG_INTRO_COPYRIGHT  
= "Copyright Ovidiu Pârvu 2014" [static, private]
```

Definition at line 270 of file ModelCheckingOutputWriter.hpp.

Referenced by printIntroductionMessage().

```
6.105.3.12 const std::string ModelCheckingOutputWriter::MSG_INTRO_MODEL_CHECKING_PARAMETERS  
= "Parameters: " [static, private]
```

Definition at line 272 of file ModelCheckingOutputWriter.hpp.

Referenced by printIntroductionMessage().

```
6.105.3.13 const std::string ModelCheckingOutputWriter::MSG_INTRO_MODEL_CHECKING_TYPE  
= "Model checker type: " [static, private]
```

Definition at line 271 of file ModelCheckingOutputWriter.hpp.

Referenced by printIntroductionMessage().

```
6.105.3.14 const std::string ModelCheckingOutputWriter::MSG_INTRO_NAME =  
"Mule 1.0.513 (Multidimensional multiscale model checker)" [static,  
private]
```

Definition at line 269 of file ModelCheckingOutputWriter.hpp.

Referenced by printIntroductionMessage().

```
6.105.3.15 const std::string ModelCheckingOutputWriter::MSG_LOGIC_PROPERTY_HOLDS  
= "The logic property holds: " [static, private]
```

Definition at line 294 of file ModelCheckingOutputWriter.hpp.

Referenced by printModelCheckingResult().

```
6.105.3.16 const std::string ModelCheckingOutputWriter::MSG_LOG-
IC_PROPERTY HOLDS FALSE = "FALSE" [static,
private]
```

Definition at line 296 of file ModelCheckingOutputWriter.hpp.

Referenced by printModelCheckingResult().

```
6.105.3.17 const std::string ModelCheckingOutputWriter::MSG_ЛО-
ГИЧ_PROPERTY HOLDS TRUE = "TRUE" [static,
private]
```

Definition at line 295 of file ModelCheckingOutputWriter.hpp.

Referenced by printModelCheckingResult().

```
6.105.3.18 const std::string ModelCheckingOutputWriter::MSG_PARSING_INT-
RODUCTION = "I am starting to parse logic properties..." [static,
private]
```

Definition at line 280 of file ModelCheckingOutputWriter.hpp.

Referenced by printParsingLogicPropertiesBeginMessage().

```
6.105.3.19 const std::string ModelCheckingOutputWriter::MSG_RESULTS_INTROD-
UCTION = "I have finished evaluating the logic properties and will display the
results..." [static, private]
```

Definition at line 288 of file ModelCheckingOutputWriter.hpp.

Referenced by printModelCheckingResultsIntroductionMessage().

```
6.105.3.20 const std::string ModelCheckingOutputWriter::MSG_START_EXTRA-
_EVALUATION_PROGRAM_EXECUTION = "I am starting the execution
of the extra evaluation program located at the following path:" [static,
private]
```

Definition at line 284 of file ModelCheckingOutputWriter.hpp.

Referenced by printExecuteExtraEvaluationProgramMessage().

```
6.105.3.21 const std::string ModelCheckingOutputWriter::MSG_START_MODEL_C-
HECKING_EXECUTION = "I am starting the execution of the model checkers..." [static,
private]
```

Definition at line 282 of file ModelCheckingOutputWriter.hpp.

Referenced by printStartModelCheckingExecutionMessage().

## **6.105 multiscale::verification::ModelCheckingOutputWriter Class Reference 609**

```
6.105.3.22 const std::string ModelCheckingOutputWriter::MSG_START_TRACE-
_EVALUATION = "Evaluating the spatio-temporal trace: " [static,
private]
```

Definition at line 283 of file ModelCheckingOutputWriter.hpp.

Referenced by printStartTraceEvaluationMessage().

```
6.105.3.23 const std::string ModelCheckingOutputWriter::TAG_DETAILS = "[ DETAILS
]" [static, private]
```

Definition at line 262 of file ModelCheckingOutputWriter.hpp.

Referenced by printDetailedEvaluationResultsForLogicProperties(), printDetailedEvaluationResultsIntroductionMessage(), and printEvaluationResultsSummary().

```
6.105.3.24 const std::string ModelCheckingOutputWriter::TAG_EXECUTE = "[ EXECUTE
]" [static, private]
```

Definition at line 259 of file ModelCheckingOutputWriter.hpp.

Referenced by printExecuteExtraEvaluationProgramMessage(), printStartModelCheckingExecutionMessage(), and printStartTraceEvaluationMessage().

```
6.105.3.25 const std::string ModelCheckingOutputWriter::TAG_FAILED = "[ FAILED ]"
[static, private]
```

Definition at line 264 of file ModelCheckingOutputWriter.hpp.

Referenced by printFailedMessage().

```
6.105.3.26 const std::string ModelCheckingOutputWriter::TAG_FALSE = "[ FALSE ]"
[static, private]
```

Definition at line 266 of file ModelCheckingOutputWriter.hpp.

Referenced by printTraceEvaluationResult().

```
6.105.3.27 const std::string ModelCheckingOutputWriter::TAG_INIT = "[ INIT ]"
[static, private]
```

Definition at line 257 of file ModelCheckingOutputWriter.hpp.

Referenced by printInitialisationMessage().

```
6.105.3.28 const std::string ModelCheckingOutputWriter::TAG_INTRO = "[ INTRO ]"  
[static, private]
```

Definition at line 256 of file ModelCheckingOutputWriter.hpp.

Referenced by printIntroductionMessage().

```
6.105.3.29 const std::string ModelCheckingOutputWriter::TAG_PARSING = "[  
PARSING ]" [static, private]
```

Definition at line 258 of file ModelCheckingOutputWriter.hpp.

Referenced by printParsingLogicPropertiesBeginMessage(), and printParsingLogicPropertyMessage().

```
6.105.3.30 const std::string ModelCheckingOutputWriter::TAG_RESULT = "[ RESULT  
]" [static, private]
```

Definition at line 261 of file ModelCheckingOutputWriter.hpp.

Referenced by printLogicPropertyForResult(), printModelCheckingDetailedResult(), printModelCheckingResult(), printModelCheckingResultsIntroductionMessage(), and printResultTag().

```
6.105.3.31 const std::string ModelCheckingOutputWriter::TAG_SEPARATOR =  
"[=====]" [static, private]
```

Definition at line 267 of file ModelCheckingOutputWriter.hpp.

Referenced by printSeparatorTag().

```
6.105.3.32 const std::string ModelCheckingOutputWriter::TAG_SUCCESS = "[  
SUCCESS ]" [static, private]
```

Definition at line 263 of file ModelCheckingOutputWriter.hpp.

Referenced by printSuccessMessage().

```
6.105.3.33 const std::string ModelCheckingOutputWriter::TAG_TIMEOUT = "[  
TIMEOUT ]" [static, private]
```

Definition at line 260 of file ModelCheckingOutputWriter.hpp.

Referenced by printTimeoutMessage().

**6.105.3.34 const std::string ModelCheckingOutputWriter::TAG\_TRUE = "[ TRUE ]"**  
[static, private]

Definition at line 265 of file ModelCheckingOutputWriter.hpp.

Referenced by printTraceEvaluationResult().

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

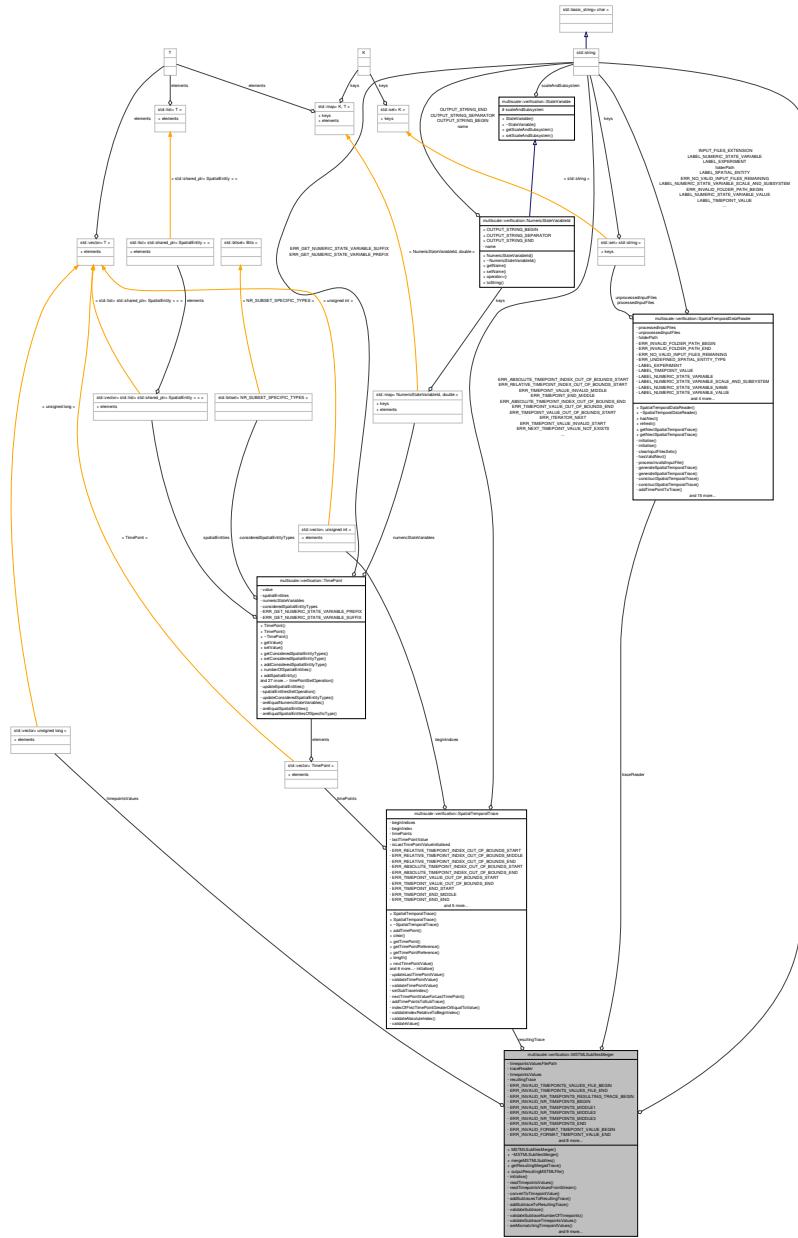
- ModelCheckingOutputWriter.hpp
  
- ModelCheckingOutputWriter.cpp

## **6.106 multiscale::verification::MSTMLSubfilesMerger Class Reference**

Class for merging MSTML subfiles.

```
#include <MSTMLSubfilesMerger.hpp>
```

## Collaboration diagram for multiscale::verification::MSTMLSubfilesMerger:



# Public Member Functions

- **MSTMLSubfilesMerger** (const std::string &mstmlSubfilesFolderPath, const std::string &timepointsValuesFilePath)
  - **~MSTMLSubfilesMerger** ()

- void [mergeMSTMLSubfiles \(\)](#)  
*Merge the MSTML subfiles from the provided folder considering the given timepoints values.*
- SpatialTemporalTrace [getResultingMergedTrace \(\)](#)  
*Get the resulting merged spatial temporal trace.*
- void [outputResultingMSTMLFile \(const std::string &mstmlFileOutputPath\)](#)  
*Output the resulting MSTML file to the file having the provided output path.*

### Private Member Functions

- void [initialise \(\)](#)  
*Initialisation function.*
- void [readTimepointsValues \(\)](#)  
*Read timepoints' values.*
- void [readTimepointsValuesFromStream \(std::ifstream &fin\)](#)  
*Read timepoints' values from the provided input stream.*
- unsigned long [convertToTimepointValue \(const std::string &timepointValueAsString\)](#)  
*Convert the provided string to a timepoint value (i.e. unsigned long)*
- void [addSubtracesToResultingTrace \(\)](#)  
*Add the subtraces corresponding to the MSTML subfiles to the resulting trace.*
- void [addSubtraceToResultingTrace \(const SpatialTemporalTrace &subtrace, const std::string &subtraceFilepath\)](#)  
*Add the provided spatial temporal subtrace to the resulting trace.*
- void [validateSubtrace \(const SpatialTemporalTrace &subtrace, const std::string &subtraceFilepath\)](#)  
*Validate the provided subtrace.*
- void [validateSubtraceNumberOfTimepoints \(const SpatialTemporalTrace &subtrace, const std::string &subtraceFilepath\)](#)  
*Check if the provided subtrace contains the correct number of timepoints.*
- void [validateSubtraceTimepointsValues \(const SpatialTemporalTrace &subtrace, const std::string &subtraceFilepath\)](#)  
*Check if the provided subtrace timepoints values are valid.*
- bool [areMismatchingTimepointValues \(const SpatialTemporalTrace &subtrace\)](#)  
*Check if there are any mismatching timepoint values in the provided and resulting trace.*
- void [addSubtraceStateVariablesToResultingTrace \(const SpatialTemporalTrace &subtrace, const std::string &subtraceFilepath\)](#)  
*Add the state variables from the provided to the resulting trace.*
- void [addSubtraceStateVariablesToEmptyResultingTrace \(const SpatialTemporalTrace &subtrace, const std::string &subtraceFilepath\)](#)  
*Add the state variables from the provided to the empty resulting trace.*
- void [addSubtraceStateVariablesToNonEmptyResultingTrace \(const SpatialTemporalTrace &subtrace, const std::string &subtraceFilepath\)](#)

- Add the state variables from the provided to the non-empty resulting trace.
  - void `addNumericStateVariablesToResultingTraceTimepoint` (const `TimePoint` &subtraceTimepoint, `TimePoint` &resultingTraceTimepoint, const `std::string` &subtraceFilepath)
    - Add numeric state variables from the provided subtrace to the resulting trace timepoint.
  - void `addNumericStateVariableToTimepoint` (const `NumericStateVariableId` &numericStateVariableId, double numericStateVariableValue, `TimePoint` &timepoint, const `std::string` &subtraceFilepath)
    - Add numeric state variable to timepoint.
  - void `addSpatialEntitiesToResultingTraceTimepoint` (const `TimePoint` &subtraceTimepoint, `TimePoint` &resultingTraceTimepoint, const `std::string` &subtraceFilepath)
    - Add spatial entities from the provided subtrace to the resulting trace timepoint.
  - void `addSpatialEntitiesToResultingTraceTimepoint` (const `SubsetSpecificType` &spatialEntityType, const `TimePoint` &subtraceTimepoint, `TimePoint` &resultingTraceTimepoint, const `std::string` &subtraceFilepath)
    - Add spatial entities of the given type from the provided subtrace to the resulting trace timepoint.
  - void `addSpatialEntityToTimepoint` (const `std::shared_ptr< SpatialEntity >` &spatialEntity, const `SubsetSpecificType` &spatialEntityType, `TimePoint` &timepoint, const `std::string` &subtraceFilepath)
    - Add spatial entity to timepoint.
  - void `updateResultingTraceTimepointsValues` ()
    - Replace the resulting trace timepoints values with the timepoints values read from file.
  - void `validateNumberOfTimepointsInResultingTrace` ()
    - Check if the number of timepoints is equal in the resulting trace and the timepoints values file.

## Private Attributes

- `std::string` `timepointsValuesFilePath`
- `SpatialTemporalDataReader` `traceReader`
- `std::vector< unsigned long >` `timepointsValues`
- `SpatialTemporalTrace` `resultingTrace`

## Static Private Attributes

- static const `std::string` `ERR_INVALID_TIMEPOINTS_VALUES_FILE_BEGIN` = "- The provided timepoints' values input file ("
- static const `std::string` `ERR_INVALID_TIMEPOINTS_VALUES_FILE_END` = ") could not be opened. Please make sure that the file path is valid and the file accessible."
- static const `std::string` `ERR_INVALID_NR_TIMEPOINTS_RESULTING_TRACE_BEGIN` = "The resulting MSTM trace"
- static const `std::string` `ERR_INVALID_NR_TIMEPOINTS_BEGIN` = "The MSTM subfile "

- static const std::string `ERR_INVALID_NR_TIMEPOINTS_MIDDLE1` = " contains "
- static const std::string `ERR_INVALID_NR_TIMEPOINTS_MIDDLE2` = " timepoints instead of the expected number of timepoints ("
- static const std::string `ERR_INVALID_NR_TIMEPOINTS_MIDDLE3` = ") specified in the timepoints values file "
- static const std::string `ERR_INVALID_NR_TIMEPOINTS_END` = ". Please change."
- static const std::string `ERR_INVALID_FORMAT_TIMEPOINT_VALUE_BEGIN` = "The provided timepoints values input file "
- static const std::string `ERR_INVALID_FORMAT_TIMEPOINT_VALUE_END` = " contains incorrectly formatted timepoint values. "
- static const std::string `ERR_NON_MATCHING_TIMEPOINT_VALUE_BEGIN` = "The MSTML subfile "
- static const std::string `ERR_NON_MATCHING_TIMEPOINT_VALUE_END` = ") which does not match the corresponding timepoint value from the resulting trace. Please change."
- static const std::string `ERR_NUMERIC_STATE_VARIABLE_EXISTS_BEGIN` = "The resulting trace contains a numeric state variable which has the same id "
- static const std::string `ERR_NUMERIC_STATE_VARIABLE_EXISTS_MIDDLE` = " as one of the numeric state variables in the subtrace "
- static const std::string `ERR_NUMERIC_STATE_VARIABLE_EXISTS_END` = ". Please update the subtrace such that the numeric state variable id is unique among all subtraces."
- static const std::string `ERR_SPATIAL_ENTITY_EXISTS_BEGIN` = "The resulting trace contains a spatial entity which has the same values ("
- static const std::string `ERR_SPATIAL_ENTITY_EXISTS_MIDDLE` = ") as one of the spatial entities in the subtrace "
- static const std::string `ERR_SPATIAL_ENTITY_EXISTS_END` = ". Please update the subtrace such that each spatial entity is unique among all subtraces."
- static const std::string `ERR_EMPTY_RESULTING_MSTML_FILE` = "The resulting trace should contain at least one timepoint but it does not. Please update the MSTML subfiles such that the resulting trace contains at least one timepoint."

### 6.106.1 Detailed Description

Class for merging MSTML subfiles.

Definition at line 13 of file MSTMLSubfilesMerger.hpp.

### 6.106.2 Constructor & Destructor Documentation

#### 6.106.2.1 MSTMLSubfilesMerger::MSTMLSubfilesMerger ( const std::string & `mstmlSubfilesFolderPath`, const std::string & `timepointsValuesFilePath` )

Definition at line 14 of file MSTMLSubfilesMerger.cpp.

References initialise().

### 6.106.2.2 **MSTMLSubfilesMerger::~MSTMLSubfilesMerger( )**

Definition at line 21 of file MSTMLSubfilesMerger.cpp.

### 6.106.3 Member Function Documentation

#### 6.106.3.1 **void MSTMLSubfilesMerger::addNumericStateVariablesToResultingTraceTimepoint ( const TimePoint & *subtraceTimepoint*, TimePoint & *resultingTraceTimepoint*, const std::string & *subtraceFilepath* ) [private]**

Add numeric state variables from the provided subtrace to the resulting trace timepoint.

##### Parameters

|                                  |                                                                  |
|----------------------------------|------------------------------------------------------------------|
| <i>subtrace-Timepoint</i>        | The provided subtrace timepoint                                  |
| <i>resulting-Trace-Timepoint</i> | The resulting trace timepoint                                    |
| <i>subtrace-Filepath</i>         | The path to the file containing the subtrace related information |

Definition at line 226 of file MSTMLSubfilesMerger.cpp.

References `addNumericStateVariableToTimepoint()`, `multiscale::verification::TimePoint::getNumericStateVariablesBeginIterator()`, and `multiscale::verification::TimePoint::getNumericStateVariablesEndIterator()`.

Referenced by `addSubtraceStateVariablesToNonEmptyResultingTrace()`.

#### 6.106.3.2 **void MSTMLSubfilesMerger::addNumericStateVariableToTimepoint ( const NumericStateVariableId & *numericStateVariableId*, double *numericStateVariableValue*, TimePoint & *timepoint*, const std::string & *subtraceFilepath* ) [private]**

Add numeric state variable to timepoint.

If an equal value numeric state variable already exists in the timepoint throw an exception.

##### Parameters

|                                     |                                                            |
|-------------------------------------|------------------------------------------------------------|
| <i>numeric-State-VariableId</i>     | The id of the numeric state variable                       |
| <i>numeric-State-Variable-Value</i> | The value of the numeric state variable                    |
| <i>timepoint</i>                    | The timepoint to which the numeric state variable is added |

|                          |                                                                      |
|--------------------------|----------------------------------------------------------------------|
| <i>subtrace-Filepath</i> | The path to the subtrace to which the numeric state variable belongs |
|--------------------------|----------------------------------------------------------------------|

Definition at line 246 of file MSTMLSubfilesMerger.cpp.

References multiscale::verification::TimePoint::addNumericStateVariable(), multiscale::verification::TimePoint::containsNumericStateVariable(), ERR\_NUMERIC\_STATE\_VARIABLE\_EXISTS\_BEGIN, ERR\_NUMERIC\_STATE\_VARIABLE\_EXISTS\_END, ERR\_NUMERIC\_STATE\_VARIABLE\_EXISTS\_MIDDLE, and multiscale::verification::NumericStateVariableId::toString().

Referenced by addNumericStateVariablesToResultingTraceTimepoint().

#### 6.106.3.3 void MSTMLSubfilesMerger::addSpatialEntitiesToResultingTraceTimepoint ( const TimePoint & *subtraceTimepoint*, TimePoint & *resultingTraceTimepoint*, const std::string & *subtraceFilepath* ) [private]

Add spatial entities from the provided subtrace to the resulting trace timepoint.

##### Parameters

|                                  |                                                                  |
|----------------------------------|------------------------------------------------------------------|
| <i>subtrace-Timepoint</i>        | The provided subtrace timepoint                                  |
| <i>resulting-Trace-Timepoint</i> | The resulting trace timepoint                                    |
| <i>subtrace-Filepath</i>         | The path to the file containing the subtrace related information |

Definition at line 266 of file MSTMLSubfilesMerger.cpp.

References multiscale::verification::subsetspecific::computeSubsetSpecificType(), and multiscale::verification::NR\_SUBSET\_SPECIFIC\_TYPES.

Referenced by addSubtraceStateVariablesToNonEmptyResultingTrace().

#### 6.106.3.4 void MSTMLSubfilesMerger::addSpatialEntitiesToResultingTraceTimepoint ( const SubsetSpecificType & *spatialEntityType*, const TimePoint & *subtraceTimepoint*, TimePoint & *resultingTraceTimepoint*, const std::string & *subtraceFilepath* ) [private]

Add spatial entities of the given type from the provided subtrace to the resulting trace timepoint.

##### Parameters

|                           |                                    |
|---------------------------|------------------------------------|
| <i>spatialEntityType</i>  | The considered spatial entity type |
| <i>subtrace-Timepoint</i> | The provided subtrace timepoint    |

|                                  |                                                                  |
|----------------------------------|------------------------------------------------------------------|
| <i>resulting-Trace-Timepoint</i> | The resulting trace timepoint                                    |
| <i>subtrace-Filepath</i>         | The path to the file containing the subtrace related information |

Definition at line 282 of file MSTMLSubfilesMerger.cpp.

References `addSpatialEntityToTimepoint()`, `multiscale::verification::TimePoint::getSpatialEntitiesBeginIterator()`, and `multiscale::verification::TimePoint::getSpatialEntitiesEndIterator()`.

```
6.106.3.5 void MSTMLSubfilesMerger::addSpatialEntityToTimepoint ( const
    std::shared_ptr< SpatialEntity > & spatialEntity, const SubsetSpecificType &
    spatialEntityType, TimePoint & timepoint, const std::string & subtraceFilepath )
    [private]
```

Add spatial entity to timepoint.

If an equal value spatial entity already exists in the timepoint throw an exception.

#### Parameters

|                           |                                                              |
|---------------------------|--------------------------------------------------------------|
| <i>spatialEntity</i>      | The considered spatial entity                                |
| <i>spatialEntity-Type</i> | The considered spatial entity type                           |
| <i>timepoint</i>          | The timepoint to which the spatial entity is added           |
| <i>subtrace-Filepath</i>  | The path to the subtrace to which the spatial entity belongs |

Definition at line 300 of file MSTMLSubfilesMerger.cpp.

References `multiscale::verification::TimePoint::addSpatialEntityAndType()`, `multiscale::verification::TimePoint::containsSpatialEntity()`, `ERR_SPATIAL_ENTITY_EXISTS_BEGIN`, `ERR_SPATIAL_ENTITY_EXISTS_END`, and `ERR_SPATIAL_ENTITY_EXISTS_MIDDLE`.

Referenced by `addSpatialEntitiesToResultingTraceTimepoint()`.

```
6.106.3.6 void MSTMLSubfilesMerger::addSubtraceStateVariablesToEmpty-
    ResultingTrace ( const SpatialTemporalTrace & subtrace, const std::string &
    subtraceFilepath ) [private]
```

Add the state variables from the provided to the empty resulting trace.

#### Parameters

|                          |                                                                  |
|--------------------------|------------------------------------------------------------------|
| <i>subtrace</i>          | The provided spatial temporal subtrace                           |
| <i>subtrace-Filepath</i> | The path to the file containing the subtrace related information |

Definition at line 196 of file MSTMLSubfilesMerger.cpp.

References multiscale::verification::SpatialTemporalTrace::addTimePoint(), multiscale::verification::SpatialTemporalTrace::getTimePoint(), resultingTrace, and timepointsValues.

Referenced by addSubtraceStateVariablesToResultingTrace().

**6.106.3.7 void MSTMLSubfilesMerger::addSubtraceStateVariablesToNonEmptyResultingTrace ( const SpatialTemporalTrace & *subtrace*, const std::string & *subtraceFilepath* ) [private]**

Add the state variables from the provided to the non-empty resulting trace.

#### Parameters

|                          |                                                                  |
|--------------------------|------------------------------------------------------------------|
| <i>subtrace</i>          | The provided spatial temporal subtrace                           |
| <i>subtrace-Filepath</i> | The path to the file containing the subtrace related information |

Definition at line 208 of file MSTMLSubfilesMerger.cpp.

References addNumericStateVariablesToResultingTraceTimepoint(), addSpatialEntitiesToResultingTraceTimepoint(), multiscale::verification::SpatialTemporalTrace::getTimePointReference(), resultingTrace, and timepointsValues.

Referenced by addSubtraceStateVariablesToResultingTrace().

**6.106.3.8 void MSTMLSubfilesMerger::addSubtraceStateVariablesToResultingTrace ( const SpatialTemporalTrace & *subtrace*, const std::string & *subtraceFilepath* ) [private]**

Add the state variables from the provided to the resulting trace.

#### Parameters

|                          |                                                                  |
|--------------------------|------------------------------------------------------------------|
| <i>subtrace</i>          | The provided spatial temporal subtrace                           |
| <i>subtrace-Filepath</i> | The path to the file containing the subtrace related information |

Definition at line 186 of file MSTMLSubfilesMerger.cpp.

References addSubtraceStateVariablesToEmptyResultingTrace(), addSubtraceStateVariablesToNonEmptyResultingTrace(), multiscale::verification::SpatialTemporalTrace::length(), and resultingTrace.

Referenced by addSubtraceToResultingTrace().

---

**6.106.3.9 void MSTMLSubfilesMerger::addSubtracesToResultingTrace ( )**  
[private]

Add the subtraces corresponding to the MSTML subfiles to the resulting trace.

Definition at line 102 of file MSTMLSubfilesMerger.cpp.

References addSubtraceToResultingTrace(), multiscale::verification::SpatialTemporalDataReader::getNextSpatialTemporalTrace(), multiscale::verification::SpatialTemporalDataReader::hasNext(), and traceReader.

Referenced by mergeMSTMLSubfiles().

**6.106.3.10 void MSTMLSubfilesMerger::addSubtraceToResultingTrace ( const SpatialTemporalTrace & subtrace, const std::string & subtraceFilepath )**  
[private]

Add the provided spatial temporal subtrace to the resulting trace.

**Parameters**

|                          |                                                                  |
|--------------------------|------------------------------------------------------------------|
| <i>subtrace</i>          | The provided spatial temporal subtrace                           |
| <i>subtrace-Filepath</i> | The path to the file containing the subtrace related information |

Definition at line 115 of file MSTMLSubfilesMerger.cpp.

References addSubtraceStateVariablesToResultingTrace(), and validateSubtrace().

Referenced by addSubtracesToResultingTrace().

**6.106.3.11 bool MSTMLSubfilesMerger::areMismatchingTimepointValues ( const SpatialTemporalTrace & subtrace )** [private]

Check if there are any mismatching timepoint values in the provided and resulting trace.

**Parameters**

|                 |                                        |
|-----------------|----------------------------------------|
| <i>subtrace</i> | The provided spatial temporal subtrace |
|-----------------|----------------------------------------|

Definition at line 165 of file MSTMLSubfilesMerger.cpp.

References multiscale::verification::SpatialTemporalTrace::getTimePointReference(), multiscale::verification::TimePoint::getValue(), multiscale::verification::SpatialTemporalTrace::length(), resultingTrace, and timepointsValues.

Referenced by validateSubtraceTimepointsValues().

6.106.3.12 `unsigned long MSTMLSubfilesMerger::convertToTimepointValue ( const std::string & timepointValueAsString ) [private]`

Convert the provided string to a timepoint value (i.e. unsigned long)

If the provided string cannot be converted to an unsigned long an error is thrown.

Parameters

|                                     |                                             |
|-------------------------------------|---------------------------------------------|
| <code>timepointValueAsString</code> | The timepoint value represented as a string |
|-------------------------------------|---------------------------------------------|

Definition at line 86 of file MSTMLSubfilesMerger.cpp.

References `ERR_INVALID_FORMAT_TIMEPOINT_VALUE_BEGIN`, `ERR_INVALID_FORMAT_TIMEPOINT_VALUE_END`, and `timepointsValuesFilePath`.

Referenced by `readTimepointsValuesFromStream()`.

6.106.3.13 `SpatialTemporalTrace MSTMLSubfilesMerger::getResultingMergedTrace ( )`

Get the resulting merged spatial temporal trace.

Definition at line 28 of file MSTMLSubfilesMerger.cpp.

References `resultingTrace`.

6.106.3.14 `void MSTMLSubfilesMerger::initialise ( ) [private]`

Initialisation function.

Definition at line 48 of file MSTMLSubfilesMerger.cpp.

References `readTimepointsValues()`.

Referenced by `MSTMLSubfilesMerger()`.

6.106.3.15 `void MSTMLSubfilesMerger::mergeMSTMLSubfiles ( )`

Merge the MSTML sufles from the provided folder considering the given timepoints values.

Definition at line 23 of file MSTMLSubfilesMerger.cpp.

References `addSubtracesToResultingTrace()`, and `updateResultingTraceTimepointsValues()`.

---

**6.106.3.16 void MSTMLSubfilesMerger::outputResultingMSTMLFile ( const std::string & *mstmlFileOutputPath* )**

Output the resulting MSTML file to the file having the provided output path.

If the number of timepoints in the trace is greater than zero output the trace to an xml file. Otherwise throw an exception.

**Parameters**

|                                        |                                               |
|----------------------------------------|-----------------------------------------------|
| <i>mstmlFile-</i><br><i>OutputPath</i> | The path to the resulting (merged) MSTML file |
|----------------------------------------|-----------------------------------------------|

Definition at line 32 of file MSTMLSubfilesMerger.cpp.

References ERR\_EMPTY\_RESULTING\_MSTML\_FILE, multiscale::verification::SpatialTemporalTrace::length(), multiscale::verification::SpatialTemporalDataWriter::outputTraceInXmlFormatToFile(), and resultingTrace.

**6.106.3.17 void MSTMLSubfilesMerger::readTimepointsValues ( ) [private]**

Read timepoints' values.

Definition at line 52 of file MSTMLSubfilesMerger.cpp.

References ERR\_INVALID\_TIMEPOINTS\_VALUES\_FILE\_BEGIN, ERR\_INVALID\_TIMEPOINTS\_VALUES\_FILE\_END, readTimepointsValuesFromStream(), and timepointsValuesFilePath.

Referenced by initialise().

**6.106.3.18 void MSTMLSubfilesMerger::readTimepointsValuesFromStream ( std::ifstream & *fin* ) [private]**

Read timepoints' values from the provided input stream.

The format of the timepoints values input file is: Line (L) 1: Header (usually "Time") L2: Timepoint value 1 L3: Timepoint value 2 ... ... Ln: Timepoint value n-1

**Parameters**

|            |                                    |
|------------|------------------------------------|
| <i>fin</i> | Input stream of timepoints' values |
|------------|------------------------------------|

Definition at line 69 of file MSTMLSubfilesMerger.cpp.

References convertToTimepointValue(), and timepointsValues.

Referenced by readTimepointsValues().

---

```
6.106.3.19 void MSTMLSubfilesMerger::updateResultingTraceTimepointsValues (
) [private]
```

Replace the resulting trace timepoints values with the timepoints values read from file.

If the number of timepoints in the timepoints values input file and the resulting trace differ throw an exception.

Definition at line 320 of file MSTMLSubfilesMerger.cpp.

References multiscale::verification::SpatialTemporalTrace::getTimePointReference(), resultingTrace, multiscale::verification::TimePoint::setValue(), timepointsValues, and validateNumberOfTimepointsInResultingTrace().

Referenced by mergeMSTMLSubfiles().

---

```
6.106.3.20 void MSTMLSubfilesMerger::validateNumberOfTimepointsInResulting-
Trace( ) [private]
```

Check if the number of timepoints is equal in the resulting trace and the timepoints values file.

Definition at line 335 of file MSTMLSubfilesMerger.cpp.

References ERR\_INVALID\_NR\_TIMEPOINTS\_END, ERR\_INVALID\_NR\_TIMEPOIN-  
TS\_MIDDLE1, ERR\_INVALID\_NR\_TIMEPOINTS\_MIDDLE2, ERR\_INVALID\_NR\_TI-  
MEPOINTS\_MIDDLE3, ERR\_INVALID\_NR\_TIMEPOINTS\_RESULTING\_TRACE\_BE-  
GIN, multiscale::verification::SpatialTemporalTrace::length(), resultingTrace, timepoints-  
Values, timepointsValuesFilePath, and multiscale::StringManipulator::toString().

Referenced by updateResultingTraceTimepointsValues().

---

```
6.106.3.21 void MSTMLSubfilesMerger::validateSubtrace ( const
SpatialTemporalTrace & subtrace, const std::string & subtraceFilepath )
[private]
```

Validate the provided subtrace.

A subtrace is valid if: 1. The number of timepoints it contains is equal to the number of timepoint values read from the provided timepointsValuesFilePath file; 2. The timepoints values of the subtrace match the timepoints values in the resulting trace.

#### Parameters

|                               |                                                                  |
|-------------------------------|------------------------------------------------------------------|
| <i>subtrace</i>               | The provided spatial temporal subtrace                           |
| <i>subtrace-<br/>Filepath</i> | The path to the file containing the subtrace related information |

Definition at line 122 of file MSTMLSubfilesMerger.cpp.

References validateSubtraceNumberOfTimepoints(), and validateSubtraceTimepoints-  
Values().

Referenced by addSubtraceToResultingTrace().

---

**6.106.3.22 void MSTMLSubfilesMerger::validateSubtraceNumberOfTimepoints ( const SpatialTemporalTrace & *subtrace*, const std::string & *subtraceFilepath* ) [private]**

Check if the provided subtrace contains the correct number of timepoints.

A subtrace is valid if the number of timepoints it contains is equal to the number of timepoint values read from the provided timepointsValuesFilePath file;

**Parameters**

|                          |                                                                  |
|--------------------------|------------------------------------------------------------------|
| <i>subtrace</i>          | The provided spatial temporal subtrace                           |
| <i>subtrace-Filepath</i> | The path to the file containing the subtrace related information |

Definition at line 128 of file MSTMLSubfilesMerger.cpp.

References ERR\_INVALID\_NR\_TIMEPOINTS\_BEGIN, ERR\_INVALID\_NR\_TIMEPOINTS\_END, ERR\_INVALID\_NR\_TIMEPOINTS\_MIDDLE1, ERR\_INVALID\_NR\_TIMEPOINTS\_MIDDLE2, ERR\_INVALID\_NR\_TIMEPOINTS\_MIDDLE3, multiscale::verification::SpatialTemporalTrace::length(), timepointsValues, and timepointsValuesFilePath.

Referenced by validateSubtrace().

**6.106.3.23 void MSTMLSubfilesMerger::validateSubtraceTimepointsValues ( const SpatialTemporalTrace & *subtrace*, const std::string & *subtraceFilepath* ) [private]**

Check if the provided subtrace timepoints values are valid.

A subtrace is valid if the timepoints values of the subtrace match the timepoints values in the resulting trace

**Parameters**

|                          |                                                                  |
|--------------------------|------------------------------------------------------------------|
| <i>subtrace</i>          | The provided spatial temporal subtrace                           |
| <i>subtrace-Filepath</i> | The path to the file containing the subtrace related information |

Definition at line 149 of file MSTMLSubfilesMerger.cpp.

References areMismatchingTimepointValues(), ERR\_NON\_MATCHING\_TIMEPOINT\_VALUE\_BEGIN, ERR\_NON\_MATCHING\_TIMEPOINT\_VALUE\_END, multiscale::verification::SpatialTemporalTrace::length(), and resultingTrace.

Referenced by validateSubtrace().

---

## 6.106.4 Member Data Documentation

6.106.4.1 `const std::string MSTMLSubfilesMerger::ERR_EMPTY_RESULTING_MST-ML_FILE = "The resulting trace should contain at least one timepoint but it does not. Please update the MSTML subfiles such that the resulting trace contains at least one timepoint." [static, private]`

Definition at line 248 of file MSTMLSubfilesMerger.hpp.

Referenced by outputResultingMSTMLFile().

6.106.4.2 `const std::string MSTMLSubfilesMerger::ERR_INVALID_FORMAT_TI-MEPOINT_VALUE_BEGIN = "The provided timepoints values input file " [static, private]`

Definition at line 234 of file MSTMLSubfilesMerger.hpp.

Referenced by convertToTimepointValue().

6.106.4.3 `const std::string MSTMLSubfilesMerger::ERR_INVALID_FORMAT_TIM-EPOINT_VALUE_END = " contains incorrectly formatted timepoint values. " [static, private]`

Definition at line 235 of file MSTMLSubfilesMerger.hpp.

Referenced by convertToTimepointValue().

6.106.4.4 `const std::string MSTMLSubfilesMerger::ERR_INVALID_NR_-TIMEPOINTS_BEGIN = "The MSTML subfile " [static, private]`

Definition at line 228 of file MSTMLSubfilesMerger.hpp.

Referenced by validateSubtraceNumberOfTimepoints().

6.106.4.5 `const std::string MSTMLSubfilesMerger::ERR_INVALID_-NR_TIMEPOINTS_END = ". Please change." [static, private]`

Definition at line 232 of file MSTMLSubfilesMerger.hpp.

Referenced by validateNumberOfTimepointsInResultingTrace(), and validateSubtrace-NumberOfTimepoints().

6.106.4.6 `const std::string MSTMLSubfilesMerger::ERR_INVALID_-NR_TIMEPOINTS_MIDDLE1 = " contains " [static, private]`

Definition at line 229 of file MSTMLSubfilesMerger.hpp.

Referenced by validateNumberOfTimepointsInResultingTrace(), and validateSubtraceNumberOfTimepoints().

6.106.4.7 `const std::string MSTMLSubfilesMerger::ERR_INVALID_NR_TIMEPOI-  
NTS_MIDDLE2 = " timepoints instead of the expected number of timepoints ("`  
[static, private]

Definition at line 230 of file MSTMLSubfilesMerger.hpp.

Referenced by validateNumberOfTimepointsInResultingTrace(), and validateSubtraceNumberOfTimepoints().

6.106.4.8 `const std::string MSTMLSubfilesMerger::ERR_INVALID_NR_TIMEPOI-  
NTS_MIDDLE3 = ") specified in the timepoints values file "` [static,  
private]

Definition at line 231 of file MSTMLSubfilesMerger.hpp.

Referenced by validateNumberOfTimepointsInResultingTrace(), and validateSubtraceNumberOfTimepoints().

6.106.4.9 `const std::string MSTMLSubfilesMerger::ERR_INVALID_NR_TIMEPOINTS-  
_RESULTING_TRACE_BEGIN = "The resulting MSTML trace"` [static,  
private]

Definition at line 226 of file MSTMLSubfilesMerger.hpp.

Referenced by validateNumberOfTimepointsInResultingTrace().

6.106.4.10 `const std::string MSTMLSubfilesMerger::ERR_INVALID_TIMEPOINTS-  
_VALUES_FILE_BEGIN = "The provided timepoints' values input file ("`  
[static, private]

Definition at line 223 of file MSTMLSubfilesMerger.hpp.

Referenced by readTimepointsValues().

6.106.4.11 `const std::string MSTMLSubfilesMerger::ERR_INVALID_TIMEPOINTS_V-  
ALUES_FILE_END = ") could not be opened. Please make sure that the file path is  
valid and the file accessible."` [static, private]

Definition at line 224 of file MSTMLSubfilesMerger.hpp.

Referenced by readTimepointsValues().

```
6.106.4.12 const std::string MSTMLSubfilesMerger::ERR_NON_MATCHING_-  
TIMEPOINT_VALUE_BEGIN = "The MSTML subfile " [static,  
private]
```

Definition at line 237 of file MSTMLSubfilesMerger.hpp.

Referenced by validateSubtraceTimepointsValues().

```
6.106.4.13 const std::string MSTMLSubfilesMerger::ERR_NON_MATCHING_TIMEP-  
OINT_VALUE_END = ") which does not match the corresponding timepoint value  
from the resulting trace. Please change." [static, private]
```

Definition at line 238 of file MSTMLSubfilesMerger.hpp.

Referenced by validateSubtraceTimepointsValues().

```
6.106.4.14 const std::string MSTMLSubfilesMerger::ERR_NUMERIC_STATE_VARIA-  
BLE_EXISTS_BEGIN = "The resulting trace contains a numeric state variable  
which has the same id " [static, private]
```

Definition at line 240 of file MSTMLSubfilesMerger.hpp.

Referenced by addNumericStateVariableToTimepoint().

```
6.106.4.15 const std::string MSTMLSubfilesMerger::ERR_NUMERIC_STATE_VARIA-  
BLE_EXISTS_END = ". Please update the subtrace such that the numeric state  
variable id is unique among all subtraces." [static, private]
```

Definition at line 242 of file MSTMLSubfilesMerger.hpp.

Referenced by addNumericStateVariableToTimepoint().

```
6.106.4.16 const std::string MSTMLSubfilesMerger::ERR_NUMERIC_STATE_VARIA-  
BLE_EXISTS_MIDDLE = "as one of the numeric state variables in the subtrace "  
[static, private]
```

Definition at line 241 of file MSTMLSubfilesMerger.hpp.

Referenced by addNumericStateVariableToTimepoint().

```
6.106.4.17 const std::string MSTMLSubfilesMerger::ERR_SPATIAL_ENTITY_EXIST-  
S_BEGIN = "The resulting trace contains a spatial entity which has the same values  
(" [static, private]
```

Definition at line 244 of file MSTMLSubfilesMerger.hpp.

Referenced by addSpatialEntityToTimepoint().

6.106.4.18 `const std::string MSTMLSubfilesMerger::ERR_SPATIAL_ENTITY_EXIST-S_END = ". Please update the subtrace such that each spatial entity is unique among all subtraces." [static, private]`

Definition at line 246 of file MSTMLSubfilesMerger.hpp.

Referenced by `addSpatialEntityToTimepoint()`.

6.106.4.19 `const std::string MSTMLSubfilesMerger::ERR_SPATIAL_ENTITY_EXIST-S_MIDDLE = ") as one of the spatial entities in the subtrace" [static, private]`

Definition at line 245 of file MSTMLSubfilesMerger.hpp.

Referenced by `addSpatialEntityToTimepoint()`.

6.106.4.20 `SpatialTemporalTrace multiscale::verification::MSTMLSubfilesMerger-::resultingTrace [private]`

The trace obtained after merging all subtraces corresponding to MSTML subfiles

Definition at line 29 of file MSTMLSubfilesMerger.hpp.

Referenced by `addSubtraceStateVariablesToEmptyResultingTrace()`, `addSubtraceStateVariablesToNonEmptyResultingTrace()`, `addSubtraceStateVariablesToResultingTrace()`, `areMismatchingTimepointValues()`, `getResultingMergedTrace()`, `outputResultingMSTMLFile()`, `updateResultingTraceTimepointsValues()`, `validateNumberOfTimepointsInResultingTrace()`, and `validateSubtraceTimepointsValues()`.

6.106.4.21 `std::vector<unsigned long> multiscale::verification::MSTMLSubfiles- Merger::timepointsValues [private]`

The considered timepoints' values

Definition at line 26 of file MSTMLSubfilesMerger.hpp.

Referenced by `addSubtraceStateVariablesToEmptyResultingTrace()`, `addSubtraceStateVariablesToNonEmptyResultingTrace()`, `areMismatchingTimepointValues()`, `readTimepointsValuesFromStream()`, `updateResultingTraceTimepointsValues()`, `validateNumberOfTimepointsInResultingTrace()`, and `validateSubtraceNumberOfTimepoints()`.

6.106.4.22 `std::string multiscale::verification::MSTMLSubfilesMerger::timepoints- ValuesFilePath [private]`

The path to the file containing the considered timepoints' values

Definition at line 18 of file MSTMLSubfilesMerger.hpp.

Referenced by `convertToTimepointValue()`, `readTimepointsValues()`, `validateNumberOfTimepointsInResultingTrace()`, and `validateSubtraceNumberOfTimepoints()`.

#### 6.106.4.23 SpatialTemporalDataReader multiscale::verification::MSTMLSubfilesMerger::traceReader [private]

The reader used to input spatial temporal traces from the MSTML subfiles

Definition at line 22 of file MSTMLSubfilesMerger.hpp.

Referenced by addSubtracesToResultingTrace().

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

- MSTMLSubfilesMerger.hpp
- MSTMLSubfilesMerger.cpp

## 6.107 multiscale::MultiplicationOperation Class Reference

Functor representing a multiplication operation.

```
#include <Numeric.hpp>
```

### Public Member Functions

- template<typename Operand >  
Operand **operator()** (Operand operand1, Operand operand2) const  
*Multiply the two operands.*

#### 6.107.1 Detailed Description

Functor representing a multiplication operation.

Definition at line 574 of file Numeric.hpp.

#### 6.107.2 Member Function Documentation

##### 6.107.2.1 template<typename Operand > Operand multiscale::MultiplicationOperation::operator() ( Operand *operand1*, Operand *operand2* ) const [inline]

Multiply the two operands.

#### Parameters

|                 |                    |
|-----------------|--------------------|
| <i>operand1</i> | The first operand  |
| <i>operand2</i> | The second operand |

Definition at line 584 of file Numeric.hpp.

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

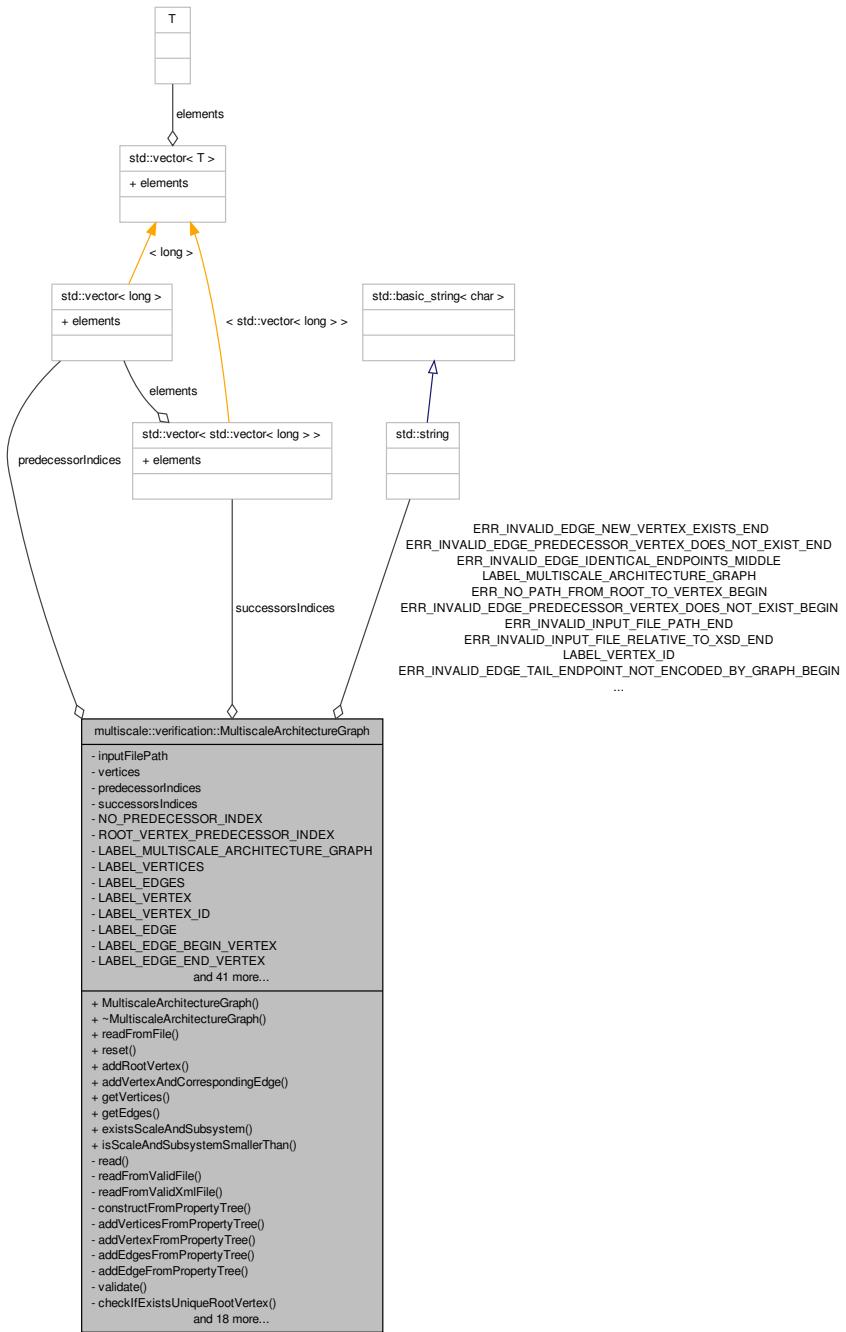
- Numeric.hpp

## 6.108 multiscale::verification::MultiscaleArchitectureGraph Class - Reference

Class for defining a multiscale architecture graph.

```
#include <MultiscaleArchitectureGraph.hpp>
```

## Collaboration diagram for multiscale::verification::MultiscaleArchitectureGraph:



## Public Member Functions

- `MultiscaleArchitectureGraph ()`
- `~MultiscaleArchitectureGraph ()`
- `void readFromFile (const std::string &inputFilePath)`  
`Read the multiscale architecture graph from the provided input file.`
- `void reset ()`  
`Reset the multiscale architecture graph such that it does not contain any vertices or edges.`
- `void addRootVertex (const std::string &rootVertexScaleAndSubsystem)`  
`Add the root vertex to the multiscale architecture graph using the provided scale and subsystem.`
- `void addVertexAndCorrespondingEdge (const std::string &newVertexScaleAndSubsystem, const std::string &predecessorVertexScaleAndSubsystem)`  
`Add a vertex and its corresponding incoming edge to the graph.`
- `std::vector< std::pair < std::string, long > > getVertices () const`  
`Return a copy of the vertices scales and subsystems, and indices.`
- `std::vector< std::pair < long, long > > getEdges () const`  
`Return a copy of the edges where the edges' endpoints are specified by vertex indices.`
- `bool existsScaleAndSubsystem (const std::string &scaleAndSubsystem) const`  
`Check if the given scale and subsystem exists in the multiscale architecture graph.`
- `bool isScaleAndSubsystemSmallerThan (const std::string &lhsScaleAndSubsystem, const std::string &rhsScaleAndSubsystem) const`  
`Check if the left hand side scale and subsystem < right hand side scale and subsystem.`

## Private Member Functions

- `void read ()`  
`Read the multiscale architecture graph from the provided input file.`
- `void readFromFileValidFile ()`  
`Read the multiscale architecture graph from the provided valid input file.`
- `void readFromValidXmlFile ()`  
`Read the multiscale architecture graph from the provided valid xml input file.`
- `void constructFromPropertyTree (const pt::ptree &propertyTree)`  
`Construct the multiscale architecture graph from the provided property tree.`
- `void addVerticesFromPropertyTree (const pt::ptree &propertyTree)`  
`Add vertices to the multiscale architecture graph.`
- `void addVertexFromPropertyTree (const pt::ptree &vertexPropertyTree)`  
`Add a vertex to the multiscale architecture graph corresponding to the given property tree.`
- `void addEdgesFromPropertyTree (const pt::ptree &propertyTree)`  
`Add the edges to the multiscale architecture graph.`
- `void addEdgeFromPropertyTree (const pt::ptree &edgePropertyTree)`

- **Add an edge to the multiscale architecture graph corresponding to the given property tree.**
  - void [validate \(\) const](#)  
*Check if the multiscale architecture graph is a rooted directed tree.*
  - void [checkIfExistsUniqueRootVertex \(\) const](#)  
*Check if the multiscale architecture graph contains a unique root vertex and throw exception if not.*
  - bool [isUniqueRootVertex \(\) const](#)  
*Check if the multiscale architecture graph contains a unique root vertex.*
  - void [checkIfValidIndegreeForNonRootVertices \(\) const](#)  
*Check if all vertices except the root have indegree 1 and throw exception if not.*
  - void [checkIfExistsPathFromRootToOtherVertices \(\) const](#)  
*Check if there is a path from the root to all other vertices and throw exception if not.*
  - void [checkIfExistsPathFromRootToOtherVertices \(long rootVertexIndex\) const](#)  
*Check if there is a path from the root to all other vertices and throw exception if not.*
  - void [checkIfExistUnvisitedVertices \(long rootVertexIndex, const std::vector< bool > &visitedVertices\) const](#)  
*Check if there exist unvisited vertices in the given collection and throw an exception if yes.*
  - bool [isEmpty \(\) const](#)  
*Check if the multiscale architecture graph does not contain vertices and edges.*
  - void [addRootVertexToEmptyMultiscaleArchitectureGraph \(const std::string &rootVertexScaleAndSubsystem\)](#)  
*Add the root vertex to the empty multiscale architecture graph using the given scale and subsystem.*
  - void [addVertexAndCorrespondingValidEdge \(const std::string &vertexScaleAndSubsystem, const std::string &predecessorVertexScaleAndSubsystem\)](#)  
*Add a vertex and its corresponding valid incoming edge to the graph.*
  - void [addVertexByScaleAndSubsystem \(const std::string &vertexScaleAndSubsystem\)](#)  
*Add a vertex with the given scale and subsystem to the graph.*
  - void [addEdgeByEndpoints \(const std::string &edgeTailEndpoint, const std::string &edgeHeadEndpoint\)](#)  
*Add an edge to the graph considering the given endpoints (i.e. scales and subsystems)*
  - void [checkIfPredecessorVertexExists \(const std::string &predecessorVertexScaleAndSubsystem, const std::string &newVertexScaleAndSubsystem\) const](#)  
*Check if predecessor vertex belongs to graph and throw exception if not.*
  - void [checkIfNewVertexDoesNotExist \(const std::string &predecessorVertexScaleAndSubsystem, const std::string &newVertexScaleAndSubsystem\) const](#)  
*Check if new vertex does not belong to graph and throw exception if it does.*
  - void [checkIfEdgeEndpointsAreDistinct \(const std::string &firstEdgeEndpoint, const std::string &secondEdgeEndpoint\) const](#)  
*Check if the provided edge endpoints are distinct and throw exception if not.*
  - void [checkIfEdgeEndpointsAreEncodedByGraph \(const std::string &firstEdgeEndpoint, const std::string &secondEdgeEndpoint\) const](#)

- void `checkIfEdgeHeadEndpointHasIndegreeZero` (const std::string &edgeTailEndpoint, const std::string &edgeHeadEndpoint) const
 

*Check if the provided edge head endpoint has indegree zero and throw exception if not.*
- void `checkIfComparedScaleAndSubsystemsAreEncodedByGraph` (const std::string &firstScaleAndSubsystem, const std::string &secondScaleAndSubsystem) const
 

*Check if the compared scales and subsystems are encoded by the graph and throw exception if not.*
- bool `isValidScaleAndSubsystemSmallerThan` (const std::string &lhsScaleAndSubsystem, const std::string &rhsScaleAndSubsystem) const
 

*Check if the valid left hand side scale and subsystem < right hand side scale and subsystem.*
- long `computeRootVertexIndex` () const
 

*Compute the index of the root vertex.*
- std::string `computeVertexName` (long vertexIndex) const
 

*Retrieve the name corresponding to the provided vertex index.*

### Private Attributes

- std::string `inputFilePath`
- std::unordered\_map<std::string, long> `vertices`
- std::vector<long> `predecessorIndices`
- std::vector<std::vector<long>> `successorsIndices`

### Static Private Attributes

- static const long `NO_PREDECESSOR_INDEX` = -1
- static const long `ROOT_VERTEX_PREDECESSOR_INDEX` = `NO_PREDECESSOR_INDEX`
- static const std::string `LABEL_MULTISCALE_ARCHITECTURE_GRAPH` = "multiscaleArchitectureGraph"
- static const std::string `LABEL_VERTICES` = ".vertices"
- static const std::string `LABEL_EDGES` = ".edges"
- static const std::string `LABEL_VERTEX` = "vertex"
- static const std::string `LABEL_VERTEX_ID` = "<xmattr>.id"
- static const std::string `LABEL_EDGE` = "edge"
- static const std::string `LABEL_EDGE_BEGIN_VERTEX` = "<xmattr>.beginVertex"
- static const std::string `LABEL_EDGE_END_VERTEX` = "<xmattr>.endVertex"
- static const std::string `MULTISCALE_ARCHITECTURE_GRAPH_INPUT_FILE_EXTENSION` = ".xml"

- static const std::string `MULTISCALE_ARCHITECTURE_GRAPH_XSD_PATH` = "/usr/local/share/mule/config/verification/spatial-temporal/schema/multiscale\_- architecture\_graph.xsd"
- static const std::string `ERR_INVALID_INPUT_FILE_PATH_BEGIN` = "The pro- vided input file path ("
- static const std::string `ERR_INVALID_INPUT_FILE_PATH_END` = "). Please change."
- static const std::string `ERR_INVALID_INPUT_FILE_RELATIVE_TO_XSD_BEG- IN` = "The provided xml input file ("
- static const std::string `ERR_INVALID_INPUT_FILE_RELATIVE_TO_XSD_END` = "). "
- static const std::string `ERR_ADD_ROOT_VERTEX_NON_EMPTY_GRAPH` = "It is not possible to add a root vertex to a non-empty multiscale architecture graph. Please clear the contents of the graph before adding a root vertex to it."
- static const std::string `ERR_ROOT_VERTEX_NOT_UNIQUE` = "The multiscale architecture graph is invalid because it does not contain a unique root vertex. Please change."
- static const std::string `ERR_NO_ROOT_VERTEX` = "The multiscale architecture graph should but does not contain any root vertices. Please change."
- static const std::string `ERR_NO_PATH_FROM_ROOT_TO_VERTEX_BEGIN` = "The multiscale architecture graph is invalid because there exists no path from the root vertex "
- static const std::string `ERR_NO_PATH_FROM_ROOT_TO_VERTEX_MIDDLE` = " to the vertex "
- static const std::string `ERR_NO_PATH_FROM_ROOT_TO_VERTEX_END` = ". Please change."
- static const std::string `ERR_INVALID_EDGE_BEGIN` = "The edge ("
- static const std::string `ERR_INVALID_EDGE_MIDDLE1` = ", "
- static const std::string `ERR_INVALID_EDGE_MIDDLE2` = ")"
- static const std::string `ERR_INVALID_EDGE_PREDECESSOR_VERTEX_DOE- S_NOT_EXIST_BEGIN` = `ERR_INVALID_EDGE_BEGIN`
- static const std::string `ERR_INVALID_EDGE_PREDECESSOR_VERTEX_DOE- S_NOT_EXIST_MIDDLE1` = `ERR_INVALID_EDGE_MIDDLE1`
- static const std::string `ERR_INVALID_EDGE_PREDECESSOR_VERTEX_DO- ES_NOT_EXIST_MIDDLE2` = " cannot be added to the multiscale architecture graph because the edge tail endpoint \ ""
- static const std::string `ERR_INVALID_EDGE_PREDECESSOR_VERTEX_DOE- S_NOT_EXIST_END` = "\ " is not a vertex of the graph. Please change."
- static const std::string `ERR_INVALID_EDGE_NEW_VERTEX_EXISTS_BEGIN` = `ERR_INVALID_EDGE_BEGIN`
- static const std::string `ERR_INVALID_EDGE_NEW_VERTEX_EXISTS_MIDD- E1` = `ERR_INVALID_EDGE_MIDDLE1`
- static const std::string `ERR_INVALID_EDGE_NEW_VERTEX_EXISTS_MIDD- E2` = " cannot be added to the multiscale architecture graph because the edge head endpoint \ ""
- static const std::string `ERR_INVALID_EDGE_NEW_VERTEX_EXISTS_END` = "\ " is not a new but an existing vertex of the graph. Please change."

- static const std::string `ERR_INVALID_EDGE_IDENTICAL_ENDPOINTS_BEGIN` = `ERR_INVALID_EDGE_BEGIN`
- static const std::string `ERR_INVALID_EDGE_IDENTICAL_ENDPOINTS_MIDDLE` = `ERR_INVALID_EDGE_MIDDLE1`
- static const std::string `ERR_INVALID_EDGE_IDENTICAL_ENDPOINTS_END` = " cannot be added to the multiscale architecture graph because the edge endpoints are identical and they should be distinct. Please change."
- static const std::string `ERR_INVALID_EDGE_TAIL_ENDPOINT_NOT_ENCODED_BY_GRAPH_BEGIN` = `ERR_INVALID_EDGE_BEGIN`
- static const std::string `ERR_INVALID_EDGE_TAIL_ENDPOINT_NOT_ENCODED_BY_GRAPH_MIDDLE1` = `ERR_INVALID_EDGE_MIDDLE1`
- static const std::string `ERR_INVALID_EDGE_TAIL_ENDPOINT_NOT_ENCODED_BY_GRAPH_MIDDLE2` = " cannot be added to the multiscale architecture graph because the edge tail endpoint \\""
- static const std::string `ERR_INVALID_EDGE_TAIL_ENDPOINT_NOT_ENCODED_BY_GRAPH_END` = "\\" is not a vertex of the graph. Please change."
- static const std::string `ERR_INVALID_EDGE_HEAD_ENDPOINT_NOT_ENCODED_BY_GRAPH_BEGIN` = `ERR_INVALID_EDGE_BEGIN`
- static const std::string `ERR_INVALID_EDGE_HEAD_ENDPOINT_NOT_ENCODED_BY_GRAPH_MIDDLE1` = `ERR_INVALID_EDGE_MIDDLE1`
- static const std::string `ERR_INVALID_EDGE_HEAD_ENDPOINT_NOT_ENCODED_BY_GRAPH_MIDDLE2` = " cannot be added to the multiscale architecture graph because the edge head endpoint \\""
- static const std::string `ERR_INVALID_EDGE_HEAD_ENDPOINT_NOT_ENCODED_BY_GRAPH_END` = `ERR_INVALID_EDGE_TAIL_ENDPOINT_NOT_ENCODED_BY_GRAPH_END`
- static const std::string `ERR_INVALID_EDGE_HEAD_ENDPOINT_INDEGREE_NOT_ZERO_BEGIN` = `ERR_INVALID_EDGE_BEGIN`
- static const std::string `ERR_INVALID_EDGE_HEAD_ENDPOINT_INDEGREE_NOT_ZERO_MIDDLE1` = `ERR_INVALID_EDGE_MIDDLE1`
- static const std::string `ERR_INVALID_EDGE_HEAD_ENDPOINT_INDEGREE_NOT_ZERO_MIDDLE2` = " cannot be added to the multiscale architecture graph because the edge head endpoint \\""
- static const std::string `ERR_INVALID_EDGE_HEAD_ENDPOINT_INDEGREE_NOT_ZERO_END` = "\\" has an indegree different from zero. Please change."
- static const std::string `ERR_COMPARE_SCALE_AND_SUBSYSTEMS_NOT_ENCODED_BY_GRAPH_BEGIN` = "The provided scales and subsystems \\""
- static const std::string `ERR_COMPARE_SCALE_AND_SUBSYSTEMS_NOT_ENCODED_BY_GRAPH_MIDDLE1` = "\\" and \\""
- static const std::string `ERR_COMPARE_SCALE_AND_SUBSYSTEMS_NOT_ENCODED_BY_GRAPH_MIDDLE2` = "\\" cannot be compared because the scale and subsystem \\""
- static const std::string `ERR_COMPARE_SCALE_AND_SUBSYSTEMS_NOT_ENCODED_BY_GRAPH_END` = "\\" is not encoded by the multiscale architecture graph. Please change."

### **6.108.1 Detailed Description**

Class for defining a multiscale architecture graph.

The multiscale architecture graph encodes the hierarchical organization of the scales and subsystems and is represented as a rooted directed tree.

Definition at line 21 of file MultiscaleArchitectureGraph.hpp.

### **6.108.2 Constructor & Destructor Documentation**

#### **6.108.2.1 MultiscaleArchitectureGraph::MultiscaleArchitectureGraph( )**

Definition at line 13 of file MultiscaleArchitectureGraph.cpp.

#### **6.108.2.2 MultiscaleArchitectureGraph::~MultiscaleArchitectureGraph( )**

Definition at line 15 of file MultiscaleArchitectureGraph.cpp.

### **6.108.3 Member Function Documentation**

#### **6.108.3.1 void MultiscaleArchitectureGraph::addEdgeByEndpoints( const std::string & edgeTailEndpoint, const std::string & edgeHeadEndpoint ) [private]**

Add an edge to the graph considering the given endpoints (i.e. scales and subsystems)

Precondition: The edge is valid and can be added to the graph.

##### **Parameters**

|                          |                                                                             |
|--------------------------|-----------------------------------------------------------------------------|
| <i>edgeTail-Endpoint</i> | The edge tail end point identified by its corresponding scale and subsystem |
| <i>edgeHead-Endpoint</i> | The edge head end point identified by its corresponding scale and subsystem |

Definition at line 347 of file MultiscaleArchitectureGraph.cpp.

References predecessorIndices, successorsIndices, and vertices.

Referenced by addEdgeFromPropertyTree().

#### **6.108.3.2 void MultiscaleArchitectureGraph::addEdgeFromPropertyTree( const pt::ptree & edgePropertyTree ) [private]**

Add an edge to the multiscale architecture graph corresponding to the given property tree.

**Parameters**

|                                     |                                            |
|-------------------------------------|--------------------------------------------|
| <i>edge-<br/>Property-<br/>Tree</i> | The property tree corresponding to an edge |
|-------------------------------------|--------------------------------------------|

Definition at line 188 of file MultiscaleArchitectureGraph.cpp.

References addEdgeByEndpoints(), checkIfEdgeEndpointsAreDistinct(), checkIfEdgeEndpointsAreEncodedByGraph(), checkIfEdgeHeadEndpointHasIndegreeZero(), LABEL\_EDGE\_BEGIN\_VERTEX, and LABEL\_EDGE\_END\_VERTEX.

Referenced by addEdgesFromPropertyTree().

#### 6.108.3.3 void MultiscaleArchitectureGraph::addEdgesFromPropertyTree ( const pt::ptree & *propertyTree* ) [private]

Add the edges to the multiscale architecture graph.

**Parameters**

|                     |                                                       |
|---------------------|-------------------------------------------------------|
| <i>propertyTree</i> | The property tree corresponding to the xml input file |
|---------------------|-------------------------------------------------------|

Definition at line 179 of file MultiscaleArchitectureGraph.cpp.

References addEdgeFromPropertyTree(), and LABEL\_EDGES.

Referenced by constructFromPropertyTree().

#### 6.108.3.4 void MultiscaleArchitectureGraph::addRootVertex ( const std::string & *rootVertexScaleAndSubsystem* )

Add the root vertex to the multiscale architecture graph using the provided scale and subsystem.

Precondition: The provided scale and subsystem is in a valid format.

**Parameters**

|                                                |                                            |
|------------------------------------------------|--------------------------------------------|
| <i>rootVertex-<br/>ScaleAnd-<br/>Subsystem</i> | The scale and subsystem of the root vertex |
|------------------------------------------------|--------------------------------------------|

Definition at line 34 of file MultiscaleArchitectureGraph.cpp.

References addRootVertexToEmptyMultiscaleArchitectureGraph(), ERR\_ADD\_ROOT\_VERTEX\_NON\_EMPTY\_GRAPH, and isEmpty().

Referenced by multiscaletest::TraceEvaluationTest::InitialiseMultiscaleArchitectureGraph().

**6.108.3.5 void MultiscaleArchitectureGraph::addRootVertexToEmptyMultiscaleArchitectureGraph ( const std::string & *rootVertexScaleAndSubsystem* ) [private]**

Add the root vertex to the empty multiscale architecture graph using the given scale and subsystem.

Precondition: The number of vertices and edges in the graph is 0.

**Parameters**

|                                      |                                            |
|--------------------------------------|--------------------------------------------|
| <i>rootVertex-ScaleAnd-Subsystem</i> | The scale and subsystem of the root vertex |
|--------------------------------------|--------------------------------------------|

Definition at line 308 of file MultiscaleArchitectureGraph.cpp.

References addVertexByScaleAndSubsystem(), predecessorIndices, and ROOT\_VERTEX\_PREDECESSOR\_INDEX.

Referenced by addRootVertex().

**6.108.3.6 void MultiscaleArchitectureGraph::addVertexAndCorrespondingEdge ( const std::string & *newVertexScaleAndSubsystem*, const std::string & *predecessorVertexScaleAndSubsystem* )**

Add a vertex and its corresponding incoming edge to the graph.

The method requires specifying both the vertex and its predecessor because this will ensure that the resulting graph is connected and acyclic.

The intuitive proof for the (weakly) connected property is that all vertices except the root will have indegree 1 and there will exist a directed path from the root to them.

The intuitive proof for the acyclic property is that the number of edges in the graph will be always |vertices| - 1, where |vertices| represents the number of elements in the set vertices.

**Parameters**

|                                              |                                                                                                       |
|----------------------------------------------|-------------------------------------------------------------------------------------------------------|
| <i>newVertex-ScaleAnd-Subsystem</i>          | The scale and subsystem of the new vertex to be added to the multiscale architecture graph            |
| <i>predecessor-VertexScale-And-Subsystem</i> | The scale and subsystem of the existing vertex which will be the predecessor/parent of the new vertex |

Definition at line 46 of file MultiscaleArchitectureGraph.cpp.

References addVertexAndCorrespondingValidEdge(), checkIfNewVertexDoesNotExist(), and checkIfPredecessorVertexExists().

Referenced by multiscaletest::TraceEvaluationTest::InitialiseMultiscaleArchitectureGraph().

**6.108.3.7 void MultiscaleArchitectureGraph::addVertexAndCorrespondingValidEdge ( const std::string & *vertexScaleAndSubsystem*, const std::string & *predecessorVertexScaleAndSubsystem* ) [private]**

Add a vertex and its corresponding valid incoming edge to the graph.

**Parameters**

|                                              |                                                                                                       |
|----------------------------------------------|-------------------------------------------------------------------------------------------------------|
| <i>vertexScale-And-Subsystem</i>             | The scale and subsystem of the new vertex to be added to the multi-scale architecture graph           |
| <i>predecessor-VertexScale-And-Subsystem</i> | The scale and subsystem of the existing vertex which will be the predecessor/parent of the new vertex |

Definition at line 317 of file MultiscaleArchitectureGraph.cpp.

References addVertexByScaleAndSubsystem(), predecessorIndices, successorsIndices, and vertices.

Referenced by addVertexAndCorrespondingEdge().

**6.108.3.8 void MultiscaleArchitectureGraph::addVertexByScaleAndSubsystem ( const std::string & *vertexScaleAndSubsystem* ) [private]**

Add a vertex with the given scale and subsystem to the graph.

Precondition: The provided scale and subsystem is in a valid format.

**Parameters**

|                                  |                                       |
|----------------------------------|---------------------------------------|
| <i>vertexScale-And-Subsystem</i> | The scale and subsystem of the vertex |
|----------------------------------|---------------------------------------|

Definition at line 333 of file MultiscaleArchitectureGraph.cpp.

References NO\_PREDECESSOR\_INDEX, predecessorIndices, successorsIndices, and vertices.

Referenced by addRootVertexToEmptyMultiscaleArchitectureGraph(), addVertexAndCorrespondingValidEdge(), and addVertexFromPropertyTree().

## **6.108 multiscale::verification::MultiscaleArchitectureGraph Class Reference 641**

**6.108.3.9 void MultiscaleArchitectureGraph::addVertexFromPropertyTree ( const pt::ptree & *vertexPropertyTree* ) [private]**

Add a vertex to the multiscale architecture graph corresponding to the given property tree.

### Parameters

|                                       |                                             |
|---------------------------------------|---------------------------------------------|
| <i>vertex-<br/>Property-<br/>Tree</i> | The property tree corresponding to a vertex |
|---------------------------------------|---------------------------------------------|

Definition at line 171 of file MultiscaleArchitectureGraph.cpp.

References addVertexByScaleAndSubsystem(), and LABEL\_VERTEX\_ID.

Referenced by addVerticesFromPropertyTree().

**6.108.3.10 void MultiscaleArchitectureGraph::addVerticesFromPropertyTree ( const pt::ptree & *propertyTree* ) [private]**

Add vertices to the multiscale architecture graph.

### Parameters

|                     |                                                       |
|---------------------|-------------------------------------------------------|
| <i>propertyTree</i> | The property tree corresponding to the xml input file |
|---------------------|-------------------------------------------------------|

Definition at line 162 of file MultiscaleArchitectureGraph.cpp.

References addVertexFromPropertyTree(), LABEL\_VERTICES, and vertices.

Referenced by constructFromPropertyTree().

**6.108.3.11 void MultiscaleArchitectureGraph::checkIfComparedScale-  
AndSubsystemsAreEncodedByGraph ( const std::string &  
firstScaleAndSubsystem, const std::string & secondScaleAndSubsystem ) const  
[private]**

Check if the compared scales and subsystems are encoded by the graph and throw exception if not.

This function is called to check if the scales and subsystems are encoded by the graph before attempting to compare them.

### Parameters

|                                            |                                           |
|--------------------------------------------|-------------------------------------------|
| <i>firstScale-<br/>And-<br/>Subsystem</i>  | The first considered scale and subsystem  |
| <i>second-<br/>ScaleAnd-<br/>Subsystem</i> | The second considered scale and subsystem |

Definition at line 457 of file MultiscaleArchitectureGraph.cpp.

References `ERR_COMPARE_SCALE_AND_SUBSYSTEMS_NOT_ENCODED_BY_GRAPH_BEGIN`, `ERR_COMPARE_SCALE_AND_SUBSYSTEMS_NOT_ENCODED_BY_GRAPH_END`, `ERR_COMPARE_SCALE_AND_SUBSYSTEMS_NOT_ENCODED_BY_GRAPH_MIDDLE1`, `ERR_COMPARE_SCALE_AND_SUBSYSTEMS_NOT_ENCODED_BY_GRAPH_MIDDLE2`, and `existsScaleAndSubsystem()`.

Referenced by `isScaleAndSubsystemSmallerThan()`.

```
6.108.3.12 void MultiscaleArchitectureGraph::checkIfEdgeEndpointsAreDistinct (
    const std::string & firstEdgeEndpoint, const std::string & secondEdgeEndpoint )
    const [private]
```

Check if the provided edge endpoints are distinct and throw exception if not.

#### Parameters

|                                       |                                                                |
|---------------------------------------|----------------------------------------------------------------|
| <i>firstEdge-<br/>Endpoint</i>        | The first considered edge endpoint (i.e. scale and subsystem)  |
| <i>second-<br/>Edge-<br/>Endpoint</i> | The second considered edge endpoint (i.e. scale and subsystem) |

Definition at line 394 of file MultiscaleArchitectureGraph.cpp.

References `ERR_INVALID_EDGE_IDENTICAL_ENDPOINTS_BEGIN`, `ERR_INVALID_EDGE_IDENTICAL_ENDPOINTS_END`, and `ERR_INVALID_EDGE_IDENTICAL_ENDPOINTS_MIDDLE`.

Referenced by `addEdgeFromPropertyTree()`.

```
6.108.3.13 void MultiscaleArchitectureGraph::checkIfEdgeEndpointsAre-
EncodedByGraph ( const std::string & firstEdgeEndpoint, const std::string &
secondEdgeEndpoint ) const [private]
```

Check if the provided edge endpoints are encoded by the graph and throw exception if not.

#### Parameters

|                                       |                                                                |
|---------------------------------------|----------------------------------------------------------------|
| <i>firstEdge-<br/>Endpoint</i>        | The first considered edge endpoint (i.e. scale and subsystem)  |
| <i>second-<br/>Edge-<br/>Endpoint</i> | The second considered edge endpoint (i.e. scale and subsystem) |

Definition at line 409 of file MultiscaleArchitectureGraph.cpp.

References `ERR_INVALID_EDGE_HEAD_ENDPOINT_NOT_ENCODED_BY_GRAPH_BEGIN`, `ERR_INVALID_EDGE_HEAD_ENDPOINT_NOT_ENCODED_BY_GRAPH_END`.

H\_END, ERR\_INVALID\_EDGE\_HEAD\_ENDPOINT\_NOT\_ENCODED\_BY\_GRAPH\_MIDDLE1, ERR\_INVALID\_EDGE\_HEAD\_ENDPOINT\_NOT\_ENCODED\_BY\_GRAPH\_MIDDLE2, ERR\_INVALID\_EDGE\_TAIL\_ENDPOINT\_NOT\_ENCODED\_BY\_GRAPH\_BEGIN, ERR\_INVALID\_EDGE\_TAIL\_ENDPOINT\_NOT\_ENCODED\_BY\_GRAPH\_END, ERR\_INVALID\_EDGE\_TAIL\_ENDPOINT\_NOT\_ENCODED\_BY\_GRAPH\_MIDDLE1, ERR\_INVALID\_EDGE\_TAIL\_ENDPOINT\_NOT\_ENCODED\_BY\_GRAPH\_MIDDLE2, and existsScaleAndSubsystem().

Referenced by addEdgeFromPropertyTree().

**6.108.3.14 void MultiscaleArchitectureGraph::checkIfEdgeHeadEndpointHasIndegreeZero ( const std::string & *edgeTailEndpoint*, const std::string & *edgeHeadEndpoint* ) const [private]**

Check if the provided edge head endpoint has indegree zero and throw exception if not.

**Parameters**

|                          |                                                                             |
|--------------------------|-----------------------------------------------------------------------------|
| <i>edgeTail-Endpoint</i> | The edge tail end point identified by its corresponding scale and subsystem |
| <i>edgeHead-Endpoint</i> | The edge head end point identified by its corresponding scale and subsystem |

Definition at line 437 of file MultiscaleArchitectureGraph.cpp.

References ERR\_INVALID\_EDGE\_HEAD\_ENDPOINT\_INDEGREE\_NOT\_ZERO\_BEGIN, ERR\_INVALID\_EDGE\_HEAD\_ENDPOINT\_INDEGREE\_NOT\_ZERO\_END, ERR\_INVALID\_EDGE\_HEAD\_ENDPOINT\_INDEGREE\_NOT\_ZERO\_MIDDLE1, ERR\_INVALID\_EDGE\_HEAD\_ENDPOINT\_INDEGREE\_NOT\_ZERO\_MIDDLE2, NO\_PREDECESSOR\_INDEX, predecessorIndices, and vertices.

Referenced by addEdgeFromPropertyTree().

**6.108.3.15 void MultiscaleArchitectureGraph::checkIfExistsPathFromRootToOtherVertices ( ) const [private]**

Check if there is a path from the root to all other vertices and throw exception if not.

Definition at line 243 of file MultiscaleArchitectureGraph.cpp.

References computeRootVertexIndex().

Referenced by validate().

**6.108.3.16 void MultiscaleArchitectureGraph::checkIfExistsPathFromRootToOtherVertices ( long *rootVertexIndex* ) const [private]**

Check if there is a path from the root to all other vertices and throw exception if not.

Using the successor relation we will check if all vertices are (in)directly connected to the

root vertex. If not an exception is thrown.

**Parameters**

|                         |                                        |
|-------------------------|----------------------------------------|
| <i>rootVertex-Index</i> | Index corresponding to the root vertex |
|-------------------------|----------------------------------------|

Definition at line 251 of file MultiscaleArchitectureGraph.cpp.

References checkIfExistUnvisitedVertices(), successorsIndices, and vertices.

**6.108.3.17 void MultiscaleArchitectureGraph::checkIfExistsUniqueRootVertex ( ) const [private]**

Check if the multiscale architecture graph contains a unique root vertex and throw exception if not.

Definition at line 209 of file MultiscaleArchitectureGraph.cpp.

References ERR\_ROOT\_VERTEX\_NOT\_UNIQUE, and isUniqueRootVertex().

Referenced by validate().

**6.108.3.18 void MultiscaleArchitectureGraph::checkIfExistUnvisitedVertices ( long *rootVertexIndex*, const std::vector< bool > & *visitedVertices* ) const [private]**

Check if there exist unvisited vertices in the given collection and throw an exception if yes.

**Parameters**

|                         |                                                                                |
|-------------------------|--------------------------------------------------------------------------------|
| <i>rootVertex-Index</i> | Index corresponding to the root vertex                                         |
| <i>visited-Vertices</i> | Collection recording which vertices can be visited when starting from the root |

Definition at line 280 of file MultiscaleArchitectureGraph.cpp.

References computeVertexName(), ERR\_NO\_PATH\_FROM\_ROOT\_TO\_VERTEX\_BEGIN, ERR\_NO\_PATH\_FROM\_ROOT\_TO\_VERTEX\_END, and ERR\_NO\_PATH\_FROM\_ROOT\_TO\_VERTEX\_MIDDLE.

Referenced by checkIfExistsPathFromRootToOtherVertices().

**6.108.3.19 void MultiscaleArchitectureGraph::checkIfNewVertexDoesNotExist ( const std::string & *predecessorVertexScaleAndSubsystem*, const std::string & *newVertexScaleAndSubsystem* ) const [private]**

Check if new vertex does not belong to graph and throw exception if it does.

## **6.108 multiscale::verification::MultiscaleArchitectureGraph Class Reference 645**

### Parameters

|                                              |                                                                 |
|----------------------------------------------|-----------------------------------------------------------------|
| <i>predecessor-VertexScale-And-Subsystem</i> | The scale and subsystem corresponding to the predecessor vertex |
| <i>newVertex-ScaleAnd-Subsystem</i>          | The scale and subsystem corresponding to the new vertex         |

Definition at line 377 of file MultiscaleArchitectureGraph.cpp.

References `ERR_INVALID_EDGE_NEW_VERTEX_EXISTS_BEGIN`, `ERR_INVALID_EDGE_NEW_VERTEX_EXISTS_END`, `ERR_INVALID_EDGE_NEW_VERTEX_EXISTS_MIDDLE1`, `ERR_INVALID_EDGE_NEW_VERTEX_EXISTS_MIDDLE2`, and `existsScaleAndSubsystem()`.

Referenced by `addVertexAndCorrespondingEdge()`.

```
6.108.3.20 void MultiscaleArchitectureGraph::checkIfPredecessorVertexExists (
    const std::string & predecessorVertexScaleAndSubsystem, const std::string &
    newVertexScaleAndSubsystem ) const [private]
```

Check if predecessor vertex belongs to graph and throw exception if not.

### Parameters

|                                              |                                                                 |
|----------------------------------------------|-----------------------------------------------------------------|
| <i>predecessor-VertexScale-And-Subsystem</i> | The scale and subsystem corresponding to the predecessor vertex |
| <i>newVertex-ScaleAnd-Subsystem</i>          | The scale and subsystem corresponding to the new vertex         |

Definition at line 360 of file MultiscaleArchitectureGraph.cpp.

References `ERR_INVALID_EDGE_PREDECESSOR_VERTEX_DOES_NOT_EXIST_BEGIN`, `ERR_INVALID_EDGE_PREDECESSOR_VERTEX_DOES_NOT_EXIST_END`, `ERR_INVALID_EDGE_PREDECESSOR_VERTEX_DOES_NOT_EXIST_MIDDLE1`, `ERR_INVALID_EDGE_PREDECESSOR_VERTEX_DOES_NOT_EXIST_MIDDLE2`, and `existsScaleAndSubsystem()`.

Referenced by `addVertexAndCorrespondingEdge()`.

```
6.108.3.21 void MultiscaleArchitectureGraph::checkIfValidIndegreeForNonRoot-
Vertices ( ) const [private]
```

Check if all vertices except the root have indegree 1 and throw exception if not.

Definition at line 236 of file MultiscaleArchitectureGraph.cpp.

Referenced by validate().

**6.108.3.22 long MultiscaleArchitectureGraph::computeRootVertexIndex ( ) const [private]**

Compute the index of the root vertex.

Precondition: The root vertex is unique.

Definition at line 505 of file MultiscaleArchitectureGraph.cpp.

References predecessorIndices, ROOT\_VERTEX\_PREDECESSOR\_INDEX, and vertices.

Referenced by checkIfExistsPathFromRootToOtherVertices().

**6.108.3.23 std::string MultiscaleArchitectureGraph::computeVertexName ( long vertexIndex ) const [private]**

Retrieve the name corresponding to the provided vertex index.

Warning: The complexity of this method is  $O(n)$  where  $n$  is the number of vertices. - Therefore it is inefficient and should be used rarely. If this method needs to be executed often then a different data structure should be employed for recording vertices names.

#### Parameters

|                    |                                       |
|--------------------|---------------------------------------|
| <i>vertexIndex</i> | The index corresponding to the vertex |
|--------------------|---------------------------------------|

Definition at line 519 of file MultiscaleArchitectureGraph.cpp.

References vertices.

Referenced by checkIfExistUnvisitedVertices().

**6.108.3.24 void MultiscaleArchitectureGraph::constructFromPropertyTree ( const pt::ptree & *propertyTree* ) [private]**

Construct the multiscale architecture graph from the provided property tree.

#### Parameters

|                     |                                                       |
|---------------------|-------------------------------------------------------|
| <i>propertyTree</i> | The property tree corresponding to the xml input file |
|---------------------|-------------------------------------------------------|

Definition at line 156 of file MultiscaleArchitectureGraph.cpp.

References addEdgesFromPropertyTree(), and addVerticesFromPropertyTree().

Referenced by readFromValidXmlFile().

## **6.108 multiscale::verification::MultiscaleArchitectureGraph Class Reference 647**

**6.108.3.25 bool MultiscaleArchitectureGraph::existsScaleAndSubsystem ( const std::string & *scaleAndSubsystem* ) const**

Check if the given scale and subsystem exists in the multiscale architecture graph.

Precondition: The provided scale and subsystem is in a valid format.

### Parameters

|                                |                                  |
|--------------------------------|----------------------------------|
| <i>scaleAnd-<br/>Subsystem</i> | The provided scale and subsystem |
|--------------------------------|----------------------------------|

Definition at line 93 of file MultiscaleArchitectureGraph.cpp.

References vertices.

Referenced by checkIfComparedScaleAndSubsystemsAreEncodedByGraph(), checkIfEdgeEndpointsAreEncodedByGraph(), checkIfNewVertexDoesNotExist(), checkIfPredecessorVertexExists(), and multiscale::verification::ScaleAndSubsystemEvaluator::validateScaleAndSubsystem().

**6.108.3.26 std::vector< std::pair< long, long > > MultiscaleArchitectureGraph::get-  
Edges ( ) const**

Return a copy of the edges where the edges' endpoints are specified by vertex indices.

Definition at line 75 of file MultiscaleArchitectureGraph.cpp.

References predecessorIndices, and vertices.

**6.108.3.27 std::vector< std::pair< std::string, long > > MultiscaleArchitectureGraph-  
::getVertices ( ) const**

Return a copy of the vertices scales and subsystems, and indices.

Definition at line 64 of file MultiscaleArchitectureGraph.cpp.

References vertices.

**6.108.3.28 bool MultiscaleArchitectureGraph::isEmpty ( ) const [private]**

Check if the multiscale architecture graph does not contain vertices and edges.

Definition at line 299 of file MultiscaleArchitectureGraph.cpp.

References predecessorIndices, successorsIndices, and vertices.

Referenced by addRootVertex().

---

```
6.108.3.29 bool MultiscaleArchitectureGraph::isScaleAndSubsystemSmallerThan
( const std::string & lhsScaleAndSubsystem, const std::string &
rhsScaleAndSubsystem ) const
```

Check if the left hand side scale and subsystem < right hand side scale and subsystem.

To determine the truth value the partial order imposed by the rooted directed tree representation is considered i.e.  $v_1 < v_2$  if and only if the path from the root to  $v_1$  passes through  $v_2$ .

**Parameters**

|                                         |                                         |
|-----------------------------------------|-----------------------------------------|
| <i>lhsScale-<br/>And-<br/>Subsystem</i> | The left hand side scale and subsystem  |
| <i>rhsScale-<br/>And-<br/>Subsystem</i> | The right hand side scale and subsystem |

Definition at line 100 of file MultiscaleArchitectureGraph.cpp.

References `checkIfComparedScaleAndSubsystemsAreEncodedByGraph()`, and `isValidScaleAndSubsystemSmallerThan()`.

Referenced by `multiscale::verification::ComparatorEvaluator::evaluate()`.

```
6.108.3.30 bool MultiscaleArchitectureGraph::isUniqueRootVertex( ) const
[private]
```

Check if the multiscale architecture graph contains a unique root vertex.

Definition at line 219 of file MultiscaleArchitectureGraph.cpp.

References `NO_PREDECESSOR_INDEX`, and `predecessorIndices`.

Referenced by `checkIfExistsUniqueRootVertex()`.

```
6.108.3.31 bool MultiscaleArchitectureGraph::isValidScaleAndSubsystem-
SmallerThan( const std::string & lhsScaleAndSubsystem, const std::string &
rhsScaleAndSubsystem ) const [private]
```

Check if the valid left hand side scale and subsystem < right hand side scale and subsystem.

Precondition: The multiscale architecture graph is a valid rooted directed tree.

**Parameters**

|                                         |                                        |
|-----------------------------------------|----------------------------------------|
| <i>lhsScale-<br/>And-<br/>Subsystem</i> | The left hand side scale and subsystem |
|-----------------------------------------|----------------------------------------|

## **6.108 multiscale::verification::MultiscaleArchitectureGraph Class Reference 649**

|                               |                                         |
|-------------------------------|-----------------------------------------|
| <i>rhsScale-And-Subsystem</i> | The right hand side scale and subsystem |
|-------------------------------|-----------------------------------------|

Definition at line 487 of file MultiscaleArchitectureGraph.cpp.

References predecessorIndices, ROOT\_VERTEX\_PREDECESSOR\_INDEX, and vertices.

Referenced by isScaleAndSubsystemSmallerThan().

### **6.108.3.32 void MultiscaleArchitectureGraph::read( ) [private]**

Read the multiscale architecture graph from the provided input file.

Definition at line 116 of file MultiscaleArchitectureGraph.cpp.

References ERR\_INVALID\_INPUT\_FILE\_PATH\_BEGIN, ERR\_INVALID\_INPUT\_FILE\_PATH\_END, inputFilePath, multiscale::Filesystem::isValidFilePath(), MULTISCALE\_ARCHITECTURE\_GRAPH\_INPUT\_FILE\_EXTENSION, and readFromValidFile().

Referenced by readFromFile().

### **6.108.3.33 void MultiscaleArchitectureGraph::readFromFile( const std::string &inputFilePath )**

Read the multiscale architecture graph from the provided input file.

Warning: The contents of the existing multiscale architecture graph will be overwritten by the contents stored in the provided input file.

#### **Parameters**

|                      |                            |
|----------------------|----------------------------|
| <i>inputFilePath</i> | The path to the input file |
|----------------------|----------------------------|

Definition at line 18 of file MultiscaleArchitectureGraph.cpp.

References inputFilePath, read(), reset(), and validate().

Referenced by multiscale::verification::ModelCheckingManager::initialiseMultiscaleArchitectureGraph().

### **6.108.3.34 void MultiscaleArchitectureGraph::readFromValidFile( ) [private]**

Read the multiscale architecture graph from the provided valid input file.

Precondition: The provided input file path points to a regular file with the correct extension.

Definition at line 130 of file MultiscaleArchitectureGraph.cpp.

References `ERR_INVALID_INPUT_FILE_RELATIVE_TO_XSD_BEGIN`, `ERR_INVALID_INPUT_FILE_RELATIVE_TO_XSD_END`, `inputFilePath`, `multiscale::XmlValidator::isValidXmlFile()`, `MULTISCALE_ARCHITECTURE_GRAPH_XSD_PATH`, and `readFromValidXmlFile()`.

Referenced by `read()`.

#### **6.108.3.35 void MultiscaleArchitectureGraph::readFromValidXmlFile ( ) [private]**

Read the multiscale architecture graph from the provided valid xml input file.

Precondition: The provided input file path points to a valid xml file (relative to xsd).

Definition at line 147 of file `MultiscaleArchitectureGraph.cpp`.

References `constructFromPropertyTree()`, and `inputFilePath`.

Referenced by `readFromFile()`.

#### **6.108.3.36 void MultiscaleArchitectureGraph::reset ( )**

Reset the multiscale architecture graph such that it does not contain any vertices or edges.

Definition at line 27 of file `MultiscaleArchitectureGraph.cpp`.

References `predecessorIndices`, `successorsIndices`, and `vertices`.

Referenced by `multiscaletest::TraceEvaluationTest::InitialiseMultiscaleArchitectureGraph()`, and `readFromFile()`.

#### **6.108.3.37 void MultiscaleArchitectureGraph::validate ( ) const [private]**

Check if the multiscale architecture graph is a rooted directed tree.

Definition at line 202 of file `MultiscaleArchitectureGraph.cpp`.

References `checkIfExistsPathFromRootToOtherVertices()`, `checkIfExistsUniqueRootVertex()`, and `checkIfValidIndegreeForNonRootVertices()`.

Referenced by `readFromFile()`.

### **6.108.4 Member Data Documentation**

#### **6.108.4.1 const std::string MultiscaleArchitectureGraph::ERR\_ADD\_ROOT\_VERTEX\_NON\_EMPTY\_GRAPH = "It is not possible to add a root vertex to a non-empty multiscale architecture graph. Please clear the contents of the graph before adding a root vertex to it." [static, private]**

Definition at line 338 of file `MultiscaleArchitectureGraph.hpp`.

Referenced by `addRootVertex()`.

```
6.108.4.2 const std::string MultiscaleArchitectureGraph::ERR_COMPARE_SCALE_AND_SUBSYSTEMS_NOT_ENCODED_BY_GRAPH_BEGIN = "The provided scales and subsystems \\" [static, private]
```

Definition at line 381 of file MultiscaleArchitectureGraph.hpp.

Referenced by checkIfComparedScaleAndSubsystemsAreEncodedByGraph().

```
6.108.4.3 const std::string MultiscaleArchitectureGraph::ERR_COMPARE_SCALE_AND_SUBSYSTEMS_NOT_ENCODED_BY_GRAPH_END = "\\" is not encoded by the multiscale architecture graph. Please change." [static, private]
```

Definition at line 384 of file MultiscaleArchitectureGraph.hpp.

Referenced by checkIfComparedScaleAndSubsystemsAreEncodedByGraph().

```
6.108.4.4 const std::string MultiscaleArchitectureGraph::ERR_COMPARE_SCALE_AND_SUBSYSTEMS_NOT_ENCODED_BY_GRAPH_MIDDLE1 = "\\" and \\" [static, private]
```

Definition at line 382 of file MultiscaleArchitectureGraph.hpp.

Referenced by checkIfComparedScaleAndSubsystemsAreEncodedByGraph().

```
6.108.4.5 const std::string MultiscaleArchitectureGraph::ERR_COMPARE_SCALE_AND_SUBSYSTEMS_NOT_ENCODED_BY_GRAPH_MIDDLE2 = "\\" cannot be compared because the scale and subsystem \\" [static, private]
```

Definition at line 383 of file MultiscaleArchitectureGraph.hpp.

Referenced by checkIfComparedScaleAndSubsystemsAreEncodedByGraph().

```
6.108.4.6 const std::string MultiscaleArchitectureGraph::ERR_INVALID_EDGE_BEGIN = "The edge (" [static, private]
```

Definition at line 348 of file MultiscaleArchitectureGraph.hpp.

```
6.108.4.7 const std::string MultiscaleArchitectureGraph::ERR_INVALID_EDGE_HEAD_ENDPOINT_INDEGREE_NOT_ZERO_BEGIN = ERR_INVALID_EDGE_BEGIN [static, private]
```

Definition at line 376 of file MultiscaleArchitectureGraph.hpp.

Referenced by checkIfEdgeHeadEndpointHasIndegreeZero().

```
6.108.4.8 const std::string MultiscaleArchitectureGraph::ERR_INVALID_EDGE_HE-
AD_ENDPOINT_INDEGREE_NOT_ZERO_END = "\\" has an indegree different
from zero. Please change." [static, private]
```

Definition at line 379 of file MultiscaleArchitectureGraph.hpp.

Referenced by checkIfEdgeHeadEndpointHasIndegreeZero().

```
6.108.4.9 const std::string MultiscaleArchitectureGraph::ERR_INVALID_E-
DGE_HEAD_ENDPOINT_INDEGREE_NOT_ZERO_MIDDLE1 =
ERR_INVALID_EDGE_MIDDLE1 [static, private]
```

Definition at line 377 of file MultiscaleArchitectureGraph.hpp.

Referenced by checkIfEdgeHeadEndpointHasIndegreeZero().

```
6.108.4.10 const std::string MultiscaleArchitectureGraph::ERR_INVALID_EDGE_-
HEAD_ENDPOINT_INDEGREE_NOT_ZERO_MIDDLE2 = "cannot be
added to the multiscale architecture graph because the edge head endpoint \\""
[static, private]
```

Definition at line 378 of file MultiscaleArchitectureGraph.hpp.

Referenced by checkIfEdgeHeadEndpointHasIndegreeZero().

```
6.108.4.11 const std::string MultiscaleArchitectureGraph::ERR_INVALID_ED-
GE_HEAD_ENDPOINT_NOT_ENCODED_BY_GRAPH_BEGIN =
ERR_INVALID_EDGE_BEGIN [static, private]
```

Definition at line 371 of file MultiscaleArchitectureGraph.hpp.

Referenced by checkIfEdgeEndpointsAreEncodedByGraph().

```
6.108.4.12 const std::string MultiscaleArchitectureGraph::ERR_INVALID_E-
DGE_HEAD_ENDPOINT_NOT_ENCODED_BY_GRAPH_END =
ERR_INVALID_EDGE_TAIL_ENDPOINT_NOT_ENCODED_BY_GRAPH-
-END [static, private]
```

Definition at line 374 of file MultiscaleArchitectureGraph.hpp.

Referenced by checkIfEdgeEndpointsAreEncodedByGraph().

```
6.108.4.13 const std::string MultiscaleArchitectureGraph::ERR_INVALID_EDG-
E_HEAD_ENDPOINT_NOT_ENCODED_BY_GRAPH_MIDDLE1 =
ERR_INVALID_EDGE_MIDDLE1 [static, private]
```

Definition at line 372 of file MultiscaleArchitectureGraph.hpp.

Referenced by checkIfEdgeEndpointsAreEncodedByGraph().

```
6.108.4.14 const std::string MultiscaleArchitectureGraph::ERR_INVALID_EDGE_HE-
AD_ENDPOINT_NOT_ENCODED_BY_GRAPH_MIDDLE2 = " cannot
be added to the multiscale architecture graph because the edge head endpoint "
[static, private]
```

Definition at line 373 of file MultiscaleArchitectureGraph.hpp.

Referenced by checkIfEdgeEndpointsAreEncodedByGraph().

```
6.108.4.15 const std::string MultiscaleArchitectureGraph::ERR_INVALID_EDGE-
_IDENTICAL_ENDPOINTS_BEGIN = ERR_INVALID_EDGE_BEGIN
[static, private]
```

Definition at line 362 of file MultiscaleArchitectureGraph.hpp.

Referenced by checkIfEdgeEndpointsAreDistinct().

```
6.108.4.16 const std::string MultiscaleArchitectureGraph::ERR_INVALID_EDGE_ID-
ENTICAL_ENDPOINTS_END = " cannot be added to the multiscale architecture
graph because the edge endpoints are identical and they should be distinct. Please
change." [static, private]
```

Definition at line 364 of file MultiscaleArchitectureGraph.hpp.

Referenced by checkIfEdgeEndpointsAreDistinct().

```
6.108.4.17 const std::string MultiscaleArchitectureGraph::ERR_INVALID_EDGE_ID-
ENTICAL_ENDPOINTS_MIDDLE = ERR_INVALID_EDGE_MIDDLE1
[static, private]
```

Definition at line 363 of file MultiscaleArchitectureGraph.hpp.

Referenced by checkIfEdgeEndpointsAreDistinct().

```
6.108.4.18 const std::string MultiscaleArchitectureGraph::ER-
R_INVALID_EDGE_MIDDLE1 = "," [static,
private]
```

Definition at line 349 of file MultiscaleArchitectureGraph.hpp.

```
6.108.4.19 const std::string MultiscaleArchitectureGraph::E-
RR_INVALID_EDGE_MIDDLE2 = ")" [static,
private]
```

Definition at line 350 of file MultiscaleArchitectureGraph.hpp.

```
6.108.4.20 const std::string MultiscaleArchitectureGraph::ERR_INVALID_EDGE_
    _NEW_VERTEX_EXISTS_BEGIN = ERR_INVALID_EDGE_BEGIN
    [static, private]
```

Definition at line 357 of file MultiscaleArchitectureGraph.hpp.

Referenced by checkIfNewVertexDoesNotExist().

```
6.108.4.21 const std::string MultiscaleArchitectureGraph::ERR_INVALID_EDGE_NE-
    W_VERTEX_EXISTS_END = "\\" is not a new but an existing vertex of the graph.
    Please change." [static, private]
```

Definition at line 360 of file MultiscaleArchitectureGraph.hpp.

Referenced by checkIfNewVertexDoesNotExist().

```
6.108.4.22 const std::string MultiscaleArchitectureGraph::ERR_INVALID_EDGE_N-
    EW_VERTEX_EXISTS_MIDDLE1 = ERR_INVALID_EDGE_MIDDLE1
    [static, private]
```

Definition at line 358 of file MultiscaleArchitectureGraph.hpp.

Referenced by checkIfNewVertexDoesNotExist().

```
6.108.4.23 const std::string MultiscaleArchitectureGraph::ERR_INVALID_EDGE_-
    NEW_VERTEX_EXISTS_MIDDLE2 = " cannot be added to the multiscale
    architecture graph because the edge head endpoint \"" [static,
    private]
```

Definition at line 359 of file MultiscaleArchitectureGraph.hpp.

Referenced by checkIfNewVertexDoesNotExist().

```
6.108.4.24 const std::string MultiscaleArchitectureGraph::ERR_INVALID_E-
    DGE_PREDECESSOR_VERTEX_DOES_NOT_EXIST_BEGIN =
    ERR_INVALID_EDGE_BEGIN [static, private]
```

Definition at line 352 of file MultiscaleArchitectureGraph.hpp.

Referenced by checkIfPredecessorVertexExists().

```
6.108.4.25 const std::string MultiscaleArchitectureGraph::ERR_INVALID_EDGE_PR-
    EDECESSOR_VERTEX_DOES_NOT_EXIST_END = "\\" is not a vertex of the
    graph. Please change." [static, private]
```

Definition at line 355 of file MultiscaleArchitectureGraph.hpp.

Referenced by checkIfPredecessorVertexExists().

```
6.108.4.26 const std::string MultiscaleArchitectureGraph::ERR_INVALID_EDGE_PREDECESSOR_VERTEX_DOES_NOT_EXIST_MIDDLE1 =  
ERR_INVALID_EDGE_MIDDLE1 [static, private]
```

Definition at line 353 of file MultiscaleArchitectureGraph.hpp.

Referenced by checkIfPredecessorVertexExists().

```
6.108.4.27 const std::string MultiscaleArchitectureGraph::ERR_INVALID_EDGE_PREDECESSOR_VERTEX_DOES_NOT_EXIST_MIDDLE2 = "cannot  
be added to the multiscale architecture graph because the edge tail endpoint \\""  
[static, private]
```

Definition at line 354 of file MultiscaleArchitectureGraph.hpp.

Referenced by checkIfPredecessorVertexExists().

```
6.108.4.28 const std::string MultiscaleArchitectureGraph::ERR_INVALID_EDGE_TAIL_ENDPOINT_NOT_ENCODED_BY_GRAPH_BEGIN =  
ERR_INVALID_EDGE_BEGIN [static, private]
```

Definition at line 366 of file MultiscaleArchitectureGraph.hpp.

Referenced by checkIfEdgeEndpointsAreEncodedByGraph().

```
6.108.4.29 const std::string MultiscaleArchitectureGraph::ERR_INVALID_EDGE_TAIL_ENDPOINT_NOT_ENCODED_BY_GRAPH_END = "\\" is not a vertex of  
the graph. Please change." [static, private]
```

Definition at line 369 of file MultiscaleArchitectureGraph.hpp.

Referenced by checkIfEdgeEndpointsAreEncodedByGraph().

```
6.108.4.30 const std::string MultiscaleArchitectureGraph::ERR_INVALID_EDGE_TAIL_ENDPOINT_NOT_ENCODED_BY_GRAPH_MIDDLE1 =  
ERR_INVALID_EDGE_MIDDLE1 [static, private]
```

Definition at line 367 of file MultiscaleArchitectureGraph.hpp.

Referenced by checkIfEdgeEndpointsAreEncodedByGraph().

```
6.108.4.31 const std::string MultiscaleArchitectureGraph::ERR_INVALID_EDGE_TAIL_ENDPOINT_NOT_ENCODED_BY_GRAPH_MIDDLE2 = "cannot  
be added to the multiscale architecture graph because the edge tail endpoint \\""  
[static, private]
```

Definition at line 368 of file MultiscaleArchitectureGraph.hpp.

Referenced by checkIfEdgeEndpointsAreEncodedByGraph().

```
6.108.4.32 const std::string MultiscaleArchitectureGraph::ERR_INVALID_INPU-
T_FILE_PATH_BEGIN = "The provided input file path (" [static,
private]
```

Definition at line 332 of file MultiscaleArchitectureGraph.hpp.

Referenced by read().

```
6.108.4.33 const std::string MultiscaleArchitectureGraph::ERR_INVALID-
_INPUT_FILE_PATH_END = "). Please change." [static,
private]
```

Definition at line 333 of file MultiscaleArchitectureGraph.hpp.

Referenced by read().

```
6.108.4.34 const std::string MultiscaleArchitectureGraph::ERR_INVALID_INPU-
T_FILE_RELATIVE_TO_XSD_BEGIN = "The provided xml input file (" [static,
private]
```

Definition at line 335 of file MultiscaleArchitectureGraph.hpp.

Referenced by readFromValidFile().

```
6.108.4.35 const std::string MultiscaleArchitectureGraph::ERR_INVALID-
_INPUT_FILE_RELATIVE_TO_XSD_END = "). " [static,
private]
```

Definition at line 336 of file MultiscaleArchitectureGraph.hpp.

Referenced by readFromValidFile().

```
6.108.4.36 const std::string MultiscaleArchitectureGraph::ERR_NO_PATH_FROM_-
ROOT_TO_VERTEX_BEGIN = "The multiscale architecture graph is invalid
because there exists no path from the root vertex " [static, private]
```

Definition at line 344 of file MultiscaleArchitectureGraph.hpp.

Referenced by checkIfExistUnvisitedVertices().

```
6.108.4.37 const std::string MultiscaleArchitectureGraph::ERR_NO_PATH_F-
ROM_ROOT_TO_VERTEX_END = ". Please change." [static,
private]
```

Definition at line 346 of file MultiscaleArchitectureGraph.hpp.

Referenced by checkIfExistUnvisitedVertices().

## **6.108 multiscale::verification::MultiscaleArchitectureGraph Class Reference 657**

**6.108.4.38 const std::string MultiscaleArchitectureGraph::ERR\_NO\_PATH\_FROM\_ROOT\_TO\_VERTEX\_MIDDLE = " to the vertex " [static, private]**

Definition at line 345 of file MultiscaleArchitectureGraph.hpp.

Referenced by checkIfExistUnvisitedVertices().

**6.108.4.39 const std::string MultiscaleArchitectureGraph::ERR\_NO\_ROOT\_VERTEX = "The multiscale architecture graph should but does not contain any root vertices. Please change." [static, private]**

Definition at line 342 of file MultiscaleArchitectureGraph.hpp.

**6.108.4.40 const std::string MultiscaleArchitectureGraph::ERR\_ROOT\_VERTEX\_NOT\_UNIQUE = "The multiscale architecture graph is invalid because it does not contain a unique root vertex. Please change." [static, private]**

Definition at line 340 of file MultiscaleArchitectureGraph.hpp.

Referenced by checkIfExistsUniqueRootVertex().

**6.108.4.41 std::string multiscale::verification::MultiscaleArchitectureGraph::inputFilePath [private]**

The path to the input file

Definition at line 26 of file MultiscaleArchitectureGraph.hpp.

Referenced by read(), readFromFile(), readFromValidFile(), and readFromValidXmlFile().

**6.108.4.42 const std::string MultiscaleArchitectureGraph::LABEL\_EDGE = "edge" [static, private]**

Definition at line 325 of file MultiscaleArchitectureGraph.hpp.

**6.108.4.43 const std::string MultiscaleArchitectureGraph::LABEL\_EDGE\_BEGIN\_VERTEX = "<xmllatr>.beginVertex" [static, private]**

Definition at line 326 of file MultiscaleArchitectureGraph.hpp.

Referenced by addEdgeFromPropertyTree().

```
6.108.4.44 const std::string MultiscaleArchitectureGraph::LABEL_E-
DGE_END_VERTEX = "<xmattr>.endVertex" [static,
private]
```

Definition at line 327 of file MultiscaleArchitectureGraph.hpp.

Referenced by addEdgeFromPropertyTree().

```
6.108.4.45 const std::string MultiscaleArchitectureGraph::LABEL_EDGES = ".edges"
[static, private]
```

Definition at line 320 of file MultiscaleArchitectureGraph.hpp.

Referenced by addEdgesFromPropertyTree().

```
6.108.4.46 const std::string MultiscaleArchitectureGraph::LABEL_MULTISCALE-
ARCHITECTURE_GRAPH = "multiscaleArchitectureGraph" [static,
private]
```

Definition at line 318 of file MultiscaleArchitectureGraph.hpp.

```
6.108.4.47 const std::string MultiscaleArchitectureGraph::LABEL_VERTEX = "vertex"
[static, private]
```

Definition at line 322 of file MultiscaleArchitectureGraph.hpp.

```
6.108.4.48 const std::string MultiscaleArchitectureGraph::LABEL_VERTEX_ID =
"<xmattr>.id" [static, private]
```

Definition at line 323 of file MultiscaleArchitectureGraph.hpp.

Referenced by addVertexFromPropertyTree().

```
6.108.4.49 const std::string MultiscaleArchitectureGraph::LABEL_VERTICES =
".vertices" [static, private]
```

Definition at line 319 of file MultiscaleArchitectureGraph.hpp.

Referenced by addVerticesFromPropertyTree().

```
6.108.4.50 const std::string MultiscaleArchitectureGraph::MULTISCALE_ARCHI-
TECTURE_GRAPH_INPUT_FILE_EXTENSION = ".xml" [static,
private]
```

Definition at line 329 of file MultiscaleArchitectureGraph.hpp.

Referenced by read().

```
6.108.4.51 const std::string MultiscaleArchitectureGraph::MUL-
TISCALE_ARCHITECTURE_GRAPH_XSD_PATH =
"/usr/local/share/mule/config/verification/spatial-temporal/schema/multiscale_-
architecture_graph.xsd" [static, private]
```

Definition at line 330 of file MultiscaleArchitectureGraph.hpp.

Referenced by readFromFile().

```
6.108.4.52 const long MultiscaleArchitectureGraph::NO_PREDECESSOR_INDEX =
-1 [static, private]
```

Definition at line 315 of file MultiscaleArchitectureGraph.hpp.

Referenced by addVertexByScaleAndSubsystem(), checkIfEdgeHeadEndpointHasIndegreeZero(), and isUniqueRootVertex().

```
6.108.4.53 std::vector<long> multiscale::verification::MultiscaleArchitectureGraph-
::predecessorIndices [private]
```

The collection storing the index of the predecessor for each vertex. The index of each vertex is stored in the associative container called "vertices". In case a vertex does not have a predecessor (i.e. it is the root) its predecessor index is set to NO\_PREDECES-SOR\_INDEX.

Definition at line 37 of file MultiscaleArchitectureGraph.hpp.

Referenced by addEdgeByEndpoints(), addRootVertexToEmptyMultiscaleArchitectureGraph(), addVertexAndCorrespondingValidEdge(), addVertexByScaleAndSubsystem(), checkIfEdgeHeadEndpointHasIndegreeZero(), computeRootVertexIndex(), getEdges(), isEmpty(), isUniqueRootVertex(), isValidScaleAndSubsystemSmallerThan(), and reset().

```
6.108.4.54 const long MultiscaleArchitectureGraph::ROOT_VERTEX_PRED-
CESSOR_INDEX = NO_PREDECESSOR_INDEX [static,
private]
```

Definition at line 316 of file MultiscaleArchitectureGraph.hpp.

Referenced by addRootVertexToEmptyMultiscaleArchitectureGraph(), computeRootVertexIndex(), and isValidScaleAndSubsystemSmallerThan().

```
6.108.4.55 std::vector<std::vector<long> > multiscale::verification-
::MultiscaleArchitectureGraph::successorsIndices
[private]
```

The collection storing the index of the successors for each vertex. The index of each vertex is stored in the associative container called "vertices".

Definition at line 43 of file MultiscaleArchitectureGraph.hpp.

---

Referenced by addEdgeByEndpoints(), addVertexAndCorrespondingValidEdge(), addVertexByScaleAndSubsystem(), checkIfExistsPathFromRootToOtherVertices(), isEmpty(), and reset().

```
6.108.4.56 std::unordered_map<std::string, long> multiscale-
    ::verification::MultiscaleArchitectureGraph::vertices
    [private]
```

The associative container storing the scale and subsystem of each vertex and its corresponding index (0-based indexing).

TODO: In case the data structure for this field is changed update the description of the predecessorIndices field accordingly.

Definition at line 29 of file MultiscaleArchitectureGraph.hpp.

Referenced by addEdgeByEndpoints(), addVertexAndCorrespondingValidEdge(), addVertexByScaleAndSubsystem(), addVerticesFromPropertyTree(), checkIfEdgeHeadEndpointHasInDegreeZero(), checkIfExistsPathFromRootToOtherVertices(), computeRootVertexIndex(), computeVertexName(), existsScaleAndSubsystem(), getEdges(), getVertices(), isEmpty(), isValidScaleAndSubsystemSmallerThan(), and reset().

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

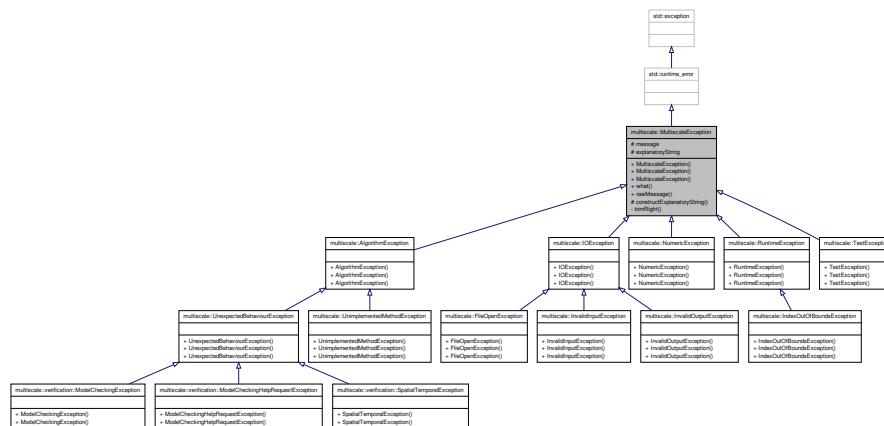
- MultiscaleArchitectureGraph.hpp
- MultiscaleArchitectureGraph.cpp

## 6.109 multiscale::MultiscaleException Class Reference

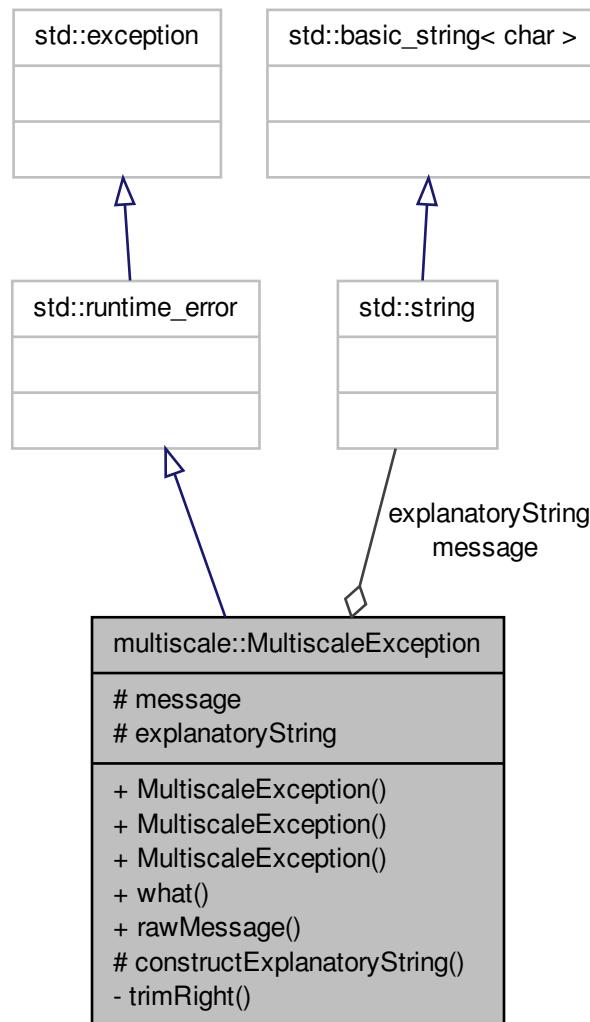
Parent exception class for the project.

```
#include <MultiscaleException.hpp>
```

Inheritance diagram for multiscale::MultiscaleException:



Collaboration diagram for multiscale::MultiscaleException:



## Public Member Functions

- `MultiscaleException ()`
- `MultiscaleException (const std::string &file, int line, const std::string &msg)`
- `MultiscaleException (const std::string &file, int line, const char *msg)`
- `const char * what () const noexcept override`

*Returns an explanatory string.*

- std::string `rawMessage () const noexcept`

*Return the raw message of the exception.*

### Protected Member Functions

- template<typename T >

void `constructExplanatoryString (const std::string &file, int line, T msg)`

*Construct the explanatory string.*

### Protected Attributes

- std::string `message`

- std::string `explanatoryString`

### Private Member Functions

- std::string `trimRight (const std::string &inputString)`

*Trim the right hand side of the provided string.*

#### 6.109.1 Detailed Description

Parent exception class for the project.

Definition at line 19 of file MultiscaleException.hpp.

#### 6.109.2 Constructor & Destructor Documentation

##### 6.109.2.1 multiscale::MultiscaleException::MultiscaleException( ) [inline]

Definition at line 28 of file MultiscaleException.hpp.

##### 6.109.2.2 multiscale::MultiscaleException::MultiscaleException ( const std::string & file, int line, const std::string & msg ) [inline, explicit]

Definition at line 30 of file MultiscaleException.hpp.

##### 6.109.2.3 multiscale::MultiscaleException::MultiscaleException ( const std::string & file, int line, const char \* msg ) [inline, explicit]

Definition at line 32 of file MultiscaleException.hpp.

### 6.109.3 Member Function Documentation

**6.109.3.1 template<typename T > void multiscale::MultiscaleException::constructExplanatoryString ( const std::string & *file*, int *line*, T *msg* ) [inline, protected]**

Construct the explanatory string.

#### Parameters

|             |                                      |
|-------------|--------------------------------------|
| <i>file</i> | File where the error occurred        |
| <i>line</i> | Line number where the error occurred |
| <i>msg</i>  | Error message                        |

Definition at line 54 of file MultiscaleException.hpp.

References explanatoryString, and trimRight().

**6.109.3.2 std::string multiscale::MultiscaleException::rawMessage ( ) const [inline]**

Return the raw message of the exception.

Definition at line 41 of file MultiscaleException.hpp.

References message.

Referenced by multiscale::OperatingSystem::executeProgram(), multiscale::verification::LogicPropertyVisitor::operator()(), multiscale::ExceptionHandler::printRawErrorMessage(), and multiscale::XmlValidator::verifyIfValidXmlFile().

**6.109.3.3 std::string multiscale::MultiscaleException::trimRight ( const std::string & *inputString* ) [inline, private]**

Trim the right hand side of the provided string.

#### Parameters

|                    |                           |
|--------------------|---------------------------|
| <i>inputString</i> | The provided input string |
|--------------------|---------------------------|

Definition at line 71 of file MultiscaleException.hpp.

Referenced by constructExplanatoryString().

**6.109.3.4 const char\* multiscale::MultiscaleException::what ( ) const [inline, override]**

Returns an explanatory string.

Definition at line 36 of file MultiscaleException.hpp.

References explanatoryString.

#### **6.109.4 Member Data Documentation**

6.109.4.1 std::string multiscale::MultiscaleException::explanatoryString  
[protected]

User friendly exception message

Definition at line 24 of file MultiscaleException.hpp.

Referenced by `constructExplanatoryString()`, and `what()`.

#### 6.109.4.2 std::string multiscale::MultiscaleException::message [protected]

## The raw message of the exception

Definition at line 23 of file MultiscaleException.hpp.

Referenced by rawMessage().

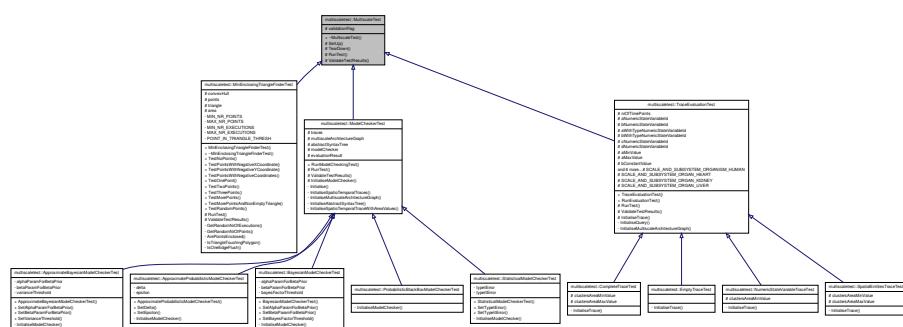
The documentation for this class was generated from the following file:

- MultiscaleException.hpp

## 6.110 multiscaletest::MultiscaleTest Class Reference

```
#include <MultiscaleTest.hpp>
```

## Inheritance diagram for multiscaletest::MultiscaleTest:



## Public Member Functions

- `virtual ~MultiscaleTest ()`

## Protected Member Functions

- virtual void [SetUp \(\)](#)
- virtual void [TearDown \(\)](#)
- virtual void [RunTest \(\)=0](#)

*Run the test.*
- virtual void [ValidateTestResults \(\)=0](#)

*Validate the results of the test.*

## Protected Attributes

- bool [validationFlag](#)

### 6.110.1 Detailed Description

Definition at line 9 of file MultiscaleTest.hpp.

### 6.110.2 Constructor & Destructor Documentation

#### 6.110.2.1 virtual multiscaletest::MultiscaleTest::~MultiscaleTest( ) [inline, virtual]

Definition at line 17 of file MultiscaleTest.hpp.

### 6.110.3 Member Function Documentation

#### 6.110.3.1 virtual void multiscaletest::MultiscaleTest::RunTest( ) [protected, pure virtual]

Run the test.

Implemented in [multiscaletest::TraceEvaluationTest](#), [multiscaletest::MinEnclosingTriangleFinderTest](#), and [multiscaletest::ModelCheckerTest](#).

#### 6.110.3.2 virtual void multiscaletest::MultiscaleTest::SetUp( ) [inline, protected, virtual]

Definition at line 21 of file MultiscaleTest.hpp.

#### 6.110.3.3 virtual void multiscaletest::MultiscaleTest::TearDown( ) [inline, protected, virtual]

Definition at line 22 of file MultiscaleTest.hpp.

6.110.3.4 **virtual void multiscaletest::MultiscaleTest::ValidateTestResults( )**  
[protected, pure virtual]

Validate the results of the test.

Implemented in [multiscaletest::TraceEvaluationTest](#), [multiscaletest::MinEnclosingTriangleFinderTest](#), and [multiscaletest::ModelCheckerTest](#).

#### 6.110.4 Member Data Documentation

6.110.4.1 **bool multiscaletest::MultiscaleTest::validationFlag** [protected]

Flag indicating if the test results are valid

Definition at line 13 of file MultiscaleTest.hpp.

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

- MultiscaleTest.hpp

### 6.111 multiscale::verification::NextKLogicPropertyAttribute Class Reference

Class for representing a "next K" logic property attribute.

```
#include <NextKLogicPropertyAttribute.hpp>
```

#### Public Attributes

- unsigned long [nrOfTimepointsAhead](#)
- [LogicPropertyAttributeType logicProperty](#)

#### 6.111.1 Detailed Description

Class for representing a "next K" logic property attribute.

Definition at line 14 of file NextKLogicPropertyAttribute.hpp.

#### 6.111.2 Member Data Documentation

6.111.2.1 **LogicPropertyAttributeType multiscale::verification::NextKLogicPropertyAttribute::logicProperty**

The logic property following the "next" operator

Definition at line 19 of file NextKLogicPropertyAttribute.hpp.

Referenced by multiscale::verification::LogicPropertyVisitor::evaluateNextKLogicProperty().

#### 6.111.2.2 **unsigned long multiscale::verification::NextKLogicPropertyAttribute::nrOfTimepointsAhead**

The number of timepoints ahead "K"

Definition at line 18 of file NextKLogicPropertyAttribute.hpp.

Referenced by multiscale::verification::LogicPropertyVisitor::evaluateNextKLogicProperty().

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

- [NextKLogicPropertyAttribute.hpp](#)

## 6.112 multiscale::verification::NextLogicPropertyAttribute Class - Reference

Class for representing a "next" logic property attribute.

```
#include <NextLogicPropertyAttribute.hpp>
```

### Public Attributes

- [LogicPropertyAttributeType logicProperty](#)

#### 6.112.1 Detailed Description

Class for representing a "next" logic property attribute.

Definition at line 14 of file NextLogicPropertyAttribute.hpp.

#### 6.112.2 Member Data Documentation

##### 6.112.2.1 **LogicPropertyAttributeType multiscale::verification::NextLogicPropertyAttribute::logicProperty**

The logic property attribute following the "next" operator

Definition at line 18 of file NextLogicPropertyAttribute.hpp.

Referenced by multiscale::verification::LogicPropertyVisitor::evaluateNextLogicProperty().

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

- [NextLogicPropertyAttribute.hpp](#)

## 6.113 multiscale::verification::Nil Class Reference

A class used to avoid run-time errors when defining a variant type.

```
#include <Nil.hpp>
```

### 6.113.1 Detailed Description

A class used to avoid run-time errors when defining a variant type.

When defining a variable of variant type "V" the default constructor of the first type within "V" is called. In order to avoid run-time errors this type needs to be different from the boost::recursive\_wrapper<T> type. In variants where all types are boost::recursive\_wrapper<T\_i> the **Nil** type can be added before them in order to avoid the potential run-time errors.

Definition at line 19 of file Nil.hpp.

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

- Nil.hpp

## 6.114 multiscale::verification::NotConstraintAttribute Class - Reference

Class for representing a "not" constraint attribute.

```
#include <NotConstraintAttribute.hpp>
```

### Public Attributes

- [ConstraintAttributeType constraint](#)

### 6.114.1 Detailed Description

Class for representing a "not" constraint attribute.

Definition at line 14 of file NotConstraintAttribute.hpp.

### 6.114.2 Member Data Documentation

#### 6.114.2.1 ConstraintAttributeType multiscale::verification::NotConstraintAttribute::constraint

The constraint which will be negated

Definition at line 18 of file NotConstraintAttribute.hpp.

Referenced by multiscale::verification::ConstraintVisitor::operator()().

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

- [NotConstraintAttribute.hpp](#)

## **6.115 multiscale::verification::NotLogicPropertyAttribute Class Reference**

Class for representing a "not" logic property attribute.

```
#include <NotLogicPropertyAttribute.hpp>
```

### **Public Attributes**

- [LogicPropertyAttributeType logicProperty](#)

#### **6.115.1 Detailed Description**

Class for representing a "not" logic property attribute.

Definition at line 14 of file NotLogicPropertyAttribute.hpp.

#### **6.115.2 Member Data Documentation**

##### **6.115.2.1 LogicPropertyAttributeType multiscale::verification::NotLogicPropertyAttribute::logicProperty**

The logic property following the "not" operator

Definition at line 18 of file NotLogicPropertyAttribute.hpp.

Referenced by multiscale::verification::LogicPropertyVisitor::operator()().

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

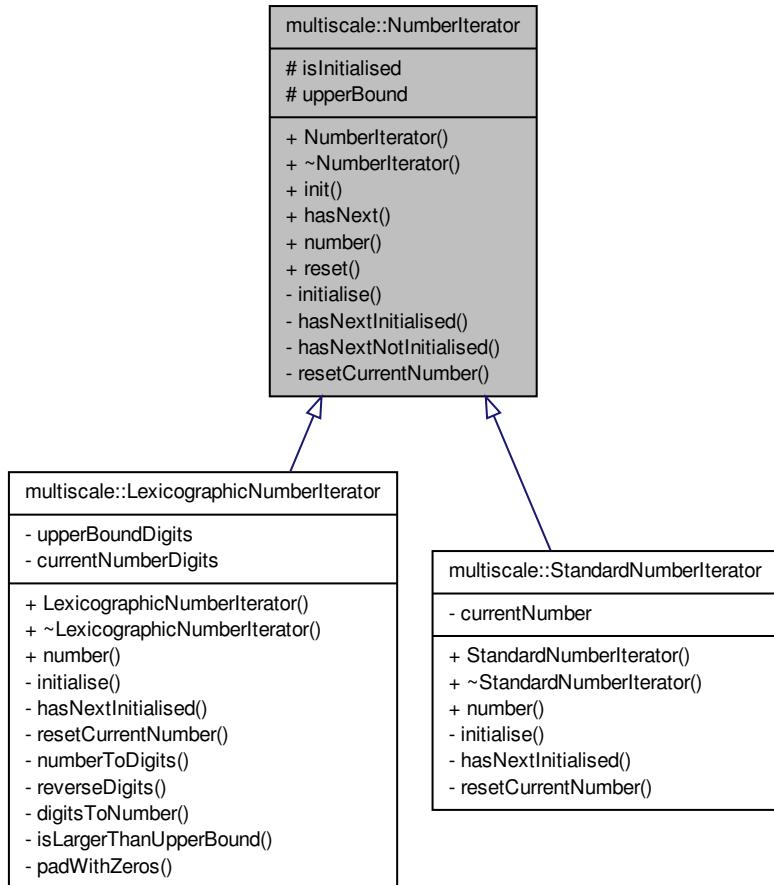
- [NotLogicPropertyAttribute.hpp](#)

## **6.116 multiscale::NumberIterator Class Reference**

Abstract class representing a number iterator.

```
#include <NumberIterator.hpp>
```

Inheritance diagram for multiscale::NumberIterator:



## Public Member Functions

- [NumberIterator](#) (`unsigned int upperBound`)
- virtual [~NumberIterator](#) ()
- void [init](#) (`unsigned int upperBound`)
 

*Initialise the iterator considering the given upper bound.*
- bool [hasNext](#) ()
 

*Check if there is a next number.*
- virtual `unsigned int number ()=0`

*Get the number pointed by the iterator.*
- void [reset](#) ()

*Reset the iterator.*

### Protected Attributes

- bool `isInitialised`
- unsigned int `upperBound`

### Private Member Functions

- virtual void `initialise ()=0`  
*Initialisation of the members of the class.*
- virtual bool `hasNextInitialised ()=0`  
*Check if there is a next number when in initialised state.*
- bool `hasNextNotInitialised ()`  
*Check if there is a next number when in not initialised state.*
- virtual void `resetCurrentNumber ()=0`  
*Reset the current number to its initial value.*

#### 6.116.1 Detailed Description

Abstract class representing a number iterator.

Definition at line 7 of file NumberIterator.hpp.

#### 6.116.2 Constructor & Destructor Documentation

##### 6.116.2.1 NumberIterator::NumberIterator ( unsigned int `upperBound` )

Definition at line 6 of file NumberIterator.cpp.

References `init()`.

##### 6.116.2.2 virtual multiscale::NumberIterator::~NumberIterator ( ) [inline, virtual]

Definition at line 17 of file NumberIterator.hpp.

#### 6.116.3 Member Function Documentation

##### 6.116.3.1 bool NumberIterator::hasNext ( )

Check if there is a next number.

Definition at line 14 of file NumberIterator.cpp.

References `hasNextInitialised()`, `hasNextNotInitialised()`, and `isInitialised`.

6.116.3.2 **virtual bool multiscale::NumberIterator::hasNextInitialised( )**  
[private, pure virtual]

Check if there is a next number when in initialised state.

Implemented in [multiscale::LexicographicNumberIterator](#), and [multiscale::StandardNumberIterator](#).

Referenced by [hasNext\(\)](#).

6.116.3.3 **bool NumberIterator::hasNextNotInitialised( )** [private]

Check if there is a next number when in not initialised state.

Definition at line 28 of file [NumberIterator.cpp](#).

References [isInitialised](#).

Referenced by [hasNext\(\)](#).

6.116.3.4 **void NumberIterator::init( unsigned int *upperBound* )**

Initialise the iterator considering the given upper bound.

**Parameters**

|                   |                 |
|-------------------|-----------------|
| <i>upperBound</i> | The upper bound |
|-------------------|-----------------|

Definition at line 10 of file [NumberIterator.cpp](#).

References [upperBound](#).

Referenced by [NumberIterator\(\)](#).

6.116.3.5 **virtual void multiscale::NumberIterator::initialise( )** [private, pure virtual]

Initialisation of the members of the class.

Implemented in [multiscale::LexicographicNumberIterator](#), and [multiscale::StandardNumberIterator](#).

6.116.3.6 **virtual unsigned int multiscale::NumberIterator::number( )** [pure virtual]

Get the number pointed by the iterator.

Implemented in [multiscale::LexicographicNumberIterator](#), and [multiscale::StandardNumberIterator](#).

**6.116.3.7 void NumberIterator::reset( )**

Reset the iterator.

Reset the iterator such that it is not initialised and the value of the current number is reset to its initial value

Definition at line 22 of file NumberIterator.cpp.

References isInitialised, and resetCurrentNumber().

Referenced by multiscale::LexicographicNumberIterator::LexicographicNumberIterator(), and multiscale::StandardNumberIterator::StandardNumberIterator().

**6.116.3.8 virtual void multiscale::NumberIterator::resetCurrentNumber( )  
[private, pure virtual]**

Reset the current number to its initial value.

Implemented in [multiscale::LexicographicNumberIterator](#), and [multiscale::StandardNumberIterator](#).

Referenced by reset().

## 6.116.4 Member Data Documentation

**6.116.4.1 bool multiscale::NumberIterator::isInitialised [protected]**

Flag for checking if the iterator was initialised

Definition at line 11 of file NumberIterator.hpp.

Referenced by hasNext(), hasNextNotInitialised(), and reset().

**6.116.4.2 unsigned int multiscale::NumberIterator::upperBound [protected]**

Upper bound of the iterator

Definition at line 12 of file NumberIterator.hpp.

Referenced by multiscale::StandardNumberIterator::hasNextInitialised(), init(), multiscale::LexicographicNumberIterator::initialise(), and multiscale::LexicographicNumberIterator::padWithZeros().

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

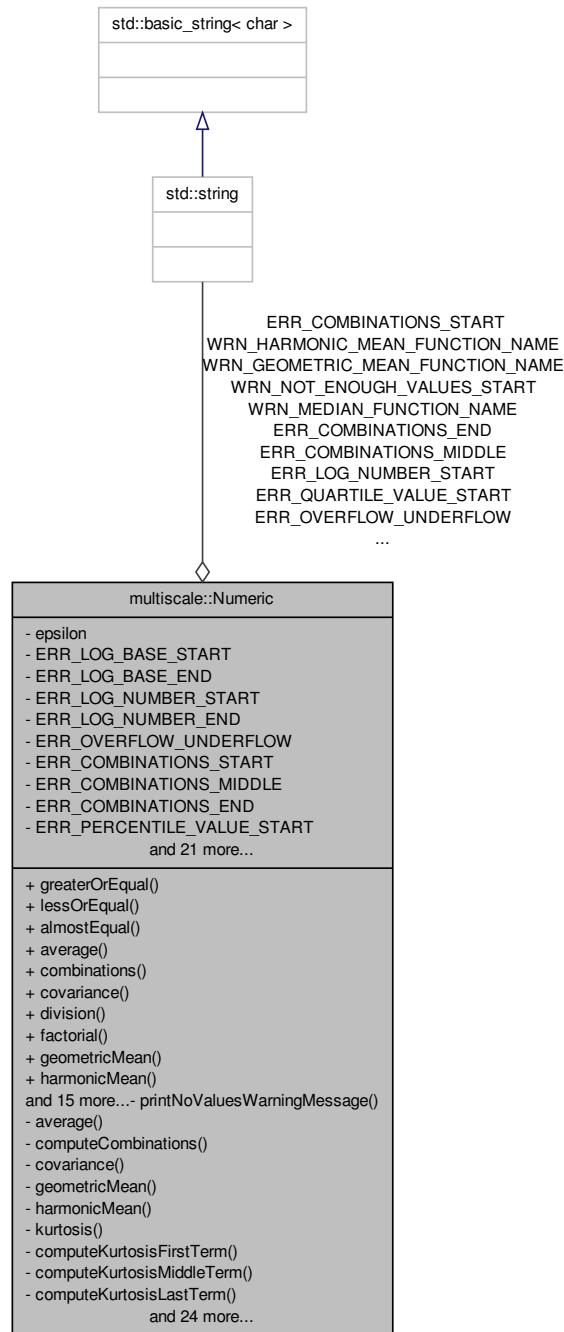
- NumberIterator.hpp
- NumberIterator.cpp

## 6.117 multiscale::Numeric Class Reference

Class for processing numeric (shorts, ints, floats, doubles etc.) expressions.

```
#include <Numeric.hpp>
```

Collaboration diagram for multiscale::Numeric:



## Static Public Member Functions

- static bool `greaterOrEqual` (double number1, double number2)  
*Check if the first number is greater than or equal to the second number.*
- static bool `lessOrEqual` (double number1, double number2)  
*Check if the first number is less than or equal to the second number.*
- static bool `almostEqual` (double number1, double number2)  
*Check if the two numbers are equal (almost)*
- static double `average` (const std::vector< double > &numbers)  
*Return the average (arithmetic mean) of the provided numbers.*
- static double `combinations` (unsigned int n, unsigned int k)  
*Return combinations of n taken as groups of k.*
- static double `covariance` (const std::vector< double > &values1, const std::vector< double > &values2)  
*Return the covariance for the provided collections of values.*
- static double `division` (double nominator, double denominator)  
*Return the result of dividing two numbers.*
- static unsigned long `factorial` (unsigned int number)  
*Return the factorial of a number.*
- static double `geometricMean` (const std::vector< double > &numbers)  
*Return the geometric mean of the provided numbers.*
- static double `harmonicMean` (const std::vector< double > &numbers)  
*Return the harmonic mean of the provided numbers.*
- static double `kurtosis` (const std::vector< double > &numbers)  
*Return the kurtosis of the provided numbers.*
- static double `log` (double number, double base)  
*Return the logarithm of a number considering the given base.*
- static double `maximum` (double number1, double number2, double number3)  
*Return the maximum of the provided numbers.*
- static double `maximum` (const std::vector< double > &numbers)  
*Return the maximum of the provided numbers.*
- static double `median` (const std::vector< double > &numbers)  
*Return the median of the provided numbers.*
- static double `mode` (const std::vector< double > &numbers)  
*Return the mode of the provided numbers.*
- static double `minimum` (const std::vector< double > &numbers)  
*Return the minimum of the provided numbers.*
- static double `percentile` (const std::vector< double > &numbers, double percentile)  
*Return the p-th percentile of the provided set of values.*
- static double `product` (const std::vector< double > &numbers)  
*Return the product of the provided numbers.*
- static double `quartile` (const std::vector< double > &numbers, double quartile)  
*Return the q-th quartile of the provided set of values.*

- template<typename FloatingPointType >  
 static FloatingPointType **round** (FloatingPointType number, unsigned int precision=2)  
*Round the number considering the given precision.*
- static double **skew** (const std::vector< double > &numbers)  
*Return the skew of the provided numbers.*
- static int **sign** (double number)  
*Return the sign of the number.*
- static double **standardDeviation** (const std::vector< double > &numbers)  
*Return the standard deviation of the provided set of values.*
- static double **sum** (const std::vector< double > &numbers)  
*Return the sum of the provided numbers.*
- static double **variance** (const std::vector< double > &numbers)  
*Return the variance of the provided set of values.*

### Static Private Member Functions

- static void **printNoValuesWarningMessage** (const std::string &functionName)  
*Print the no values warning message for the given function name.*
- static double **average** (const std::vector< double > &numbers, unsigned int nrOfValues)  
*Return the average (arithmetic mean) of the provided numbers.*
- static double **computeCombinations** (unsigned int n, unsigned int k)  
*Return combinations of n taken as groups of k.*
- static double **covariance** (const std::vector< double > &values1, const std::vector< double > &values2, unsigned int nrOfValues)  
*Return the covariance for the provided collections of values.*
- static double **geometricMean** (const std::vector< double > &numbers, unsigned int nrOfValues)  
*Return the geometric mean of the provided numbers.*
- static double **harmonicMean** (const std::vector< double > &numbers, unsigned int nrOfValues)  
*Return the harmonic mean of the provided numbers.*
- static double **kurtosis** (const std::vector< double > &numbers, unsigned int nrOfValues)  
*Return the kurtosis of the provided numbers.*
- static double **computeKurtosisFirstTerm** (unsigned int nrOfValues)  
*Compute the kurtosis first term considering the given number of values.*
- static double **computeKurtosisMiddleTerm** (const std::vector< double > &values, unsigned int nrOfValues)  
*Compute the kurtosis middle term considering the given values.*
- static double **computeKurtosisLastTerm** (unsigned int nrOfValues)  
*Compute the kurtosis last term considering the given number of values.*

- static double **maximum** (const std::vector< double > &numbers, unsigned int nrOfValues)  
*Return the maximum of the provided numbers.*
- static double **median** (const std::vector< double > &numbers, unsigned int nrOfValues)  
*Return the median of the provided numbers.*
- static double **mode** (const std::vector< double > &values, unsigned int nrOfValues)  
*Compute the mode for the provided values.*
- static double **computeMode** (const std::vector< double > &values, unsigned int nrOfValues)  
*Compute the mode for the provided values.*
- static double **minimum** (const std::vector< double > &numbers, unsigned int nrOfValues)  
*Return the minimum of the provided numbers.*
- static double **percentile** (const std::vector< double > &numbers, double percentile, unsigned int nrOfValues)  
*Return the p-th percentile of the provided set of values.*
- static double **product** (const std::vector< double > &numbers, unsigned int nrOfValues)  
*Return the product of the provided numbers.*
- static double **quartile** (const std::vector< double > &numbers, double quartile, unsigned int nrOfValues)  
*Return the q-th quartile of the provided set of values.*
- static double **computeQuartileValue** (double quartile, const std::vector< double > &values, unsigned int nrOfValues)  
*Compute the quartile for the given collection of values.*
- static double **skew** (const std::vector< double > &numbers, unsigned int nrOfValues)  
*Return the skew of the provided numbers.*
- static double **computeSkewFirstTerm** (unsigned int nrOfValues)  
*Return the skew first term considering the given values.*
- static double **computeSkewLastTerm** (const std::vector< double > &numbers, unsigned int nrOfValues)  
*Return the skew last term considering the given values.*
- static double **standardDeviation** (const std::vector< double > &numbers, unsigned int nrOfValues)  
*Return the standard deviation of the provided set of values.*
- static double **sum** (const std::vector< double > &numbers, unsigned int nrOfValues)  
*Return the sum of the provided numbers.*
- static double **variance** (const std::vector< double > &numbers, unsigned int nrOfValues)  
*Return the variance of the provided set of values.*

- template<typename Operation , typename Operand >  
 static Operand **applyOperation** (Operation operation, Operand operand1, -  
 Operand operand2)  
*Apply the operation on the given operands and throw an exception in case of overflow.*
- static void **resetOverflowUnderflowFlags** ()  
*Reset the overflow and underflow flags.*
- static bool **areOverflowUnderflowFlagsSet** ()  
*Reset the overflow and underflow flags.*
- static void **validateLogNumberAndBase** (double number, double base)  
*Check if the number and the base are positive real numbers, and if the base is different from 1.*
- static void **validateLogNumber** (double number)  
*Check if the number is a positive real number.*
- static void **validateLogBase** (double base)  
*Check if the base is a positive real number different from 1.*
- static void **validatePercentile** (double percentile)  
*Check if the value of the percentile is between 0 and 100.*
- static void **validateQuartile** (double quartile)  
*Check if the value of the quartile is either 25, 50 or 75.*
- template<typename T >  
 static bool **isPositive** (T number)  
*Check if the given number is positive.*
- template<typename T >  
 static T **numberInverse** (T number)  
*Return the inverse of a number.*

### Static Private Attributes

- static double **epsilon** = 1E-9
- static const std::string **ERR\_LOG\_BASE\_START** = "The base provided to the **log** function ("
- static const std::string **ERR\_LOG\_BASE\_END** = ") should be a positive real number different from 1. Please change."
- static const std::string **ERR\_LOG\_NUMBER\_START** = "The number provided to the **log** function ("
- static const std::string **ERR\_LOG\_NUMBER\_END** = ") should be a positive real number. Please change."
- static const std::string **ERR\_OVERFLOW\_UNDERFLOW** = "An underflow/overflow exception occurred."
- static const std::string **ERR\_COMBINATIONS\_START** = "The provided number of elements n ("
- static const std::string **ERR\_COMBINATIONS\_MIDDLE** = ") should be greater or equal to the number of elements in each group k ("
- static const std::string **ERR\_COMBINATIONS\_END** = ") when computing combinations."

- static const std::string `ERR_PERCENTILE_VALUE_START` = "The provided `percentile` value ("
- static const std::string `ERR_PERCENTILE_VALUE_END` = ") should be between 0 and 100. Please change."
- static const std::string `ERR_QUARTILE_VALUE_START` = "The provided `quartile` value ("
- static const std::string `ERR_QUARTILE_VALUE_END` = ") should be 25, 50 or 75. Please change."
- static const std::string `WRN_NUMBER_INVERSE` = "You provided the invalid value \"0\" to the `Numeric::inverse(...)` function. The default value \"0\" was returned."
- static const std::string `WRN_NOT_ENOUGH_VALUES_START` = "You provided less than the `minimum` required number of values to the `Numeric::`"
- static const std::string `WRN_NOT_ENOUGH_VALUES_END` = "(...) function. - The default value \"0\" was returned."
- static const std::string `WRN_AVERAGE_FUNCTION_NAME` = "average"
- static const std::string `WRN_COVARIANCE_FUNCTION_NAME` = "covariance"
- static const std::string `WRN_GEOMETRIC_MEAN_FUNCTION_NAME` = "geometricMean"
- static const std::string `WRN_HARMONIC_MEAN_FUNCTION_NAME` = "harmonicMean"
- static const std::string `WRN_KURTOSIS_FUNCTION_NAME` = "kurtosis"
- static const std::string `WRN_MAXIMUM_FUNCTION_NAME` = "maximum"
- static const std::string `WRN_MEDIAN_FUNCTION_NAME` = "median"
- static const std::string `WRN_MODE_FUNCTION_NAME` = "mode"
- static const std::string `WRN_MINIMUM_FUNCTION_NAME` = "minimum"
- static const std::string `WRN_PERCENTILE_FUNCTION_NAME` = "percentile"
- static const std::string `WRN_PRODUCT_FUNCTION_NAME` = "product"
- static const std::string `WRN_QUARTILE_FUNCTION_NAME` = "quartile"
- static const std::string `WRN_SKEW_FUNCTION_NAME` = "skew"
- static const std::string `WRN_STANDARD_DEVIATION_FUNCTION_NAME` = "standardDeviation"
- static const std::string `WRN_SUM_FUNCTION_NAME` = "sum"
- static const std::string `WRN_VARIANCE_FUNCTION_NAME` = "variance"

### 6.117.1 Detailed Description

Class for processing numeric (shorts, ints, floats, doubles etc.) expressions.

Definition at line 16 of file Numeric.hpp.

### 6.117.2 Member Function Documentation

#### 6.117.2.1 `bool Numeric::almostEqual ( double number1, double number2 ) [static]`

Check if the two numbers are equal (almost)

The expression for determining if two real numbers are equal is: if (`Abs(x - y) <= EPSILON * Max(1.0f, Abs(x), Abs(y))`).

**Parameters**

|                |               |
|----------------|---------------|
| <i>number1</i> | First number  |
| <i>number2</i> | Second number |

Definition at line 30 of file Numeric.cpp.

References epsilon, and maximum().

Referenced by multiscale::Geometry2D::areCollinear(), multiscale::Geometry2D::areEqualPoints(), multiscale::Geometry2D::areIdenticalLines(), multiscale::verification::BayesianModelChecker::computeBayesFactorValue(), multiscale::verification::StatisticalModelChecker::computeFPrimeValueFirstTerm(), multiscale::verification::StatisticalModelChecker::computeFPrimeValueSecondTerm(), multiscale::verification::StatisticalModelChecker::computeFValueFirstTerm(), multiscale::verification::StatisticalModelChecker::computeFValueSecondTerm(), computeMode(), computeQuartileValue(), division(), multiscale::verification::ComparatorEvaluator::evaluate(), greaterOrEqual(), multiscale::verification::TimeseriesComponentEvaluator::UniformHomogeneousComponentEvaluator< Relation >::hasValidSuccessors(), multiscale::verification::BayesianModelChecker::initialise(), multiscale::Geometry2D::isPointOnLineSegment(), lessOrEqual(), multiscale::Geometry2D::lineCircleIntersection(), multiscale::Geometry2D::lineIntersection(), numberInverse(), multiscale::DivisionOperation::operator()(), multiscale::verification::SpatialEntity::operator==(), multiscale::Geometry2D::slopeOfLine(), validateLogBase(), and validateQuartile().

**6.117.2.2 template<typename Operation , typename Operand > static Operand  
multiscale::Numeric::applyOperation ( Operation *operation*, Operand  
*operand1*, Operand *operand2* ) [inline, static, private]**

Apply the operation on the given operands and throw an exception in case of overflow.

**Parameters**

|                  |                    |
|------------------|--------------------|
| <i>operation</i> | The operation      |
| <i>operand1</i>  | The first operand  |
| <i>operand2</i>  | The second operand |

Definition at line 416 of file Numeric.hpp.

References areOverflowUnderflowFlagsSet(), ERR\_OVERFLOW\_UNDERFLOW, and resetOverflowUnderflowFlags().

Referenced by average(), computeKurtosisMiddleTerm(), computeSkewLastTerm(), covariance(), factorial(), geometricMean(), harmonicMean(), product(), standardDeviation(), sum(), and variance().

**6.117.2.3 bool Numeric::areOverflowUnderflowFlagsSet ( ) [static,  
private]**

Reset the overflow and underflow flags.

Definition at line 561 of file Numeric.cpp.

Referenced by applyOperation().

**6.117.2.4 double Numeric::average ( const std::vector< double > & numbers ) [static]**

Return the average (arithmetic mean) of the provided numbers.

$$\text{average} = \frac{1}{n} \sum_{i=1}^n x_i$$

**Parameters**

|                |                           |
|----------------|---------------------------|
| <i>numbers</i> | The collection of numbers |
|----------------|---------------------------|

Definition at line 37 of file Numeric.cpp.

References printNoValuesWarningMessage(), and WRN\_AVERAGE\_FUNCTION\_NAME.

Referenced by multiscale::analysis::RegionDetector::computeAverageIntensity(), computeKurtosisMiddleTerm(), computeSkewLastTerm(), covariance(), multiscale::verification::NumericEvaluator::evaluate(), standardDeviation(), and variance().

**6.117.2.5 double Numeric::average ( const std::vector< double > & numbers, unsigned int nrOfValues ) [static, private]**

Return the average (arithmetic mean) of the provided numbers.

$$\text{average} = \frac{1}{n} \sum_{i=1}^n x_i$$

**Parameters**

|                   |                                                   |
|-------------------|---------------------------------------------------|
| <i>numbers</i>    | The collection of numbers                         |
| <i>nrOfValues</i> | The number of values in the collection of numbers |

Definition at line 278 of file Numeric.cpp.

References applyOperation(), division(), and sum().

**6.117.2.6 double Numeric::combinations ( unsigned int n, unsigned int k ) [static]**

Return combinations of n taken as groups of k.

**Parameters**

|          |                                            |
|----------|--------------------------------------------|
| <i>n</i> | The total number of elements               |
| <i>k</i> | The number of elements in each combination |

Definition at line 49 of file Numeric.cpp.

References computeCombinations(), ERR\_COMBINATIONS\_END, ERR\_COMBINATIONS\_MIDDLE, ERR\_COMBINATIONS\_START, and multiscale::StringManipulator::toString().

**6.117.2.7 double Numeric::computeCombinations ( unsigned int *n*, unsigned int *k* )**  
[static, private]

Return combinations of *n* taken as groups of *k*.

**Parameters**

|          |                                            |
|----------|--------------------------------------------|
| <i>n</i> | The total number of elements               |
| <i>k</i> | The number of elements in each combination |

Definition at line 288 of file Numeric.cpp.

Referenced by combinations().

**6.117.2.8 double Numeric::computeKurtosisFirstTerm ( unsigned int *nrOfValues* )**  
[static, private]

Compute the kurtosis first term considering the given number of values.

**Parameters**

|                   |                      |
|-------------------|----------------------|
| <i>nrOfValues</i> | The number of values |
|-------------------|----------------------|

Definition at line 346 of file Numeric.cpp.

Referenced by kurtosis().

**6.117.2.9 double Numeric::computeKurtosisLastTerm ( unsigned int *nrOfValues* )**  
[static, private]

Compute the kurtosis last term considering the given number of values.

**Parameters**

|                   |                      |
|-------------------|----------------------|
| <i>nrOfValues</i> | The number of values |
|-------------------|----------------------|

Definition at line 368 of file Numeric.cpp.

Referenced by kurtosis().

**6.117.2.10 double Numeric::computeKurtosisMiddleTerm ( const std::vector< double > & *values*, unsigned int *nrOfValues* )** [static, private]

Compute the kurtosis middle term considering the given values.

**Parameters**

|                   |                      |
|-------------------|----------------------|
| <i>values</i>     | The values           |
| <i>nrOfValues</i> | The number of values |

Definition at line 354 of file Numeric.cpp.

References applyOperation(), average(), division(), and standardDeviation().

Referenced by kurtosis().

```
6.117.2.11 double Numeric::computeMode ( const std::vector< double > & values,  
                                     unsigned int nrOfValues ) [static, private]
```

Compute the mode for the provided values.

**Parameters**

|                   |                      |
|-------------------|----------------------|
| <i>values</i>     | The values           |
| <i>nrOfValues</i> | The number of values |

Definition at line 421 of file Numeric.cpp.

References almostEqual().

Referenced by mode().

```
6.117.2.12 double Numeric::computeQuartileValue ( double quartile, const std::vector< double > & values, unsigned int nrOfValues ) [static, private]
```

Compute the quartile for the given collection of values.

**Parameters**

|                   |                                        |
|-------------------|----------------------------------------|
| <i>quartile</i>   | The quartile                           |
| <i>values</i>     | The collection of values               |
| <i>nrOfValues</i> | The number of values in the collection |

Definition at line 479 of file Numeric.cpp.

References almostEqual().

Referenced by quartile().

```
6.117.2.13 double Numeric::computeSkewFirstTerm ( unsigned int nrOfValues )  
[static, private]
```

Return the skew first term considering the given values.

**Parameters**

|                   |                      |
|-------------------|----------------------|
| <i>nrOfValues</i> | The number of values |
|-------------------|----------------------|

Definition at line 500 of file Numeric.cpp.

Referenced by skew().

**6.117.2.14 double Numeric::computeSkewLastTerm ( const std::vector< double > & numbers, unsigned int nrOfValues ) [static, private]**

Return the skew last term considering the given values.

**Parameters**

|                   |                          |
|-------------------|--------------------------|
| <i>numbers</i>    | The collection of values |
| <i>nrOfValues</i> | The number of values     |

Definition at line 507 of file Numeric.cpp.

References applyOperation(), average(), division(), and standardDeviation().

Referenced by skew().

**6.117.2.15 double Numeric::covariance ( const std::vector< double > & values1, const std::vector< double > & values2 ) [static]**

Return the covariance for the provided collections of values.

**Parameters**

|                |                                 |
|----------------|---------------------------------|
| <i>values1</i> | The first collection of values  |
| <i>values2</i> | The second collection of values |

Definition at line 62 of file Numeric.cpp.

References printNoValuesWarningMessage(), and WRN\_COVARIANCE\_FUNCTION\_NAME.

Referenced by covariance(), and multiscale::verification::NumericEvaluator::evaluate().

**6.117.2.16 double Numeric::covariance ( const std::vector< double > & values1, const std::vector< double > & values2, unsigned int nrOfValues ) [static, private]**

Return the covariance for the provided collections of values.

**Parameters**

|                   |                                                   |
|-------------------|---------------------------------------------------|
| <i>values1</i>    | The first collection of values                    |
| <i>values2</i>    | The second collection of values                   |
| <i>nrOfValues</i> | The number of values in the collection of numbers |

Definition at line 299 of file Numeric.cpp.

References applyOperation(), average(), covariance(), and division().

**6.117.2.17 double Numeric::division ( double *nominator*, double *denominator* )  
[static]**

Return the result of dividing two numbers.

If the denominator is almost equal to zero then return zero. Otherwise return the result of dividing the nominator to the denominator.

#### Parameters

|                    |                 |
|--------------------|-----------------|
| <i>nominator</i>   | The nominator   |
| <i>denominator</i> | The denominator |

Definition at line 74 of file Numeric.cpp.

References almostEqual().

Referenced by average(), multiscale::analysis::ClusterDetector::ClusterDetector(), multiscale::analysis::CircularityMeasure< PointType >::compute(), multiscale::analysis::RegionDetector::computeAverageClusterednessDegree(), multiscale::analysis::RegionDetector::computeAverageDensity(), multiscale::analysis::Silhouette::computeAverageDissimilarityBtwEntityAndCluster(), multiscale::analysis::Silhouette::computeAverageDissimilarityWithinCluster(), multiscale::analysis::Silhouette::computeAverageMeasure(), multiscale::analysis::ClusterDetector::computeAveragePileUpDegree(), multiscale::analysis::Region::computeClusterednessDegreeIfOuterBorderDefined(), computeKurtosisMiddleTerm(), multiscale::analysis::Silhouette::computeMeasure(), multiscale::visualisation::RectangularCsvToInputFilesConverter::computeNextPositionConcentration(), multiscale::visualisation::PolarCsvToInputFilesConverter::computeNextPositionConcentration(), multiscale::verification::ChangeMeasureEvaluator::computeNumericMeasureValueChange(), multiscale::analysis::Silhouette::computeOverallAverageMeasure(), multiscale::analysis::SimulationClusterDetector::computePileUpDegreeAtPosition(), multiscale::analysis::RegionDetector::computeRegionDensity(), computeSkewLastTerm(), multiscale::NumericRangeManipulator::convertFromRange(), covariance(), multiscale::Geometry2D::distanceFromPointToLine(), multiscale::verification::ChangeMeasureEvaluator::evaluate(), multiscale::verification::NumericEvaluator::evaluate(), harmonicMean(), multiscale::analysis::SimulationClusterDetector::initialiseDetectorSpecificImageDependentFields(), multiscale::Geometry2D::lineCircleOneIntersectionPoint(), multiscale::Geometry2D::lineCircleTwoIntersectionPoints(), multiscale::analysis::SpatialEntityPseudo3D::normalisedShapeMeasure(), multiscale::Geometry2D::orthogonalLineToAnotherLineEdgePoints(), multiscale::analysis::Cluster::updateClusterednessDegree(), multiscale::analysis::Cluster::updateDensity(), multiscale::verification::ApproximateBayesianModelChecker::updateMean(), and multiscale::verification::ApproximateBayesianModelChecker::updateVariance().

6.117.2.18 **unsigned long Numeric::factorial ( unsigned int *number* ) [static]**

Return the factorial of a number.

**Parameters**

|               |                                                   |
|---------------|---------------------------------------------------|
| <i>number</i> | The number for which factorial should be computed |
|---------------|---------------------------------------------------|

Definition at line 82 of file Numeric.cpp.

References applyOperation().

6.117.2.19 **double Numeric::geometricMean ( const std::vector< double > & *numbers* ) [static]**

Return the geometric mean of the provided numbers.

$$\text{geometricMean} = e^{\frac{1}{n} \sum_{i=1}^n \log(x_i)}$$

**Parameters**

|                |                           |
|----------------|---------------------------|
| <i>numbers</i> | The collection of numbers |
|----------------|---------------------------|

Definition at line 92 of file Numeric.cpp.

References printNoValuesWarningMessage(), and WRN\_GEOMETRIC\_MEAN\_FUNCTION\_NAME.

Referenced by multiscale::verification::NumericEvaluator::evaluate().

6.117.2.20 **double Numeric::geometricMean ( const std::vector< double > & *numbers*, unsigned int *nrOfValues* ) [static, private]**

Return the geometric mean of the provided numbers.

$$\text{geometricMean} = e^{\frac{1}{n} \sum_{i=1}^n \log(x_i)}$$

**Parameters**

|                   |                                                   |
|-------------------|---------------------------------------------------|
| <i>numbers</i>    | The collection of numbers                         |
| <i>nrOfValues</i> | The number of values in the collection of numbers |

Definition at line 313 of file Numeric.cpp.

References applyOperation(), and validateLogNumber().

6.117.2.21 **bool Numeric::greaterOrEqual ( double *number1*, double *number2* ) [static]**

Check if the first number is greater than or equal to the second number.

**Parameters**

|                |                   |
|----------------|-------------------|
| <i>number1</i> | The first number  |
| <i>number2</i> | The second number |

Definition at line 16 of file Numeric.cpp.

References `almostEqual()`.

Referenced by `multiscale::MinEnclosingTriangleFinder::advanceBToRightChain()`, `multiscale::analysis::Entity::isValid()`, `multiscale::analysis::Region::isValidInputValues()`, `multiscale::analysis::Cluster::isValidOriginDependentValues()`, `multiscale::verification::ApproximateProbabilisticModelChecker::doesPropertyHoldConsideringProbabilityComparator()`, `multiscale::verification::ComparatorEvaluator::evaluate()`, `multiscale::verification::ApproximateBayesianModelChecker::isModelCheckingResultTrueConsideringComparator()`, `multiscale::analysis::SpatialEntityPseudo3D::normalisedShapeMeasure()`, and `multiscale::MinEnclosingTriangleFinder::searchForBTangency()`.

**6.117.2.22 double Numeric::harmonicMean ( const std::vector< double > & *numbers* ) [static]**

Return the harmonic mean of the provided numbers.

$$\text{harmonicMean} = \frac{n}{\sum_{i=1}^n \frac{1}{x_i}}$$

**Parameters**

|                |                           |
|----------------|---------------------------|
| <i>numbers</i> | The collection of numbers |
|----------------|---------------------------|

Definition at line 104 of file Numeric.cpp.

References `printNoValuesWarningMessage()`, and `WRN_HARMONIC_MEAN_FUNCTION_NAME`.

Referenced by `multiscale::verification::NumericEvaluator::evaluate()`.

**6.117.2.23 double Numeric::harmonicMean ( const std::vector< double > & *numbers*, unsigned int *nrOfValues* ) [static, private]**

Return the harmonic mean of the provided numbers.

$$\text{harmonicMean} = \frac{n}{\sum_{i=1}^n \frac{1}{x_i}}$$

**Parameters**

|                   |                                                   |
|-------------------|---------------------------------------------------|
| <i>numbers</i>    | The collection of numbers                         |
| <i>nrOfValues</i> | The number of values in the collection of numbers |

Definition at line 326 of file Numeric.cpp.

References applyOperation(), division(), and numberInverse().

**6.117.2.24 template<typename T > static bool multiscale::Numeric::isPositive ( T number ) [inline, static, private]**

Check if the given number is positive.

**Parameters**

|               |                  |
|---------------|------------------|
| <i>number</i> | The given number |
|---------------|------------------|

Definition at line 470 of file Numeric.hpp.

Referenced by validateLogBase(), and validateLogNumber().

**6.117.2.25 double Numeric::kurtosis ( const std::vector< double > & numbers ) [static]**

Return the kurtosis of the provided numbers.

$$\text{kurtosis} = \frac{n(n+1)}{(n-1)(n-2)(n-3)} \left( \sum_{i=1}^n \left( \frac{x_i - \text{mean}}{\text{stdev}} \right)^4 \right) - \frac{3(n-1)^2}{(n-2)(n-3)}$$

**Parameters**

|                |                           |
|----------------|---------------------------|
| <i>numbers</i> | The collection of numbers |
|----------------|---------------------------|

Definition at line 116 of file Numeric.cpp.

References printNoValuesWarningMessage(), and WRN\_KURTOSIS\_FUNCTION\_NAME.

Referenced by multiscale::verification::NumericEvaluator::evaluate().

**6.117.2.26 double Numeric::kurtosis ( const std::vector< double > & numbers, unsigned int nrOfValues ) [static, private]**

Return the kurtosis of the provided numbers.

$$\text{kurtosis} = \frac{n(n+1)}{(n-1)(n-2)(n-3)} \left( \sum_{i=1}^n \left( \frac{x_i - \text{mean}}{\text{stdev}} \right)^4 \right) - \frac{3(n-1)^2}{(n-2)(n-3)}$$

**Parameters**

|                   |                                                   |
|-------------------|---------------------------------------------------|
| <i>numbers</i>    | The collection of numbers                         |
| <i>nrOfValues</i> | The number of values in the collection of numbers |

Definition at line 338 of file Numeric.cpp.

References computeKurtosisFirstTerm(), computeKurtosisLastTerm(), and computeKurtosisMiddleTerm().

---

**6.117.2.27 bool Numeric::lessOrEqual ( double *number1*, double *number2* ) [static]**

Check if the first number is less than or equal to the second number.

**Parameters**

|                |                   |
|----------------|-------------------|
| <i>number1</i> | The first number  |
| <i>number2</i> | The second number |

Definition at line 23 of file Numeric.cpp.

References almostEqual().

Referenced by multiscale::verification::LogicPropertyVisitor::areSimilarValues(), multiscale::analysis::Entity::isValid(), multiscale::analysis::Region::isValidInputValues(), multiscale::analysis::Cluster::isValidOriginDependentValues(), multiscale::verification::ApproximateProbabilisticModelChecker::doesPropertyHoldConsideringProbabilityComparator(), multiscale::verification::ComparatorEvaluator::evaluate(), multiscale::Geometry2D::isAngleBetweenNonReflex(), multiscale::verification::ApproximateBayesianModelChecker::isModelCheckingResultTrueConsideringComparator(), multiscale::analysis::SpatialEntityPseudo3D::normalisedShapeMeasure(), multiscale::verification::BayesianModelChecker::validateBayesFactorThreshold(), and multiscale::verification::ApproximateBayesianModelChecker::validateVarianceThreshold().

**6.117.2.28 double Numeric::log ( double *number*, double *base* ) [static]**

Return the logarithm of a number considering the given base.

The conditions imposed on the number and base are:

- *number*: a positive real number
- *base*: a positive real number different from 1

**Parameters**

|               |                       |
|---------------|-----------------------|
| <i>number</i> | The considered number |
| <i>base</i>   | The considered base   |

Definition at line 128 of file Numeric.cpp.

References validateLogNumberAndBase().

Referenced by multiscale::verification::NumericEvaluator::evaluate().

**6.117.2.29 double Numeric::maximum ( double *number1*, double *number2*, double *number3* ) [static]**

Return the maximum of the provided numbers.

**Parameters**

|                |                   |
|----------------|-------------------|
| <i>number1</i> | The first number  |
| <i>number2</i> | The second number |
| <i>number3</i> | The third number  |

Definition at line 134 of file Numeric.cpp.

Referenced by almostEqual(), multiscale::verification::NumericEvaluator::evaluate(), and maximum().

**6.117.2.30 double Numeric::maximum ( const std::vector< double > & *numbers* )  
[static]**

Return the maximum of the provided numbers.

**Parameters**

|                |                           |
|----------------|---------------------------|
| <i>numbers</i> | The collection of numbers |
|----------------|---------------------------|

Definition at line 138 of file Numeric.cpp.

References maximum(), printNoValuesWarningMessage(), and WRN\_MAXIMUM\_FUNCTION\_NAME.

**6.117.2.31 double Numeric::maximum ( const std::vector< double > & *numbers*, unsigned int *nrOfValues* ) [static, private]**

Return the maximum of the provided numbers.

**Parameters**

|                   |                           |
|-------------------|---------------------------|
| <i>numbers</i>    | The collection of numbers |
| <i>nrOfValues</i> | The number of values      |

Definition at line 376 of file Numeric.cpp.

References maximum().

**6.117.2.32 double Numeric::median ( const std::vector< double > & *numbers* )  
[static]**

Return the median of the provided numbers.

**Parameters**

|                |                           |
|----------------|---------------------------|
| <i>numbers</i> | The collection of numbers |
|----------------|---------------------------|

Definition at line 150 of file Numeric.cpp.

References printNoValuesWarningMessage(), and WRN\_MEDIAN\_FUNCTION\_NAME.

Referenced by multiscale::verification::NumericEvaluator::evaluate().

**6.117.2.33 double Numeric::median ( const std::vector< double > & numbers, unsigned int nrOfValues ) [static, private]**

Return the median of the provided numbers.

#### Parameters

|                   |                           |
|-------------------|---------------------------|
| <i>numbers</i>    | The collection of numbers |
| <i>nrOfValues</i> | The number of values      |

Definition at line 389 of file Numeric.cpp.

**6.117.2.34 double Numeric::minimum ( const std::vector< double > & numbers ) [static]**

Return the minimum of the provided numbers.

#### Parameters

|                |                           |
|----------------|---------------------------|
| <i>numbers</i> | The collection of numbers |
|----------------|---------------------------|

Definition at line 162 of file Numeric.cpp.

References printNoValuesWarningMessage(), and WRN\_MINIMUM\_FUNCTION\_NAME.

Referenced by multiscale::verification::NumericEvaluator::evaluate(), and minimum().

**6.117.2.35 double Numeric::minimum ( const std::vector< double > & numbers, unsigned int nrOfValues ) [static, private]**

Return the minimum of the provided numbers.

#### Parameters

|                   |                           |
|-------------------|---------------------------|
| <i>numbers</i>    | The collection of numbers |
| <i>nrOfValues</i> | The number of values      |

Definition at line 398 of file Numeric.cpp.

References minimum().

```
6.117.2.36 double Numeric::mode ( const std::vector< double > & numbers )  
[static]
```

Return the mode of the provided numbers.

Parameters

|                |                           |
|----------------|---------------------------|
| <i>numbers</i> | The collection of numbers |
|----------------|---------------------------|

Definition at line 174 of file Numeric.cpp.

References printNoValuesWarningMessage(), and WRN\_MODE\_FUNCTION\_NAME.

Referenced by multiscale::verification::NumericEvaluator::evaluate().

```
6.117.2.37 double Numeric::mode ( const std::vector< double > & values, unsigned int  
nrOfValues ) [static, private]
```

Compute the mode for the provided values.

Parameters

|                   |                      |
|-------------------|----------------------|
| <i>values</i>     | The values           |
| <i>nrOfValues</i> | The number of values |

Definition at line 411 of file Numeric.cpp.

References computeMode().

```
6.117.2.38 template<typename T> static T multiscale::Numeric::numberInverse ( T  
number ) [inline, static, private]
```

Return the inverse of a number.

If the number is equal to zero then a warning is displayed and the default value "0" is returned

Parameters

|               |                  |
|---------------|------------------|
| <i>number</i> | The given number |
|---------------|------------------|

Definition at line 481 of file Numeric.hpp.

References almostEqual(), multiscale::ConsolePrinter::printWarningMessage(), and WRN\_NUMBER\_INVERSE.

Referenced by harmonicMean().

---

**6.117.2.39 double Numeric::percentile ( const std::vector< double > & *numbers*, double *percentile* ) [static]**

Return the p-th percentile of the provided set of values.

**Parameters**

|                   |                          |
|-------------------|--------------------------|
| <i>numbers</i>    | The collection of values |
| <i>percentile</i> | The p-th percentile      |

Definition at line 186 of file Numeric.cpp.

References printNoValuesWarningMessage(), and WRN\_PERCENTILE\_FUNCTION\_NAME.

Referenced by multiscale::verification::NumericEvaluator::evaluate().

**6.117.2.40 double Numeric::percentile ( const std::vector< double > & *numbers*, double *percentile*, unsigned int *nrOfValues* ) [static, private]**

Return the p-th percentile of the provided set of values.

**Parameters**

|                   |                          |
|-------------------|--------------------------|
| <i>numbers</i>    | The collection of values |
| <i>percentile</i> | The p-th percentile      |
| <i>nrOfValues</i> | The number of values     |

Definition at line 446 of file Numeric.cpp.

References validatePercentile().

**6.117.2.41 void Numeric::printNoValuesWarningMessage ( const std::string & *functionName* ) [static, private]**

Print the no values warning message for the given function name.

**Parameters**

|                           |                            |
|---------------------------|----------------------------|
| <i>function-<br/>Name</i> | The provided function name |
|---------------------------|----------------------------|

Definition at line 274 of file Numeric.cpp.

References multiscale::ConsolePrinter::printWarningMessage(), WRN\_NOT\_ENOUGH\_VALUES\_END, and WRN\_NOT\_ENOUGH\_VALUES\_START.

Referenced by average(), covariance(), geometricMean(), harmonicMean(), kurtosis(), maximum(), median(), minimum(), mode(), percentile(), product(), quartile(), skew(), standardDeviation(), sum(), and variance().

6.117.2.42 double Numeric::product ( const std::vector< double > & *numbers* )  
[static]

Return the product of the provided numbers.

Parameters

|                |                           |
|----------------|---------------------------|
| <i>numbers</i> | The collection of numbers |
|----------------|---------------------------|

Definition at line 198 of file Numeric.cpp.

References printNoValuesWarningMessage(), and WRN\_PRODUCT\_FUNCTION\_NAME.

Referenced by multiscale::verification::NumericEvaluator::evaluate(), and product().

6.117.2.43 double Numeric::product ( const std::vector< double > & *numbers*, unsigned int *nrOfValues* ) [static, private]

Return the product of the provided numbers.

Parameters

|                   |                           |
|-------------------|---------------------------|
| <i>numbers</i>    | The collection of numbers |
| <i>nrOfValues</i> | The number of values      |

Definition at line 458 of file Numeric.cpp.

References applyOperation(), and product().

6.117.2.44 double Numeric::quartile ( const std::vector< double > & *numbers*, double *quartile* ) [static]

Return the q-th quartile of the provided set of values.

Parameters

|                 |                          |
|-----------------|--------------------------|
| <i>numbers</i>  | The collection of values |
| <i>quartile</i> | The q-th quartile        |

Definition at line 210 of file Numeric.cpp.

References printNoValuesWarningMessage(), and WRN\_QUARTILE\_FUNCTION\_NAME.

Referenced by multiscale::verification::NumericEvaluator::evaluate().

---

**6.117.2.45 double Numeric::quartile ( const std::vector< double > & *numbers*, double *quartile*, unsigned int *nrOfValues* ) [static, private]**

Return the q-th quartile of the provided set of values.

**Parameters**

|                   |                          |
|-------------------|--------------------------|
| <i>numbers</i>    | The collection of values |
| <i>quartile</i>   | The q-th quartile        |
| <i>nrOfValues</i> | The number of values     |

Definition at line 468 of file Numeric.cpp.

References computeQuartileValue(), and validateQuartile().

**6.117.2.46 void Numeric::resetOverflowUnderflowFlags ( ) [static, private]**

Reset the overflow and underflow flags.

Definition at line 556 of file Numeric.cpp.

Referenced by applyOperation().

**6.117.2.47 template<typename FloatingPointType> static FloatingPointType multiscale::Numeric::round ( FloatingPointType *number*, unsigned int *precision* = 2 ) [inline, static]**

Round the number considering the given precision.

**Parameters**

|                  |                          |
|------------------|--------------------------|
| <i>number</i>    | The given number         |
| <i>precision</i> | The considered precision |

Definition at line 179 of file Numeric.hpp.

Referenced by multiscale::analysis::MatFactory::readNextValueFromFile().

**6.117.2.48 int Numeric::sign ( double *number* ) [static]**

Return the sign of the number.

The sign function returns: -1, if number < 0 +1, if number > 0 0, otherwise

**Parameters**

|               |                       |
|---------------|-----------------------|
| <i>number</i> | The considered number |
|---------------|-----------------------|

Definition at line 234 of file Numeric.cpp.

Referenced by multiscale::Geometry2D::areOnTheSameSideOfLine(), and multiscale::verification::NumericEvaluator::evaluate().

**6.117.2.49 double Numeric::skew ( const std::vector< double > & *numbers* ) [static]**

Return the skew of the provided numbers.

**Parameters**

|                |                           |
|----------------|---------------------------|
| <i>numbers</i> | The collection of numbers |
|----------------|---------------------------|

Definition at line 222 of file Numeric.cpp.

References printNoValuesWarningMessage(), and WRN\_SKW\_FUNCTION\_NAME.

Referenced by multiscale::verification::NumericEvaluator::evaluate().

**6.117.2.50 double Numeric::skew ( const std::vector< double > & *numbers*, unsigned int *nrOfValues* ) [static, private]**

Return the skew of the provided numbers.

**Parameters**

|                   |                           |
|-------------------|---------------------------|
| <i>numbers</i>    | The collection of numbers |
| <i>nrOfValues</i> | The number of values      |

Definition at line 493 of file Numeric.cpp.

References computeSkewFirstTerm(), and computeSkewLastTerm().

**6.117.2.51 double Numeric::standardDeviation ( const std::vector< double > & *numbers* ) [static]**

Return the standard deviation of the provided set of values.

**Parameters**

|                |                          |
|----------------|--------------------------|
| <i>numbers</i> | The collection of values |
|----------------|--------------------------|

Definition at line 238 of file Numeric.cpp.

References printNoValuesWarningMessage(), and WRN\_STANDARD\_DEVIATION\_FUNCTION\_NAME.

Referenced by computeKurtosisMiddleTerm(), computeSkewLastTerm(), and multiscale::verification::NumericEvaluator::evaluate().

---

**6.117.2.52 double Numeric::standardDeviation ( const std::vector< double > & numbers, unsigned int nrOfValues ) [static, private]**

Return the standard deviation of the provided set of values.

**Parameters**

|                   |                          |
|-------------------|--------------------------|
| <i>numbers</i>    | The collection of values |
| <i>nrOfValues</i> | The number of values     |

Definition at line 522 of file Numeric.cpp.

References applyOperation(), and average().

**6.117.2.53 double Numeric::sum ( const std::vector< double > & numbers ) [static]**

Return the sum of the provided numbers.

**Parameters**

|                |                           |
|----------------|---------------------------|
| <i>numbers</i> | The collection of numbers |
|----------------|---------------------------|

Definition at line 250 of file Numeric.cpp.

References printNoValuesWarningMessage(), and WRN\_SUM\_FUNCTION\_NAME.

Referenced by average(), multiscale::verification::NumericEvaluator::evaluate(), and sum().

**6.117.2.54 double Numeric::sum ( const std::vector< double > & numbers, unsigned int nrOfValues ) [static, private]**

Return the sum of the provided numbers.

**Parameters**

|                   |                           |
|-------------------|---------------------------|
| <i>numbers</i>    | The collection of numbers |
| <i>nrOfValues</i> | The number of values      |

Definition at line 534 of file Numeric.cpp.

References applyOperation(), and sum().

**6.117.2.55 void Numeric::validateLogBase ( double base ) [static, private]**

Check if the base is a positive real number different from 1.

**Parameters**

|                   |                     |
|-------------------|---------------------|
| <code>base</code> | The considered base |
|-------------------|---------------------|

Definition at line 580 of file Numeric.cpp.

References `almostEqual()`, `ERR_LOG_BASE_END`, `ERR_LOG_BASE_START`, `isPositive()`, and `multiscale::StringManipulator::toString()`.

Referenced by `validateLogNumberAndBase()`.

**6.117.2.56 void Numeric::validateLogNumber ( double *number* ) [static, private]**

Check if the number is a positive real number.

**Parameters**

|                     |                       |
|---------------------|-----------------------|
| <code>number</code> | The considered number |
|---------------------|-----------------------|

Definition at line 573 of file Numeric.cpp.

References `ERR_LOG_NUMBER_END`, `ERR_LOG_NUMBER_START`, `isPositive()`, and `multiscale::StringManipulator::toString()`.

Referenced by `geometricMean()`, and `validateLogNumberAndBase()`.

**6.117.2.57 void Numeric::validateLogNumberAndBase ( double *number*, double *base* ) [static, private]**

Check if the number and the base are positive real numbers, and if the base is different from 1.

**Parameters**

|                     |                       |
|---------------------|-----------------------|
| <code>number</code> | The considered number |
| <code>base</code>   | The considered base   |

Definition at line 568 of file Numeric.cpp.

References `validateLogBase()`, and `validateLogNumber()`.

Referenced by `log()`.

**6.117.2.58 void Numeric::validatePercentile ( double *percentile* ) [static, private]**

Check if the value of the percentile is between 0 and 100.

**Parameters**

|                   |                      |
|-------------------|----------------------|
| <i>percentile</i> | The percentile value |
|-------------------|----------------------|

Definition at line 587 of file Numeric.cpp.

References ERR\_PERCENTILE\_VALUE\_END, and ERR\_PERCENTILE\_VALUE\_START.

Referenced by percentile().

**6.117.2.59 void Numeric::validateQuartile ( double *quartile* ) [static, private]**

Check if the value of the quartile is either 25, 50 or 75.

**Parameters**

|                 |                    |
|-----------------|--------------------|
| <i>quartile</i> | The quartile value |
|-----------------|--------------------|

Definition at line 594 of file Numeric.cpp.

References almostEqual(), ERR\_QUARTILE\_VALUE\_END, and ERR\_QUARTILE\_VALUE\_START.

Referenced by quartile().

**6.117.2.60 double Numeric::variance ( const std::vector< double > & *numbers* ) [static]**

Return the variance of the provided set of values.

**Parameters**

|                |                          |
|----------------|--------------------------|
| <i>numbers</i> | The collection of values |
|----------------|--------------------------|

Definition at line 262 of file Numeric.cpp.

References printNoValuesWarningMessage(), and WRN\_VARIANCE\_FUNCTION\_NAME.

Referenced by multiscale::verification::NumericEvaluator::evaluate().

**6.117.2.61 double Numeric::variance ( const std::vector< double > & *numbers*, unsigned int *nrOfValues* ) [static, private]**

Return the variance of the provided set of values.

**Parameters**

|                   |                          |
|-------------------|--------------------------|
| <i>numbers</i>    | The collection of values |
| <i>nrOfValues</i> | The number of values     |

Definition at line 544 of file Numeric.cpp.

References applyOperation(), and average().

### 6.117.3 Member Data Documentation

**6.117.3.1 double Numeric::epsilon = 1E-9 [static, private]**

Value of epsilon used to compare two real numbers

Definition at line 20 of file Numeric.hpp.

Referenced by almostEqual().

**6.117.3.2 const std::string Numeric::ERR\_COMBINATIONS\_END = ") when computing combinations." [static, private]**

Definition at line 502 of file Numeric.hpp.

Referenced by combinations().

**6.117.3.3 const std::string Numeric::ERR\_COMBINATIONS\_MIDDLE = ") should be greater or equal to the number of elements in each group k (" [static, private]**

Definition at line 501 of file Numeric.hpp.

Referenced by combinations().

**6.117.3.4 const std::string Numeric::ERR\_COMBINATIONS\_START = "The provided number of elements n (" [static, private]**

Definition at line 500 of file Numeric.hpp.

Referenced by combinations().

**6.117.3.5 const std::string Numeric::ERR\_LOG\_BASE\_END = ") should be a positive real number different from 1. Please change." [static, private]**

Definition at line 494 of file Numeric.hpp.

Referenced by validateLogBase().

**6.117.3.6 const std::string Numeric::ERR\_LOG\_BASE\_START = "The base provided to the log function (" [static, private]**

Definition at line 493 of file Numeric.hpp.

Referenced by validateLogBase().

---

6.117.3.7 `const std::string Numeric::ERR_LOG_NUMBER_END = ") should be a positive real number. Please change." [static, private]`

Definition at line 496 of file Numeric.hpp.

Referenced by validateLogNumber().

6.117.3.8 `const std::string Numeric::ERR_LOG_NUMBER_START = "The number provided to the log function (" [static, private]`

Definition at line 495 of file Numeric.hpp.

Referenced by validateLogNumber().

6.117.3.9 `const std::string Numeric::ERR_OVERFLOW_UNDERFLOW = "An underflow/overflow exception occurred." [static, private]`

Definition at line 498 of file Numeric.hpp.

Referenced by applyOperation().

6.117.3.10 `const std::string Numeric::ERR_PERCENTILE_VALUE_END = ") should be between 0 and 100. Please change." [static, private]`

Definition at line 505 of file Numeric.hpp.

Referenced by validatePercentile().

6.117.3.11 `const std::string Numeric::ERR_PERCENTILE_VALUE_START = "The provided percentile value (" [static, private]`

Definition at line 504 of file Numeric.hpp.

Referenced by validatePercentile().

6.117.3.12 `const std::string Numeric::ERR_QUARTILE_VALUE_END = ") should be 25, 50 or 75. Please change." [static, private]`

Definition at line 508 of file Numeric.hpp.

Referenced by validateQuartile().

6.117.3.13 `const std::string Numeric::ERR_QUARTILE_VALUE_START = "The provided quartile value (" [static, private]`

Definition at line 507 of file Numeric.hpp.

Referenced by validateQuartile().

---

```
6.117.3.14 const std::string Numeric::WRN_AVERAGE_FUNCTION_NAME = "average"
[static, private]
```

Definition at line 515 of file Numeric.hpp.

Referenced by average().

```
6.117.3.15 const std::string Numeric::WRN_COVARIANCE_FUNCTION_NAME =
"covariance" [static, private]
```

Definition at line 516 of file Numeric.hpp.

Referenced by covariance().

```
6.117.3.16 const std::string Numeric::WRN_GEOMETRIC_MEAN_FUNCTION_NAME =
= "geometricMean" [static, private]
```

Definition at line 517 of file Numeric.hpp.

Referenced by geometricMean().

```
6.117.3.17 const std::string Numeric::WRN_HARMONIC_MEAN_FUNCTION_NAME =
= "harmonicMean" [static, private]
```

Definition at line 518 of file Numeric.hpp.

Referenced by harmonicMean().

```
6.117.3.18 const std::string Numeric::WRN_KURTOSIS_FUNCTION_NAME =
= "kurtosis" [static, private]
```

Definition at line 519 of file Numeric.hpp.

Referenced by kurtosis().

```
6.117.3.19 const std::string Numeric::WRN_MAXIMUM_FUNCTION_NAME =
= "maximum" [static, private]
```

Definition at line 520 of file Numeric.hpp.

Referenced by maximum().

```
6.117.3.20 const std::string Numeric::WRN_MEDIAN_FUNCTION_NAME = "median"
[static, private]
```

Definition at line 521 of file Numeric.hpp.

Referenced by median().

```
6.117.3.21 const std::string Numeric::WRN_MINIMUM_FUNCTION_NAME = "minimum"  
[static, private]
```

Definition at line 523 of file Numeric.hpp.

Referenced by minimum().

```
6.117.3.22 const std::string Numeric::WRN_MODE_FUNCTION_NAME = "mode"  
[static, private]
```

Definition at line 522 of file Numeric.hpp.

Referenced by mode().

```
6.117.3.23 const std::string Numeric::WRN_NOT_ENOUGH_VALUES_END = "(...)"  
function. The default value \"0\" was returned." [static, private]
```

Definition at line 513 of file Numeric.hpp.

Referenced by printNoValuesWarningMessage().

```
6.117.3.24 const std::string Numeric::WRN_NOT_ENOUGH_VALUES_START = "You  
provided less than the minimum required number of values to the Numeric::"  
[static, private]
```

Definition at line 512 of file Numeric.hpp.

Referenced by printNoValuesWarningMessage().

```
6.117.3.25 const std::string Numeric::WRN_NUMBER_INVERSE = "You provided the  
invalid value \"0\" to the Numeric::inverse(...) function. The default value \"0\" was  
returned." [static, private]
```

Definition at line 510 of file Numeric.hpp.

Referenced by numberInverse().

```
6.117.3.26 const std::string Numeric::WRN_PERCENTILE_FUNCTION_NAME =  
"percentile" [static, private]
```

Definition at line 524 of file Numeric.hpp.

Referenced by percentile().

```
6.117.3.27 const std::string Numeric::WRN_PRODUCT_FUNCTION_NAME = "product"  
[static, private]
```

Definition at line 525 of file Numeric.hpp.

---

Referenced by product().

6.117.3.28 `const std::string Numeric::WRN_QUARTILE_FUNCTION_NAME = "quartile"`  
[static, private]

Definition at line 526 of file Numeric.hpp.

Referenced by quartile().

6.117.3.29 `const std::string Numeric::WRN_SKEW_FUNCTION_NAME = "skew"`  
[static, private]

Definition at line 527 of file Numeric.hpp.

Referenced by skew().

6.117.3.30 `const std::string Numeric::WRN_STANDARD_DEVIATION_FUNCTION_NAME = "standardDeviation"` [static,  
private]

Definition at line 528 of file Numeric.hpp.

Referenced by standardDeviation().

6.117.3.31 `const std::string Numeric::WRN_SUM_FUNCTION_NAME = "sum"`  
[static, private]

Definition at line 529 of file Numeric.hpp.

Referenced by sum().

6.117.3.32 `const std::string Numeric::WRN_VARIANCE_FUNCTION_NAME = "variance"` [static, private]

Definition at line 530 of file Numeric.hpp.

Referenced by variance().

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

- Numeric.hpp
- Numeric.cpp

## 6.118 multiscale::verification::NumericEvaluator Class Reference

Class for evaluating numeric expressions.

```
#include <NumericEvaluator.hpp>
```

## Static Public Member Functions

- template<typename T >  
 static double **evaluate** (const **UnaryNumericMeasureType** &unaryNumericMeasure, T value)  
*Evaluate the given unary numeric expression.*
- template<typename T >  
 static double **evaluate** (const **BinaryNumericMeasureType** &binaryNumericMeasure, T firstValue, T secondValue)  
*Evaluate the given binary numeric expression.*
- static double **evaluate** (const **UnaryStatisticalMeasureType** &unaryStatisticalMeasure, const std::vector< double > &values)  
*Evaluate the given unary statistical measure expression.*
- static double **evaluate** (const **BinaryStatisticalMeasureType** &binaryStatisticalMeasure, const std::vector< double > &values1, const std::vector< double > &values2)  
*Evaluate the given binary statistical measure expression.*
- static double **evaluate** (const **BinaryStatisticalQuantileMeasureType** &binaryStatisticalQuantileMeasure, const std::vector< double > &values, double parameter)  
*Evaluate the given binary statistical quantile measure expression.*

### 6.118.1 Detailed Description

Class for evaluating numeric expressions.

Definition at line 14 of file NumericEvaluator.hpp.

### 6.118.2 Member Function Documentation

- 6.118.2.1 template<typename T > static double multiscale::verification::-  
**NumericEvaluator::evaluate** ( const **UnaryNumericMeasureType** &unaryNumericMeasure, T value ) [inline, static]

Evaluate the given unary numeric expression.

#### Parameters

|                              |                                                          |
|------------------------------|----------------------------------------------------------|
| <i>unary-Numeric-Measure</i> | The unary numeric measure type                           |
| <i>value</i>                 | The value for which the unary numeric measure is applied |

Definition at line 24 of file NumericEvaluator.hpp.

References multiscale::ERR\_UNDEFINED\_ENUM\_VALUE, and multiscale::Numeric::sign().

Referenced by multiscale::verification::SpatialMeasureCollectionVisitor::evaluateBinaryNumericSpatialMeasureCollection(), multiscale::verification::SpatialMeasureCollectionVisitor::evaluateUnaryNumericSpatialMeasureCollection(), multiscale::verification::FilterNumericVisitor::operator()(), multiscale::verification::TemporalNumericVisitor::operator()(), and multiscale::verification::NumericVisitor::operator()().

**6.118.2.2 template<typename T > static double multiscale::verification::-  
NumericEvaluator::evaluate ( const BinaryNumericMeasureType &  
binaryNumericMeasure, T firstValue, T secondValue ) [inline, static]**

Evaluate the given binary numeric expression.

#### Parameters

|                               |                                                                  |
|-------------------------------|------------------------------------------------------------------|
| <i>binary-Numeric-Measure</i> | The binary numeric measure type                                  |
| <i>firstValue</i>             | The first value for which the binary numeric measure is applied  |
| <i>secondValue</i>            | The second value for which the binary numeric measure is applied |

Definition at line 62 of file NumericEvaluator.hpp.

References multiscale::Numeric::division(), multiscale::ERR\_UNDEFINED\_ENUM\_VALUE, and multiscale::Numeric::log().

**6.118.2.3 static double multiscale::verification::NumericEvaluator::evaluate ( const  
UnaryStatisticalMeasureType & unaryStatisticalMeasure, const std::vector<  
double > & values ) [inline, static]**

Evaluate the given unary statistical measure expression.

#### Parameters

|                                  |                                     |
|----------------------------------|-------------------------------------|
| <i>unary-Statistical-Measure</i> | The unary statistical measure type  |
| <i>values</i>                    | The considered collection of values |

Definition at line 98 of file NumericEvaluator.hpp.

References multiscale::Numeric::average(), multiscale::ERR\_UNDEFINED\_ENUM\_VALUE, multiscale::Numeric::geometricMean(), multiscale::Numeric::harmonicMean(), multiscale::Numeric::kurtosis(), multiscale::Numeric::maximum(), multiscale::Numeric::median(), multiscale::Numeric::minimum(), multiscale::Numeric::mode(), multiscale::Numeric::product(), multiscale::Numeric::skew(), multiscale::Numeric::standardDeviation(), multiscale::Numeric::sum(), and multiscale::Numeric::variance().

---

6.118.2.4 static double multiscale::verification::NumericEvaluator::evaluate ( const BinaryStatisticalMeasureType & *binaryStatisticalMeasure*, const std::vector< double > & *values1*, const std::vector< double > & *values2* ) [inline, static]

Evaluate the given binary statistical measure expression.

**Parameters**

|                                             |                                            |
|---------------------------------------------|--------------------------------------------|
| <i>binary-<br/>Statistical-<br/>Measure</i> | The binary statistical measure type        |
| <i>values1</i>                              | The first collection of considered values  |
| <i>values2</i>                              | The second collection of considered values |

Definition at line 171 of file NumericEvaluator.hpp.

References multiscale::Numeric::covariance(), and multiscale::ERR\_UNDEFINED\_ENUM\_VALUE.

6.118.2.5 static double multiscale::verification::NumericEvaluator::evaluate ( const BinaryStatisticalQuantileMeasureType & *binaryStatisticalQuantileMeasure*, const std::vector< double > & *values*, double *parameter* ) [inline, static]

Evaluate the given binary statistical quantile measure expression.

**Parameters**

|                                                           |                                                  |
|-----------------------------------------------------------|--------------------------------------------------|
| <i>binary-<br/>Statistical-<br/>Quantile-<br/>Measure</i> | The binary statistical quantile measure type     |
| <i>values</i>                                             | The considered values                            |
| <i>parameter</i>                                          | The parameter used by the ternary subset measure |

Definition at line 191 of file NumericEvaluator.hpp.

References multiscale::ERR\_UNDEFINED\_ENUM\_VALUE, multiscale::Numeric::percentile(), and multiscale::Numeric::quartile().

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

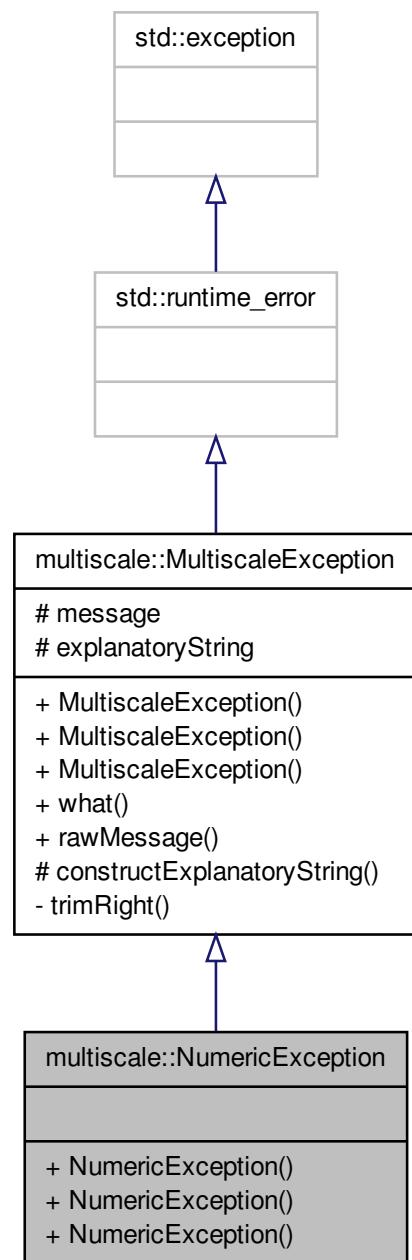
- NumericEvaluator.hpp

## 6.119 multiscale::NumericException Class Reference

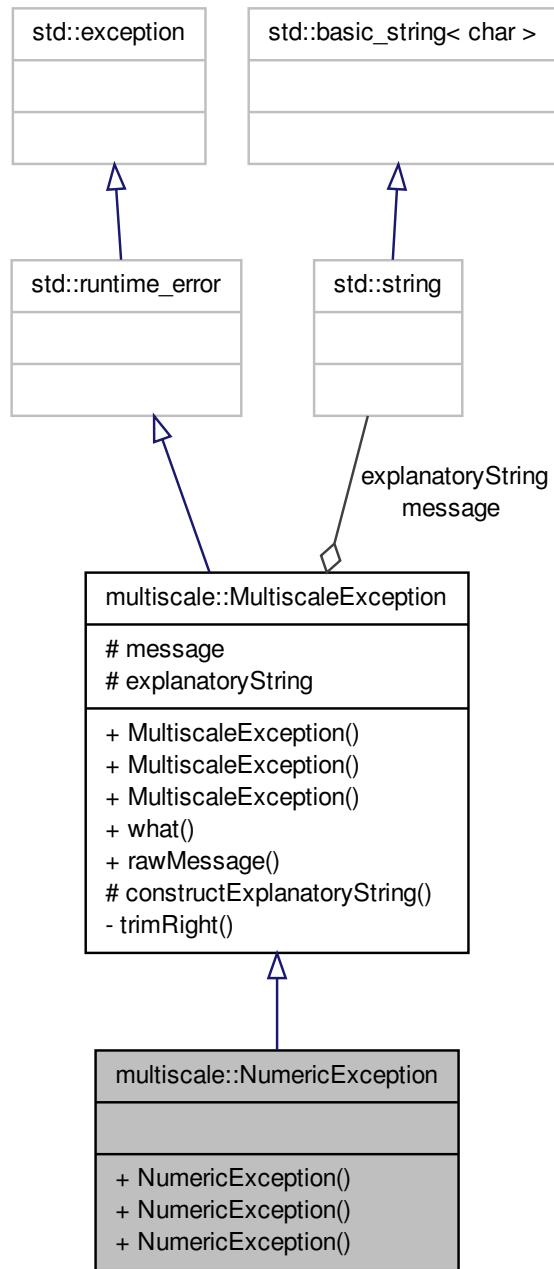
Class for representing algorithm exceptions.

```
#include <NumericException.hpp>
```

Inheritance diagram for multiscale::NumericException:



Collaboration diagram for multiscale::NumericException:



## Public Member Functions

- [NumericException \(\)](#)
- [NumericException \(const std::string &file, int line, const std::string &msg\)](#)
- [NumericException \(const std::string &file, int line, const char \\*msg\)](#)

### 6.119.1 Detailed Description

Class for representing algorithm exceptions.

Definition at line 12 of file NumericException.hpp.

### 6.119.2 Constructor & Destructor Documentation

#### 6.119.2.1 multiscale::NumericException::NumericException( ) [inline]

Definition at line 16 of file NumericException.hpp.

#### 6.119.2.2 multiscale::NumericException::NumericException( const std::string & file, int line, const std::string & msg ) [inline, explicit]

Definition at line 18 of file NumericException.hpp.

#### 6.119.2.3 multiscale::NumericException::NumericException( const std::string & file, int line, const char \* msg ) [inline, explicit]

Definition at line 23 of file NumericException.hpp.

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

- [NumericException.hpp](#)

## 6.120 multiscale::verification::NumericMeasureAttribute Class - Reference

Class for representing a numeric measure attribute.

```
#include <NumericMeasureAttribute.hpp>
```

## Public Attributes

- [NumericMeasureType numericMeasure](#)

### 6.120.1 Detailed Description

Class for representing a numeric measure attribute.

Definition at line 35 of file NumericMeasureAttribute.hpp.

### 6.120.2 Member Data Documentation

#### 6.120.2.1 **NumericMeasureType multiscale::verification::NumericMeasureAttribute::numericMeasure**

The numeric measure

Definition at line 39 of file NumericMeasureAttribute.hpp.

Referenced by multiscale::verification::NumericVisitor::operator()().

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

- NumericMeasureAttribute.hpp

## 6.121 **multiscale::verification::NumericMeasureCollectionAttribute Class Reference**

Class for representing a numeric measure collection attribute.

```
#include <NumericMeasureCollectionAttribute.hpp>
```

### Public Attributes

- [NumericMeasureCollectionType numericMeasureCollection](#)

### 6.121.1 Detailed Description

Class for representing a numeric measure collection attribute.

Definition at line 22 of file NumericMeasureCollectionAttribute.hpp.

### 6.121.2 Member Data Documentation

#### 6.121.2.1 **NumericMeasureCollectionType multiscale::verification::NumericMeasureCollectionAttribute::numericMeasureCollection**

The numeric measure collection

Definition at line 26 of file NumericMeasureCollectionAttribute.hpp.

Referenced by multiscale::verification::TemporalNumericVisitor::evaluateNumericMeasureCollection().

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

- NumericMeasureCollectionAttribute.hpp

## 6.122 multiscale::verification::NumericMeasureCollectionEvaluator Class Reference

Class used to evaluate numeric measure collections.

```
#include <NumericMeasureCollectionEvaluator.hpp>
```

### Static Public Member Functions

- static std::vector< double > evaluateTemporalNumericCollection (SpatialTemporalTrace &trace, const MultiscaleArchitectureGraph &multiscaleArchitectureGraph, const TemporalNumericCollectionAttribute &temporalNumericCollection)  
*Evaluate the given temporal numeric collection.*
- static std::vector< double > evaluateSpatialMeasureCollection (TimePoint &timePoint, const MultiscaleArchitectureGraph &multiscaleArchitectureGraph, const SpatialMeasureCollectionAttribute &spatialMeasureCollection)  
*Evaluate the spatial measure collection considering the given timepoint.*

### 6.122.1 Detailed Description

Class used to evaluate numeric measure collections.

Definition at line 12 of file NumericMeasureCollectionEvaluator.hpp.

### 6.122.2 Member Function Documentation

- #### 6.122.2.1 std::vector< double > multiscale::verification::NumericMeasureCollectionEvaluator::evaluateSpatialMeasureCollection ( TimePoint & timePoint, const MultiscaleArchitectureGraph & multiscaleArchitectureGraph, const SpatialMeasureCollectionAttribute & spatialMeasureCollection ) [inline, static]

Evaluate the spatial measure collection considering the given timepoint.

**Parameters**

|                                      |                                           |
|--------------------------------------|-------------------------------------------|
| <i>timePoint</i>                     | The given timepoint                       |
| <i>multiscale-Architecture-Graph</i> | The given multiscale architecture graph   |
| <i>spatial-Measure-Collection</i>    | The considered spatial measure collection |

Definition at line 65 of file NumericMeasureCollectionEvaluator.hpp.

References multiscale::verification::SpatialMeasureCollectionAttribute::spatialMeasureCollection.

Referenced by multiscale::verification::SpatialMeasureCollectionVisitor::evaluateSpatialMeasureCollection(), multiscale::verification::NumericMeasureCollectionVisitor::operator()(), and multiscale::verification::NumericVisitor::operator()().

```
6.122.2.2 static std::vector<double> multiscale::verification::NumericMeasure-
CollectionEvaluator::evaluateTemporalNumericCollection (
    SpatialTemporalTrace & trace, const MultiscaleArchitectureGraph &
    multiscaleArchitectureGraph, const TemporalNumericCollectionAttribute &
    temporalNumericCollection ) [inline, static]
```

Evaluate the given temporal numeric collection.

**Parameters**

|                                      |                                         |
|--------------------------------------|-----------------------------------------|
| <i>trace</i>                         | The given spatial temporal trace        |
| <i>multiscale-Architecture-Graph</i> | The given multiscale architecture graph |
| <i>temporal-Numeric-Collection</i>   | The given temporal numeric collection   |

Definition at line 23 of file NumericMeasureCollectionEvaluator.hpp.

References multiscale::verification::TemporalNumericCollectionAttribute::temporalNumericCollection.

Referenced by multiscale::verification::NumericMeasureCollectionVisitor::operator()(), and multiscale::verification::LogicPropertyVisitor::operator()().

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

- NumericMeasureCollectionEvaluator.hpp

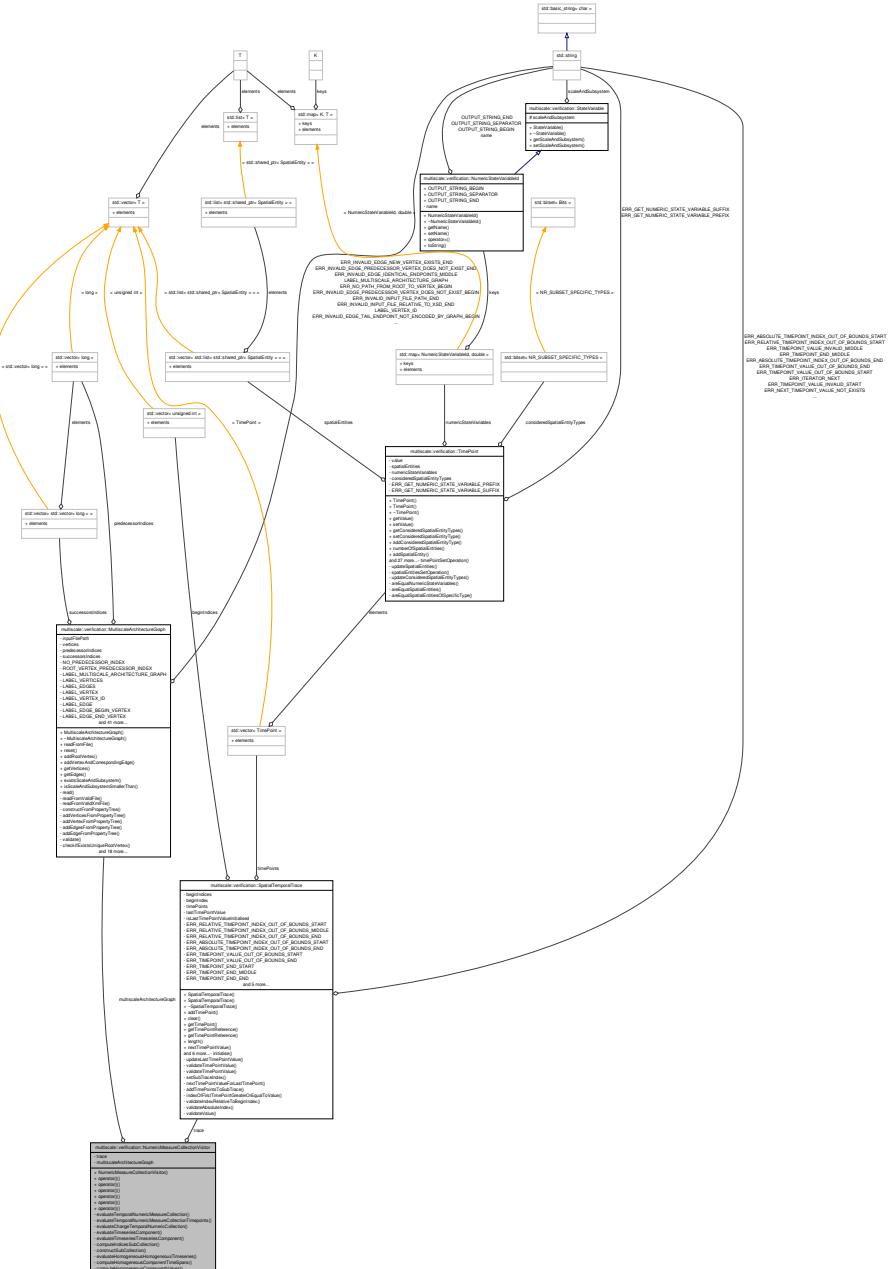
## **6.123 multiscale::verification::NumericMeasureCollectionVisitor Class Reference**

### **6.123 multiscale::verification::NumericMeasureCollectionVisitor Class Reference**

Class for evaluating numeric measure collections.

```
#include <NumericMeasureCollectionVisitor.hpp>
```

## Collaboration diagram for multiscale::verification::NumericMeasureCollectionVisitor:



## Public Member Functions

- `NumericMeasureCollectionVisitor` (`SpatialTemporalTrace &trace`, `const MultiscaleArchitectureGraph &multiscaleArchitectureGraph`)

## **6.123 multiscale::verification::NumericMeasureCollectionVisitor Class Reference**

---

- `std::vector< double > operator() (const TemporalNumericCollectionAttribute &temporalNumericCollection) const`  
*Overloading the "()" operator for the `TemporalNumericCollectionAttribute` alternative.*
- `std::vector< double > operator() (const SpatialMeasureCollectionAttribute &spatialMeasureCollection) const`  
*Overloading the "()" operator for the `SpatialMeasureCollectionAttribute` alternative.*
- `std::vector< double > operator() (const TemporalNumericMeasureCollection-Attribute &temporalNumericMeasureCollection) const`  
*Overloading the "()" operator for the `TemporalNumericMeasureCollectionAttribute` alternative.*
- `std::vector< double > operator() (const ChangeTemporalNumericCollection-Attribute &changeTemporalNumericCollection) const`  
*Overloading the "()" operator for the `ChangeTemporalNumericCollectionAttribute` alternative.*
- `std::vector< double > operator() (const TimeseriesTimeseriesComponent-Attribute &timeseriesTimeseriesComponent) const`  
*Overloading the "()" operator for the `TimeseriesTimeseriesComponent` alternative.*
- `std::vector< double > operator() (const HomogeneousHomogeneous-TimeseriesAttribute &homogeneousHomogeneousTimeseries) const`  
*Overloading the "()" operator for the `HomogeneousHomogeneousTimeseriesAttribute` alternative.*

### **Private Member Functions**

- `std::vector< double > evaluateTemporalNumericMeasureCollection (Spatial-TemporalTrace &trace, unsigned long startTimepoint, unsigned long endTimepoint, const NumericMeasureType &numericMeasure) const`  
*Evaluate the temporal numeric measure collection considering the given spatio-temporal trace.*
- `std::vector< double > evaluateTemporalNumericMeasureCollectionTimepoints (- SpatialTemporalTrace &trace, unsigned long startTimepoint, unsigned long endTimepoint) const`  
*Compute the collection of timepoints considering the given trace, and start and end timepoints.*
- `std::vector< double > evaluateChangeTemporalNumericCollection (const - ChangeMeasureAttribute &changeMeasure, const std::vector< double > &temporalNumericCollectionValues) const`  
*Evaluate the temporal numeric collection values considering the given change measure.*
- `std::vector< std::size_t > evaluateTimeseriesComponent (const std::vector< double > &values, const TimeseriesComponentAttribute &timeseries-Component) const`  
*Evaluate the timeseries component considering the given collection of values.*
- `std::vector< double > evaluateTimeseriesTimeseriesComponent (const - TimeseriesMeasureType &timeseriesMeasureType, const std::vector< double > &values, const std::vector< double > &timePoints, const std::vector< std::size_t > &indices) const`

- `std::vector< std::size_t > computeIndicesSubCollection (const std::vector< std::size_t > &indices, std::size_t startPosition, std::size_t step) const`

*Evaluate the given timeseries timeseries component.*
- `template<typename T> std::vector< T > constructSubCollection (const std::vector< T > &initialCollection, const std::vector< std::size_t > &indices) const`

*Compute the collection of sub-indices from the given collection of indices.*
- `std::vector< double > evaluateHomogeneousHomogeneousTimeseries (const HomogeneousTimeseriesMeasureType &homogeneousTimeseriesMeasureType, const std::vector< double > &values, const std::vector< double > &timePoints, const std::vector< std::size_t > &indices) const`

*Construct sub-collection considering the given indices.*
- `std::vector< double > computeHomogeneousComponentTimeSpans (const std::vector< double > &timePoints, const std::vector< std::size_t > &indices) const`

*Evaluate the given homogenous homogeneous timeseries.*
- `std::vector< double > computeHomogeneousComponentValues (const std::vector< double > &values, const std::vector< std::size_t > &indices) const`

*Compute the time spans from a timepoints collection using the given start and end timepoint indices.*
- `std::vector< double > computeHomogeneousComponentValues (const std::vector< double > &values, const std::vector< std::size_t > &indices) const`

*Compute the values subsequence from a collection considering the given start and end indices.*

## Private Attributes

- `SpatialTemporalTrace & trace`
- `const MultiscaleArchitectureGraph & multiscaleArchitectureGraph`

### 6.123.1 Detailed Description

Class for evaluating numeric measure collections.

Definition at line 15 of file NumericMeasureCollectionVisitor.hpp.

### 6.123.2 Constructor & Destructor Documentation

#### 6.123.2.1 `multiscale::verification::NumericMeasureCollectionVisitor::NumericMeasureCollectionVisitor ( SpatialTemporalTrace & trace, const MultiscaleArchitectureGraph & multiscaleArchitectureGraph ) [inline]`

Definition at line 27 of file NumericMeasureCollectionVisitor.hpp.

### 6.123.3 Member Function Documentation

## **6.123 multiscale::verification::NumericMeasureCollectionVisitor Class Reference**

```
6.123.3.1 std::vector<double> multiscale::verification::NumericMeasureCollection-
Visitor::computeHomogeneousComponentTimeSpans ( const
std::vector< double > & timePoints, const std::vector< std::size_t > & indices )
const [inline, private]
```

Compute the time spans from a timepoints collection using the given start and end timepoint indices.

The provided indices collection contains (start, end) pairs where start/end denotes the starting/ending position of a time span in the initial collection which should be included in the resulting collection.

### Parameters

|                   |                                                                                                               |
|-------------------|---------------------------------------------------------------------------------------------------------------|
| <i>timePoints</i> | The given collection of timePoints                                                                            |
| <i>indices</i>    | The collection of both start and end indices pointing to the start and end positions in the values collection |

Definition at line 376 of file NumericMeasureCollectionVisitor.hpp.

Referenced by evaluateHomogeneousHomogeneousTimeseries().

```
6.123.3.2 std::vector<double> multiscale::verification::NumericMeasure-
CollectionVisitor::computeHomogeneousComponentValues ( const
std::vector< double > & values, const std::vector< std::size_t > & indices ) const
[inline, private]
```

Compute the values subsequence from a collection considering the given start and end indices.

The provided indices collection contains (start, end) pairs where start/end denotes the starting/ending position of a sequence of values in the initial collection which should be included in the resulting collection.

### Parameters

|                |                                                                                                               |
|----------------|---------------------------------------------------------------------------------------------------------------|
| <i>values</i>  | The given collection of values                                                                                |
| <i>indices</i> | The collection of both start and end indices pointing to the start and end positions in the values collection |

Definition at line 406 of file NumericMeasureCollectionVisitor.hpp.

Referenced by evaluateHomogeneousHomogeneousTimeseries().

```
6.123.3.3 std::vector<std::size_t> multiscale::verification::NumericMeasure-
CollectionVisitor::computeIndicesSubCollection ( const std::vector<
std::size_t > & indices, std::size_t startPosition, std::size_t step ) const
[inline, private]
```

Compute the collection of sub-indices from the given collection of indices.

**Parameters**

|                      |                                                                                                                                               |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------|
| <i>indices</i>       | The collection of start and end indices                                                                                                       |
| <i>startPosition</i> | The first considered position from the indices collection                                                                                     |
| <i>step</i>          | The ("number of positions" + 1) between two consecutive elements from the initial indices collection which are included in the sub-collection |

Definition at line 287 of file NumericMeasureCollectionVisitor.hpp.

Referenced by evaluateTimeseriesTimeseriesComponent().

```
6.123.3.4 template<typename T> std::vector<T> multiscale::verification::Numeric-
MeasureCollectionVisitor::constructSubCollection( const std::vector<
T> & initialCollection, const std::vector< std::size_t > & indices ) const
[inline, private]
```

Construct sub-collection considering the given indices.

Only valid indices will be considered. If an invalid index is encountered an exception is thrown.

**Parameters**

|                           |                                                                                                            |
|---------------------------|------------------------------------------------------------------------------------------------------------|
| <i>initial-Collection</i> | The initial collection                                                                                     |
| <i>indices</i>            | The indices pointing to element positions from initial collection which will be included in sub-collection |

Definition at line 311 of file NumericMeasureCollectionVisitor.hpp.

Referenced by evaluateTimeseriesTimeseriesComponent().

```
6.123.3.5 std::vector<double> multiscale::verification::NumericMeasure-
CollectionVisitor::evaluateChangeTemporalNumericCollection( const
ChangeMeasureAttribute & changeMeasure, const std::vector< double > &
temporalNumericCollectionValues ) const [inline, private]
```

Evaluate the temporal numeric collection values considering the given change measure.

**Parameters**

|                                           |                                               |
|-------------------------------------------|-----------------------------------------------|
| <i>change-Measure</i>                     | The given change measure                      |
| <i>temporal-Numeric-Collection-Values</i> | The values in the temporal numeric collection |

Definition at line 196 of file NumericMeasureCollectionVisitor.hpp.

## **6.123 multiscale::verification::NumericMeasureCollectionVisitor Class Reference**

References [multiscale::verification::ChangeMeasureAttribute::changeMeasureType](#), and [multiscale::verification::ChangeMeasureEvaluator::evaluate\(\)](#).

```
6.123.3.6 std::vector<double> multiscale::verification::NumericMeasure-
CollectionVisitor::evaluateHomogeneousHomogeneousTimeseries
( const HomogeneousTimeseriesMeasureType &
homogeneousTimeseriesMeasureType, const std::vector< double > & values, const
std::vector< double > & timePoints, const std::vector< std::size_t > & indices )
const [inline, private]
```

Evaluate the given homogenous homogeneous timeseries.

Provided are the temporal numeric measure collection values, the corresponding timepoints and both start and end timepoints for timeseries components.

### Parameters

|                                            |                                                                                                                           |
|--------------------------------------------|---------------------------------------------------------------------------------------------------------------------------|
| <i>homogeneous-Timeseries-Measure-Type</i> | The considered specific homogeneous timeseries measure type                                                               |
| <i>values</i>                              | The temporal numeric measure collection values                                                                            |
| <i>timePoints</i>                          | The collection of timepoints corresponding to the temporal numeric collection values                                      |
| <i>indices</i>                             | The indices pointing to the starting and ending positions of timeseries components in the collection of values/timepoints |

Definition at line 338 of file NumericMeasureCollectionVisitor.hpp.

References [computeHomogeneousComponentTimeSpans\(\)](#), [computeHomogeneousComponentValues\(\)](#), and [multiscale::ERR\\_UNDEFINED\\_ENUM\\_VALUE](#).

Referenced by [operator\(\)\(\)](#).

```
6.123.3.7 std::vector< double > multiscale::verification::NumericMeasure-
CollectionVisitor::evaluateTemporalNumericMeasureCollection (
SpatialTemporalTrace & trace, unsigned long startTimepoint, unsigned
long endTimepoint, const NumericMeasureType & numericMeasure ) const
[inline, private]
```

Evaluate the temporal numeric measure collection considering the given spatio-temporal trace.

### Parameters

|                        |                                      |
|------------------------|--------------------------------------|
| <i>trace</i>           | The considered spatio-temporal trace |
| <i>start-Timepoint</i> | The considered start timepoint value |
| <i>end-Timepoint</i>   | The considered end timepoint value   |

|                             |                                     |
|-----------------------------|-------------------------------------|
| <i>numeric-<br/>Measure</i> | The numeric measure to be evaluated |
|-----------------------------|-------------------------------------|

Definition at line 487 of file NumericMeasureCollectionVisitor.hpp.

References multiscale::verification::SpatialTemporalTrace::getTimePointReference(), multiscale::verification::SpatialTemporalTrace::nextTimePointValue(), multiscale::verification::SpatialTemporalTrace::restoreSubTraceBeginIndex(), multiscale::verification::SpatialTemporalTrace::setSubTraceWithTimepointsValuesGreaterOrEqualTo(), and multiscale::verification::SpatialTemporalTrace::storeSubTraceBeginIndex().

Referenced by operator()().

```
6.123.3.8 std::vector<double> multiscale::verification::NumericMeasureCollection-
Visitor::evaluateTemporalNumericMeasureCollectionTimepoints (
    SpatialTemporalTrace & trace, unsigned long startTimepoint, unsigned long
    endTimepoint ) const [inline, private]
```

Compute the collection of timepoints considering the given trace, and start and end timepoints.

#### Parameters

|                             |                                      |
|-----------------------------|--------------------------------------|
| <i>trace</i>                | The considered spatio-temporal trace |
| <i>start-<br/>Timepoint</i> | The considered start timepoint value |
| <i>end-<br/>Timepoint</i>   | The considered end timepoint value   |

Definition at line 165 of file NumericMeasureCollectionVisitor.hpp.

References multiscale::verification::SpatialTemporalTrace::nextTimePointValue(), multiscale::verification::SpatialTemporalTrace::restoreSubTraceBeginIndex(), multiscale::verification::SpatialTemporalTrace::setSubTraceWithTimepointsValuesGreaterOrEqualTo(), and multiscale::verification::SpatialTemporalTrace::storeSubTraceBeginIndex().

Referenced by operator()().

```
6.123.3.9 std::vector<std::size_t> multiscale::verification::NumericMeasure-
CollectionVisitor::evaluateTimeseriesComponent ( const
std::vector< double > & values, const TimeseriesComponentAttribute &
timeseriesComponent ) const [inline, private]
```

Evaluate the timeseries component considering the given collection of values.

#### Parameters

|               |                                                                              |
|---------------|------------------------------------------------------------------------------|
| <i>values</i> | The collection of real values for which timeseries components are com- puted |
|---------------|------------------------------------------------------------------------------|

## **6.123 multiscale::verification::NumericMeasureCollectionVisitor Class Reference**

|                             |                                                   |
|-----------------------------|---------------------------------------------------|
| <i>timeseries-Component</i> | The considered specific timeseries component type |
|-----------------------------|---------------------------------------------------|

Definition at line 229 of file NumericMeasureCollectionVisitor.hpp.

References        multiscale::verification::TimeseriesComponentAttribute::timeseries-Component.

Referenced by operator()().

```
6.123.3.10 std::vector<double> multiscale::verification::NumericMeasure-
CollectionVisitor::evaluateTimeseriesTimeseriesComponent ( const
TimeseriesMeasureType & timeseriesMeasureType, const std::vector< double
> & values, const std::vector< double > & timePoints, const std::vector<
std::size_t > & indices ) const [inline, private]
```

Evaluate the given timeseries timeseries component.

Provided are the temporal numeric measure collection values, the corresponding timepoints and both start and end timepoints for timeseries components.

### Parameters

|                                |                                                                                                                           |
|--------------------------------|---------------------------------------------------------------------------------------------------------------------------|
| <i>timeseries-Measure-Type</i> | The considered specific timeseries measure type                                                                           |
| <i>values</i>                  | The temporal numeric measure collection values                                                                            |
| <i>timePoints</i>              | The collection of timepoints corresponding to the temporal numeric collection values                                      |
| <i>indices</i>                 | The indices pointing to the starting and ending positions of timeseries components in the collection of values/timepoints |

Definition at line 251 of file NumericMeasureCollectionVisitor.hpp.

References computeIndicesSubCollection(), constructSubCollection(), and multiscale::ERR\_UNDEFINED\_ENUM\_VALUE.

Referenced by operator()().

```
6.123.3.11 std::vector< double > multiscale::verification::NumericMeasureCollection-
Visitor::operator() ( const TemporalNumericCollectionAttribute &
temporalNumericCollection ) const [inline]
```

Overloading the "(") operator for the [TemporalNumericCollectionAttribute](#) alternative.

### Parameters

|                                    |                                 |
|------------------------------------|---------------------------------|
| <i>temporal-Numeric-Collection</i> | The temporal numeric collection |
|------------------------------------|---------------------------------|

Definition at line 442 of file NumericMeasureCollectionVisitor.hpp.

References multiscale::verification::NumericMeasureCollectionEvaluator::evaluateTemporalNumericCollection(), multiscaleArchitectureGraph, and trace.

```
6.123.3.12 std::vector< double > multiscale::verification::NumericMeasureCollection-
Visitor::operator() ( const SpatialMeasureCollectionAttribute &
spatialMeasureCollection ) const [inline]
```

Overloading the "(") operator for the [SpatialMeasureCollectionAttribute](#) alternative.

#### Parameters

|                                             |                                |
|---------------------------------------------|--------------------------------|
| <i>spatial-<br/>Measure-<br/>Collection</i> | The spatial measure collection |
|---------------------------------------------|--------------------------------|

Definition at line 455 of file NumericMeasureCollectionVisitor.hpp.

References multiscale::verification::NumericMeasureCollectionEvaluator::evaluateSpatialMeasureCollection().

```
6.123.3.13 std::vector<double> multiscale::verification::NumericMeasureCollectionVisitor-
::operator() ( const TemporalNumericMeasureCollectionAttribute &
temporalNumericMeasureCollection ) const [inline]
```

Overloading the "(") operator for the [TemporalNumericMeasureCollectionAttribute](#) alternative.

#### Parameters

|                                                           |                                         |
|-----------------------------------------------------------|-----------------------------------------|
| <i>temporal-<br/>Numeric-<br/>Measure-<br/>Collection</i> | The temporal numeric measure collection |
|-----------------------------------------------------------|-----------------------------------------|

Definition at line 51 of file NumericMeasureCollectionVisitor.hpp.

References multiscale::verification::TemporalNumericMeasureCollectionAttribute::endTimepoint, evaluateTemporalNumericMeasureCollection(), multiscale::verification::TemporalNumericMeasureCollectionAttribute::numericMeasure, multiscale::verification::TemporalNumericMeasureCollectionAttribute::startTimepoint, and trace.

```
6.123.3.14 std::vector< double > multiscale::verification::NumericMeasureCollectionVisitor-
::operator() ( const ChangeTemporalNumericCollectionAttribute &
changeTemporalNumericCollection ) const [inline]
```

Overloading the "(") operator for the [ChangeTemporalNumericCollectionAttribute](#) alternative.

## **6.123 multiscale::verification::NumericMeasureCollectionVisitor Class Reference**

### **Parameters**

|                                           |                                        |
|-------------------------------------------|----------------------------------------|
| <i>change-Temporal-Numeric-Collection</i> | The change temporal numeric collection |
|-------------------------------------------|----------------------------------------|

Definition at line 468 of file NumericMeasureCollectionVisitor.hpp.

References      multiscale::verification::ChangeTemporalNumericCollectionAttribute-  
::changeMeasure,      multiscale::verification::NumericMeasureCollectionEvaluator-  
::evaluateTemporalNumericCollection(), and multiscale::verification::ChangeTemporal-  
NumericCollectionAttribute::temporalNumericCollection.

**6.123.3.15 std::vector<double> multiscale::verification::NumericMeasureCollectionVisitor-  
::operator() ( const TimeseriesTimeseriesComponentAttribute &  
timeseriesTimeseriesComponent ) const [inline]**

Overloading the "(") operator for the TimeseriesTimeseriesComponent alternative.

### **Parameters**

|                                        |                                               |
|----------------------------------------|-----------------------------------------------|
| <i>timeseries-Timeseries-Component</i> | The timeseries measure - timeseries component |
|----------------------------------------|-----------------------------------------------|

Definition at line 74 of file NumericMeasureCollectionVisitor.hpp.

References      multiscale::verification::TemporalNumericMeasureCollectionAttribute-  
::endTimepoint, evaluateTemporalNumericMeasureCollection(), evaluateTemporal-  
NumericMeasureCollectionTimepoints(), evaluateTimeseriesComponent(), evaluate-  
TimeseriesTimeseriesComponent(),      multiscale::verification::TemporalNumeric-  
MeasureCollectionAttribute::numericMeasure,      multiscale::verification::Temporal-  
NumericMeasureCollectionAttribute::startTimepoint, multiscale::verification::Timeseries-  
TimeseriesComponentAttribute::temporalNumericMeasureCollection,      multiscale-  
::verification::TimeseriesTimeseriesComponentAttribute::timeseriesComponent,  
multiscale::verification::TimeseriesTimeseriesComponentAttribute::timeseriesMeasure,  
multiscale::verification::TimeseriesMeasureAttribute::timeseriesMeasure, and trace.

**6.123.3.16 std::vector<double> multiscale::verification::NumericMeasureCollectionVisitor-  
::operator() ( const HomogeneousHomogeneousTimeseriesAttribute &  
homogeneousHomogeneousTimeseries ) const [inline]**

Overloading the "(") operator for the HomogeneousHomogeneousTimeseriesAttribute  
alternative.

### **Parameters**

|                                                                 |                                                  |
|-----------------------------------------------------------------|--------------------------------------------------|
| <i>homogeneous-</i><br><i>Homogeneous-</i><br><i>Timeseries</i> | The homogeneous homogeneous timeseries component |
|-----------------------------------------------------------------|--------------------------------------------------|

Definition at line 111 of file NumericMeasureCollectionVisitor.hpp.

References      multiscale::verification::TemporalNumericMeasureCollectionAttribute::endTimepoint,    multiscale::verification::TimeseriesComponentEvaluator::evaluate(),    evaluateHomogeneousHomogeneousTimeseries(),        evaluateTemporalNumericMeasureCollection(),        evaluateTemporalNumericMeasureCollectionTimepoints(),    multiscale::verification::HomogeneousHomogeneousTimeseriesAttribute::homogeneousTimeseriesComponent,    multiscale::verification::HomogeneousHomogeneousTimeseriesAttribute::homogeneousTimeseriesMeasure,    multiscale::verification::HomogeneousTimeseriesMeasureAttribute::homogeneousTimeseriesMeasure,    multiscale::verification::TemporalNumericMeasureCollectionAttribute::numericMeasure,    multiscale::verification::TemporalNumericMeasureCollectionAttribute::startTimepoint,    multiscale::verification::HomogeneousHomogeneousTimeseriesAttribute::temporalNumericMeasureCollection, and trace.

#### 6.123.4 Member Data Documentation

**6.123.4.1 const MultiscaleArchitectureGraph& multiscale::verification::- NumericMeasureCollectionVisitor::multiscaleArchitectureGraph [private]**

The considered multiscale architecture graph

Definition at line 22 of file NumericMeasureCollectionVisitor.hpp.

Referenced by operator()().

**6.123.4.2 SpatialTemporalTrace& multiscale::verification::NumericMeasure- CollectionVisitor::trace [private]**

The considered spatial temporal trace

Definition at line 20 of file NumericMeasureCollectionVisitor.hpp.

Referenced by operator()().

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

- NumericMeasureCollectionVisitor.hpp

### 6.124 multiscale::NumericRangeManipulator Class Reference

Operations for ranges of numeric values.

```
#include <NumericRangeManipulator.hpp>
```

### Static Public Member Functions

- template<class T , class U >  
 static U **convertFromRange** ( T oldRangeMin, T oldRangeMax, U newRangeMin,  
 U newRangeMax, T oldValue)  
*Convert a value from an old range to a new one.*

#### 6.124.1 Detailed Description

Operations for ranges of numeric values.

Definition at line 10 of file NumericRangeManipulator.hpp.

#### 6.124.2 Member Function Documentation

**6.124.2.1 template<class T , class U > static U multiscale::NumericRangeManipulator-  
 ::convertFromRange ( T oldRangeMin, T oldRangeMax, U newRangeMin, U  
 newRangeMax, T oldValue ) [inline, static]**

Convert a value from an old range to a new one.

##### Parameters

|                          |                              |
|--------------------------|------------------------------|
| <i>oldRange-<br/>Min</i> | The minimum of the old range |
| <i>oldRange-<br/>Max</i> | The maximum of the old range |
| <i>newRange-<br/>Min</i> | The minimum of the new range |
| <i>newRange-<br/>Max</i> | The maximum of the new range |
| <i>oldValue</i>          | The old value                |

Definition at line 23 of file NumericRangeManipulator.hpp.

References multiscale::Numeric::division().

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

- NumericRangeManipulator.hpp

## 6.125 multiscale::verification::NumericSpatialMeasureAttribute - Class Reference

Class for representing a numeric spatial measure attribute.

```
#include <NumericSpatialMeasureAttribute.hpp>
```

### Public Attributes

- [NumericSpatialMeasureType numericSpatialMeasure](#)

#### 6.125.1 Detailed Description

Class for representing a numeric spatial measure attribute.

Definition at line 29 of file NumericSpatialMeasureAttribute.hpp.

#### 6.125.2 Member Data Documentation

##### 6.125.2.1 NumericSpatialMeasureType multiscale::verification::NumericSpatialMeasureAttribute::numericSpatialMeasure

The numeric spatial measure

Definition at line 33 of file NumericSpatialMeasureAttribute.hpp.

Referenced by multiscale::verification::NumericVisitor::operator()().

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

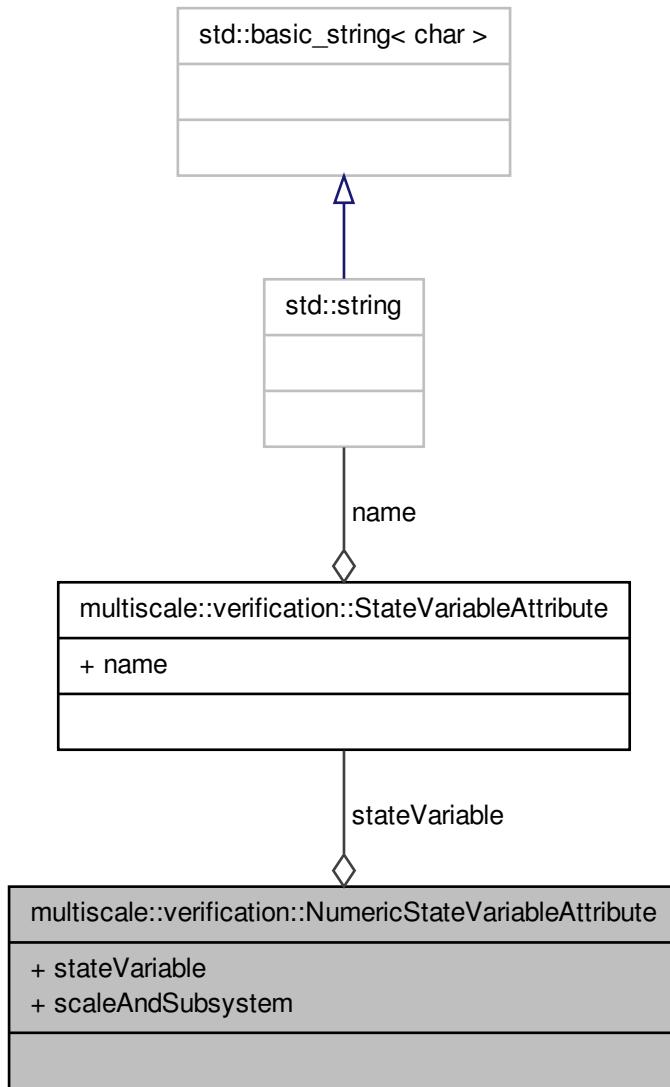
- [NumericSpatialMeasureAttribute.hpp](#)

## 6.126 multiscale::verification::NumericStateVariableAttribute - Class Reference

Class for representing a numeric state variable attribute.

```
#include <NumericStateVariableAttribute.hpp>
```

Collaboration diagram for multiscale::verification::NumericStateVariableAttribute:



### Public Attributes

- `StateVariableAttribute stateVariable`
- `boost::optional < ScaleAndSubsystemAttribute > scaleAndSubsystem`

### 6.126.1 Detailed Description

Class for representing a numeric state variable attribute.

Definition at line 16 of file NumericStateVariableAttribute.hpp.

### 6.126.2 Member Data Documentation

#### 6.126.2.1 boost::optional<ScaleAndSubsystemAttribute> multiscale::verification::NumericStateVariableAttribute::scaleAndSubsystem

The scale and subsystem

Definition at line 23 of file NumericStateVariableAttribute.hpp.

Referenced by multiscale::verification::NumericStateVariableEvaluator::evaluate().

#### 6.126.2.2 StateVariableAttribute multiscale::verification::NumericStateVariableAttribute::stateVariable

The state variable

Definition at line 21 of file NumericStateVariableAttribute.hpp.

Referenced by multiscale::verification::NumericStateVariableEvaluator::evaluate().

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

- NumericStateVariableAttribute.hpp

## 6.127 multiscale::verification::NumericStateVariableEvaluator - Class Reference

Class used to evaluate numeric state variables.

```
#include <NumericStateVariableEvaluator.hpp>
```

### Static Public Member Functions

- static double `evaluate` (const `NumericStateVariableAttribute` &numericStateVariable, const `TimePoint` &timePoint, const `MultiscaleArchitectureGraph` &multiscaleArchitectureGraph)

*Evaluate the provided numeric state variable for the given timepoint.*

### Static Private Member Functions

- static double `evaluate` (const std::string &name, const std::string &scaleAndSubsystem, const `TimePoint` &timePoint)

*Evaluate the provided numeric state variable.*

### 6.127.1 Detailed Description

Class used to evaluate numeric state variables.

Definition at line 14 of file NumericStateVariableEvaluator.hpp.

### 6.127.2 Member Function Documentation

**6.127.2.1 static double multiscale::verification::NumericStateVariableEvaluator-  
::evaluate ( const NumericStateVariableAttribute & *numericStateVariable*,  
const TimePoint & *timePoint*, const MultiscaleArchitectureGraph &  
*multiscaleArchitectureGraph* ) [inline, static]**

Evaluate the provided numeric state variable for the given timepoint.

#### Parameters

|                                                |                                         |
|------------------------------------------------|-----------------------------------------|
| <i>numeric-<br/>State-<br/>Variable</i>        | The provided numeric state variable     |
| <i>timePoint</i>                               | The given timepoint                     |
| <i>multiscale-<br/>Architecture-<br/>Graph</i> | The given multiscale architecture graph |

Definition at line 24 of file NumericStateVariableEvaluator.hpp.

References multiscale::verification::StateVariableAttribute::name, multiscale::verification-  
::NumericStateVariableAttribute::scaleAndSubsystem, multiscale::verification-  
::NumericStateVariableAttribute::stateVariable, and multiscale::verification::ScaleAnd-  
SubsystemEvaluator::validateScaleAndSubsystem().

Referenced by multiscale::verification::TemporalNumericVisitor::operator()(), and  
multiscale::verification::NumericVisitor::operator()().

**6.127.2.2 static double multiscale::verification::NumericStateVariableEvaluator-  
::evaluate ( const std::string & *name*, const std::string & *scaleAndSubsystem*,  
const TimePoint & *timePoint* ) [inline, static, private]**

Evaluate the provided numeric state variable.

#### Parameters

|                                |                                                                    |
|--------------------------------|--------------------------------------------------------------------|
| <i>name</i>                    | The name of the numeric state variable                             |
| <i>scaleAnd-<br/>Subsystem</i> | The scale and subsystem associated with the numeric state variable |
| <i>timePoint</i>               | The given timepoint                                                |

Definition at line 56 of file NumericStateVariableEvaluator.hpp.

References multiscale::verification::TimePoint::getNumericStateVariableValue().

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

- NumericStateVariableEvaluator.hpp

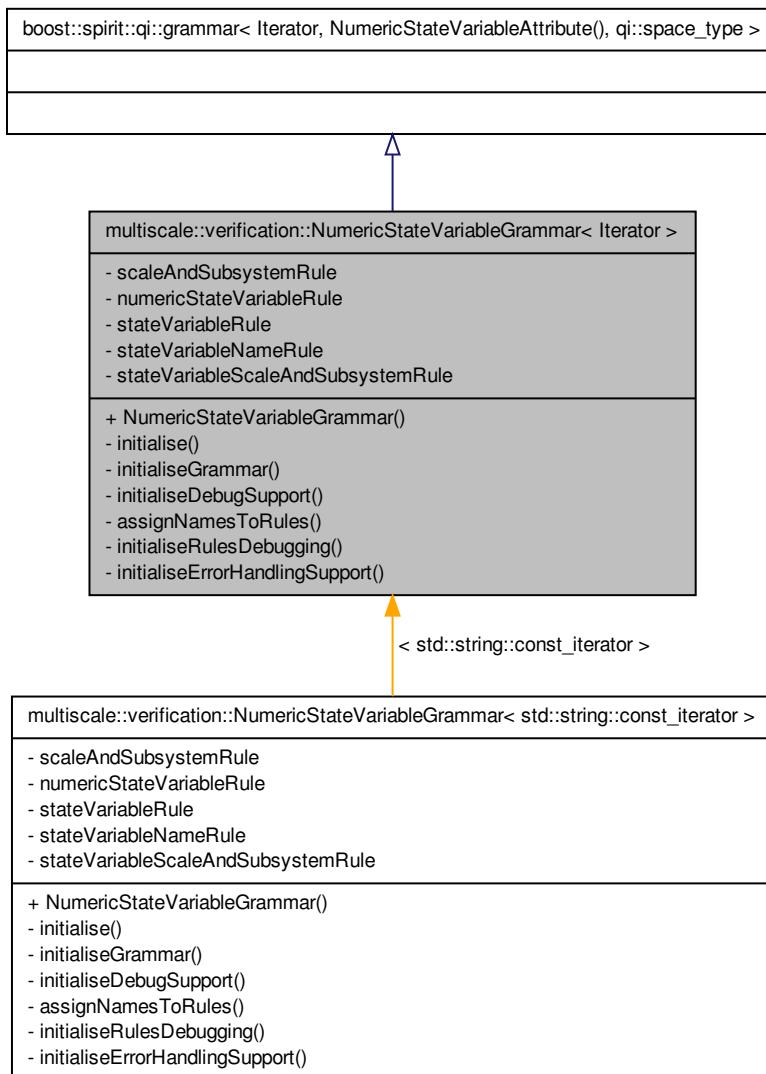
## 6.128 multiscale::verification::NumericStateVariableGrammar< - Iterator > Class Template Reference

The grammar for parsing numeric state variable statements.

```
#include <NumericStateVariableGrammar.hpp>
```

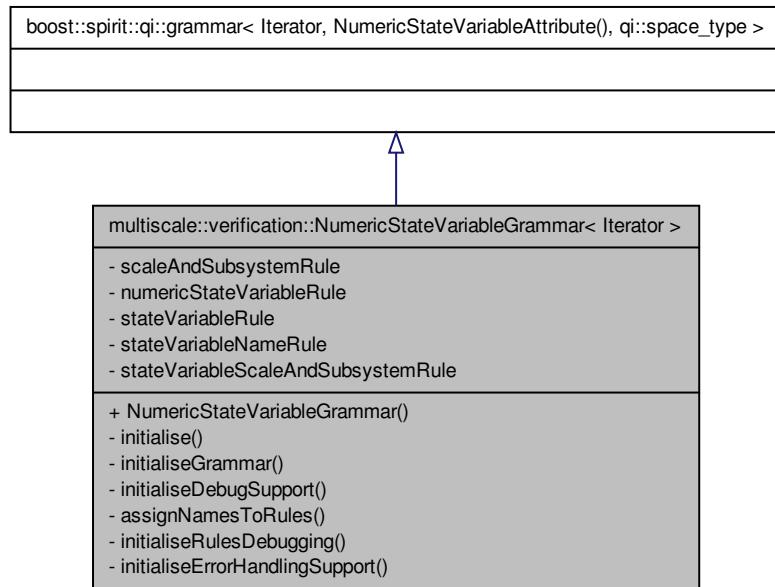
```
Inheritance diagram for multiscale::verification::NumericStateVariableGrammar< -
```

Iterator >:



Collaboration diagram for `multiscale::verification::NumericStateVariableGrammar< ->`

Iterator >:



## Public Member Functions

- [NumericStateVariableGrammar \(\)](#)

## Private Member Functions

- void [initialise \(\)](#)  
*Initialisation function.*
- void [initialiseGrammar \(\)](#)  
*Initialise the grammar.*
- void [initialiseDebugSupport \(\)](#)  
*Initialise debug support.*
- void [assignNamesToRules \(\)](#)  
*Assign names to the rules.*
- void [initialiseRulesDebugging \(\)](#)  
*Initialise the debugging of rules.*
- void [initialiseErrorHandlingSupport \(\)](#)  
*Initialise the error handling routines.*

### Private Attributes

- `ScaleAndSubsystemStringGrammar < Iterator >` `scaleAndSubsystemRule`
- `qi::rule< Iterator, NumericStateVariableAttribute(), qi::space_type >` `numericStateVariableRule`
- `qi::rule< Iterator, StateVariableAttribute(), qi::space_type >` `stateVariableRule`
- `qi::rule< Iterator, std::string(), qi::space_type >` `stateVariableNameRule`
- `qi::rule< Iterator, ScaleAndSubsystemAttribute(), qi::space_type >` `stateVariableScaleAndSubsystemRule`

### 6.128.1 Detailed Description

```
template<typename Iterator> class multiscale::verification::NumericStateVariableGrammar< Iterator >
```

The grammar for parsing numeric state variable statements.

Definition at line 23 of file NumericStateVariableGrammar.hpp.

### 6.128.2 Constructor & Destructor Documentation

```
6.128.2.1 template<typename Iterator> multiscale::verification::NumericState-  
VariableGrammar< Iterator >::NumericStateVariableGrammar ( )
```

Definition at line 23 of file NumericStateVariableGrammarDefinition.hpp.

References `multiscale::verification::NumericStateVariableGrammar< Iterator >::initialise()`.

### 6.128.3 Member Function Documentation

```
6.128.3.1 template<typename Iterator> void multiscale::verification::Numeric-  
StateVariableGrammar< Iterator >::assignNamesToRules ( ) [private]
```

Assign names to the rules.

Definition at line 70 of file NumericStateVariableGrammarDefinition.hpp.

```
6.128.3.2 template<typename Iterator> void multiscale::verification::-  
NumericStateVariableGrammar< Iterator >::initialise ( ) [private]
```

Initialisation function.

Definition at line 30 of file NumericStateVariableGrammarDefinition.hpp.

Referenced by multiscale::verification::NumericStateVariableGrammar< Iterator >::NumericStateVariableGrammar().

6.128.3.3 template<typename Iterator> void multiscale::verification::NumericStateVariableGrammar< Iterator >::initialiseDebugSupport( )  
[private]

Initialise debug support.

Definition at line 61 of file NumericStateVariableGrammarDefinition.hpp.

6.128.3.4 template<typename Iterator> void multiscale::verification::NumericStateVariableGrammar< Iterator >::initialiseErrorHandlingSupport( )  
[private]

Initialise the error handling routines.

Definition at line 88 of file NumericStateVariableGrammarDefinition.hpp.

References multiscale::verification::handleUnexpectedTokenError.

6.128.3.5 template<typename Iterator> void multiscale::verification::NumericStateVariableGrammar< Iterator >::initialiseGrammar( )  
[private]

Initialise the grammar.

Definition at line 38 of file NumericStateVariableGrammarDefinition.hpp.

6.128.3.6 template<typename Iterator> void multiscale::verification::NumericStateVariableGrammar< Iterator >::initialiseRulesDebugging( )  
[private]

Initialise the debugging of rules.

Definition at line 79 of file NumericStateVariableGrammarDefinition.hpp.

#### 6.128.4 Member Data Documentation

6.128.4.1 template<typename Iterator> qi::rule<Iterator, NumericStateVariableAttribute(), qi::space\_type> multiscale::verification::NumericStateVariableGrammar< Iterator >::numericStateVariableRule  
[private]

The rule for parsing a numeric state variable

Definition at line 36 of file NumericStateVariableGrammar.hpp.

```
6.128.4.2 template<typename Iterator> ScaleAndSubsystemStringGrammar<Iterator>
multiscale::verification::NumericStateVariableGrammar< Iterator
>::scaleAndSubsystemRule [private]
```

The grammar for parsing scales and subsystems

Definition at line 31 of file NumericStateVariableGrammar.hpp.

```
6.128.4.3 template<typename Iterator> qi::rule<Iterator, std::string(), qi::space_type>
multiscale::verification::NumericStateVariableGrammar< Iterator
>::stateVariableNameRule [private]
```

The rule for parsing the name of a state variable without escaping white space

Definition at line 40 of file NumericStateVariableGrammar.hpp.

```
6.128.4.4 template<typename Iterator> qi::rule<Iterator, StateVariableAttribute(),
qi::space_type> multiscale::verification::NumericStateVariableGrammar<
Iterator >::stateVariableRule [private]
```

The rule for parsing a state variable

Definition at line 38 of file NumericStateVariableGrammar.hpp.

```
6.128.4.5 template<typename Iterator> qi::rule<Iterator, ScaleAndSubsystemAttribute(),
qi::space_type> multiscale::verification::NumericStateVariableGrammar<
Iterator >::stateVariableScaleAndSubsystemRule [private]
```

The rule for parsing a state variable type

Definition at line 43 of file NumericStateVariableGrammar.hpp.

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

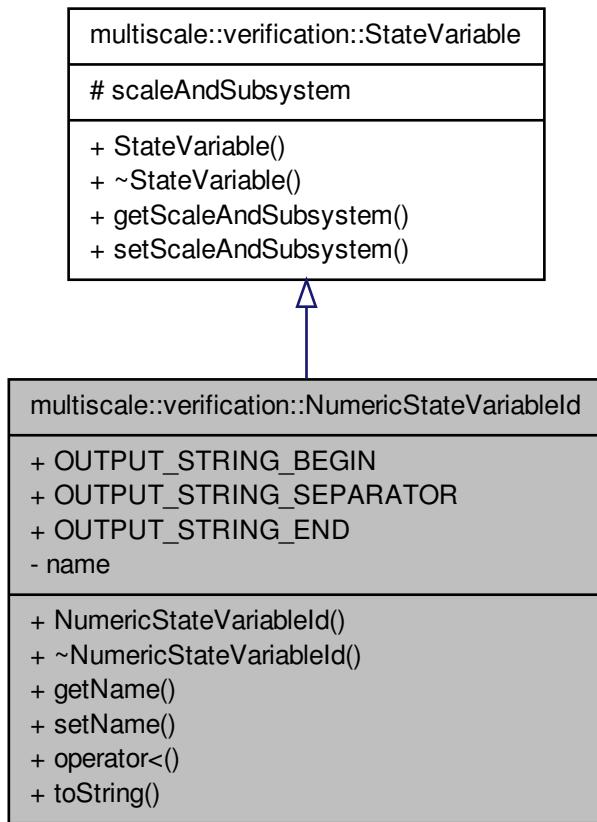
- NumericStateVariableGrammar.hpp
- NumericStateVariableGrammarDefinition.hpp

## 6.129 multiscale::verification::NumericStateVariableId Class Reference -

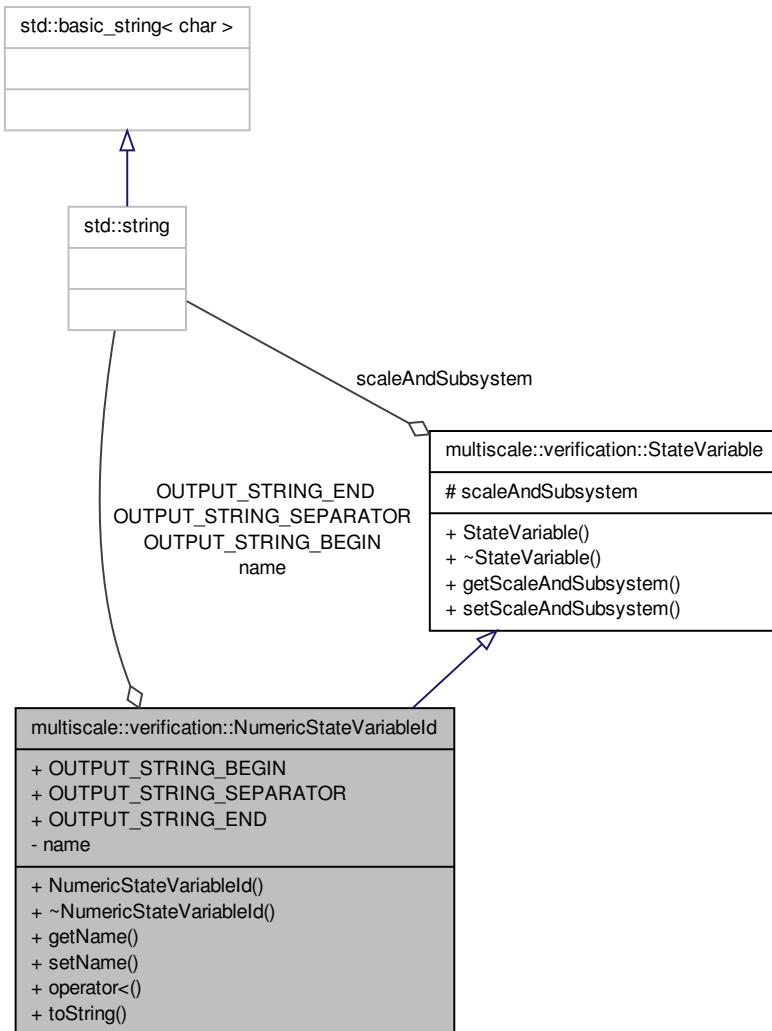
Class for representing the identity (name, type) of a numeric state variable.

```
#include <NumericStateVariableId.hpp>
```

Inheritance diagram for multiscale::verification::NumericStateVariableId:



Collaboration diagram for multiscale::verification::NumericStateVariableId:



## Public Member Functions

- `NumericStateVariableId (const std::string &name, const std::string &scaleAndSubsystem=ScaleAndSubsystem::DEFAULT_VALUE)`
- `~NumericStateVariableId ()`
- `const std::string & getName () const`

*Get the name of the numeric state variable.*

- void `setName` (const std::string &`name`)  
*Set the name of the numeric state variable.*
- bool `operator<` (const `NumericStateVariableId` &`rhs`) const  
*Overload the < operator.*
- std::string `toString` () const  
*Return the string representation of the numeric state variable identity.*

### Static Public Attributes

- static const std::string `OUTPUT_STRING_BEGIN` = "("
- static const std::string `OUTPUT_STRING_SEPARATOR` = ","
- static const std::string `OUTPUT_STRING_END` = ")"

### Private Attributes

- std::string `name`

#### 6.129.1 Detailed Description

Class for representing the identity (name, type) of a numeric state variable.

Definition at line 15 of file `NumericStateVariableId.hpp`.

#### 6.129.2 Constructor & Destructor Documentation

##### 6.129.2.1 `NumericStateVariableId::NumericStateVariableId` ( const std::string & `name`, const std::string & `scaleAndSubsystem` = `ScaleAndSubsystem::DEFAULT_VALUE` )

Definition at line 10 of file `NumericStateVariableId.cpp`.

References `multiscale::verification::StateVariable::scaleAndSubsystem`.

##### 6.129.2.2 `NumericStateVariableId::~NumericStateVariableId` ( )

Definition at line 16 of file `NumericStateVariableId.cpp`.

#### 6.129.3 Member Function Documentation

##### 6.129.3.1 `const std::string & NumericStateVariableId::getName` ( ) const

Get the name of the numeric state variable.

Definition at line 18 of file `NumericStateVariableId.cpp`.

References name.

Referenced by multiscale::verification::SpatialTemporalDataWriter::constructPropertyTreeFromNumericStateVariable().

**6.129.3.2 bool NumericStateVariableId::operator< ( const NumericStateVariableId & rhs ) const**

Overload the < operator.

#### Parameters

|            |                                                                                   |
|------------|-----------------------------------------------------------------------------------|
| <i>rhs</i> | The right hand side numeric state variable identity in the expression (lhs < rhs) |
|------------|-----------------------------------------------------------------------------------|

Definition at line 26 of file NumericStateVariableId.cpp.

References name, and multiscale::verification::StateVariable::scaleAndSubsystem.

**6.129.3.3 void NumericStateVariableId::setName ( const std::string & name )**

Set the name of the numeric state variable.

#### Parameters

|             |                                        |
|-------------|----------------------------------------|
| <i>name</i> | The name of the numeric state variable |
|-------------|----------------------------------------|

Definition at line 22 of file NumericStateVariableId.cpp.

References name.

**6.129.3.4 std::string NumericStateVariableId::toString ( ) const**

Return the string representation of the numeric state variable identity.

Definition at line 38 of file NumericStateVariableId.cpp.

References name, OUTPUT\_STRING\_BEGIN, OUTPUT\_STRING\_END, OUTPUT\_STRING\_SEPARATOR, and multiscale::verification::StateVariable::scaleAndSubsystem.

Referenced by multiscale::verification::MSTMLSubfilesMerger::addNumericStateVariableToTimepoint().

## 6.129.4 Member Data Documentation

**6.129.4.1 std::string multiscale::verification::NumericStateVariableId::name [private]**

The name of the numeric state variable

Definition at line 19 of file NumericStateVariableId.hpp.

Referenced by getName(), operator<(), setName(), and toString().

**6.129.4.2 const std::string NumericStateVariableId::OUTPUT\_STRING\_BEGIN = "("**  
[static]

Definition at line 49 of file NumericStateVariableId.hpp.

Referenced by toString().

**6.129.4.3 const std::string NumericStateVariableId::OUTPUT\_STRING\_END = ")"**  
[static]

Definition at line 51 of file NumericStateVariableId.hpp.

Referenced by toString().

**6.129.4.4 const std::string NumericStateVariableId::OUTPUT\_STRING\_SEPARATOR  
= "," [static]**

Definition at line 50 of file NumericStateVariableId.hpp.

Referenced by toString().

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

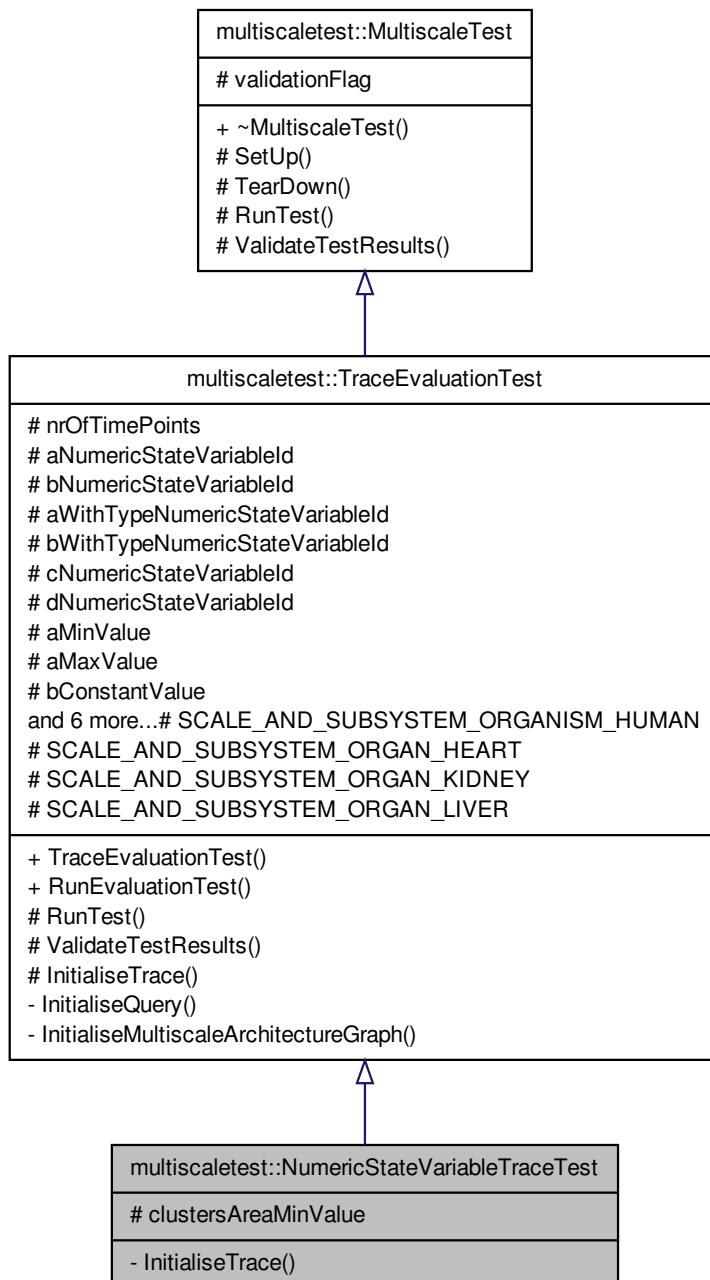
- NumericStateVariableId.hpp
- NumericStateVariableId.cpp

## **6.130 multiscaletest::NumericStateVariableTraceTest Class Reference**

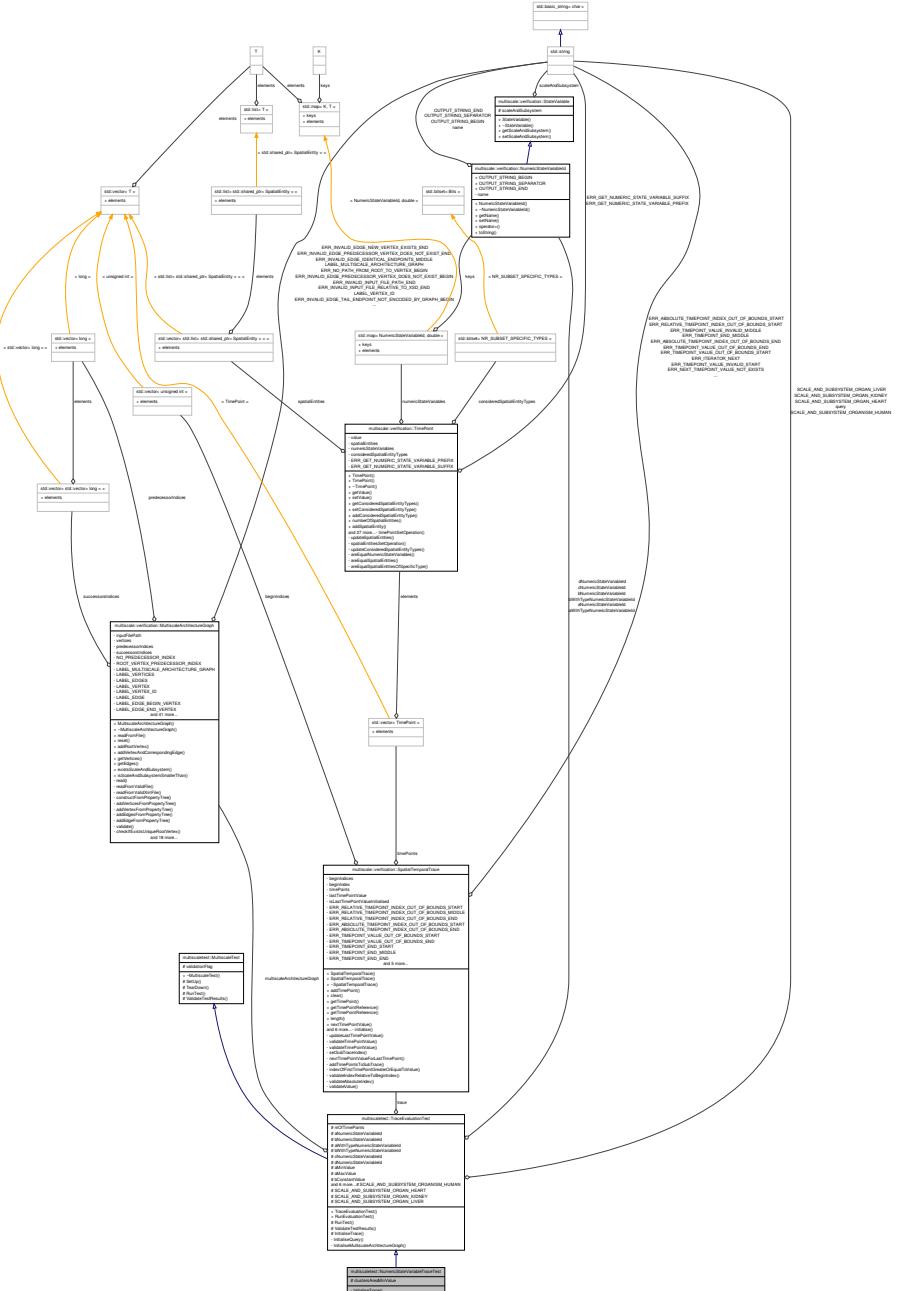
Class for testing evaluation of numeric state variable-only traces.

```
#include <NumericStateVariableTraceTest.hpp>
```

Inheritance diagram for multiscaletest::NumericStateVariableTraceTest:



## Collaboration diagram for multiscaletest::NumericStateVariableTraceTest:



## Protected Attributes

- double clustersAreaMinValue

### Private Member Functions

- virtual void [InitialiseTrace \(\)](#) override

*Initialise the trace.*

#### 6.130.1 Detailed Description

Class for testing evaluation of numeric state variable-only traces.

Definition at line 22 of file NumericStateVariableTraceTest.hpp.

#### 6.130.2 Member Function Documentation

##### 6.130.2.1 void multiscaletest::NumericStateVariableTraceTest::InitialiseTrace ( ) [override, private, virtual]

Initialise the trace.

Implements [multiscaletest::TraceEvaluationTest](#).

Definition at line 35 of file NumericStateVariableTraceTest.hpp.

#### 6.130.3 Member Data Documentation

##### 6.130.3.1 double multiscaletest::NumericStateVariableTraceTest::clustersAreaMinValue [protected]

The minimum area value for the cluster spatial entity type

Definition at line 26 of file NumericStateVariableTraceTest.hpp.

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

- [NumericStateVariableTraceTest.hpp](#)

## 6.131 multiscale::verification::NumericStatisticalMeasureAttribute Class Reference

Class for representing a numeric statistical measure attribute.

```
#include <NumericStatisticalMeasureAttribute.hpp>
```

### Public Attributes

- [NumericStatisticalMeasureType numericStatisticalMeasure](#)

### 6.131.1 Detailed Description

Class for representing a numeric statistical measure attribute.

Definition at line 24 of file NumericStatisticalMeasureAttribute.hpp.

### 6.131.2 Member Data Documentation

#### 6.131.2.1 NumericStatisticalMeasureType multiscale::verification::NumericStatisticalMeasureAttribute::numericStatisticalMeasure

The numeric statistical measure

Definition at line 28 of file NumericStatisticalMeasureAttribute.hpp.

Referenced by multiscale::verification::TemporalNumericVisitor::operator()().

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

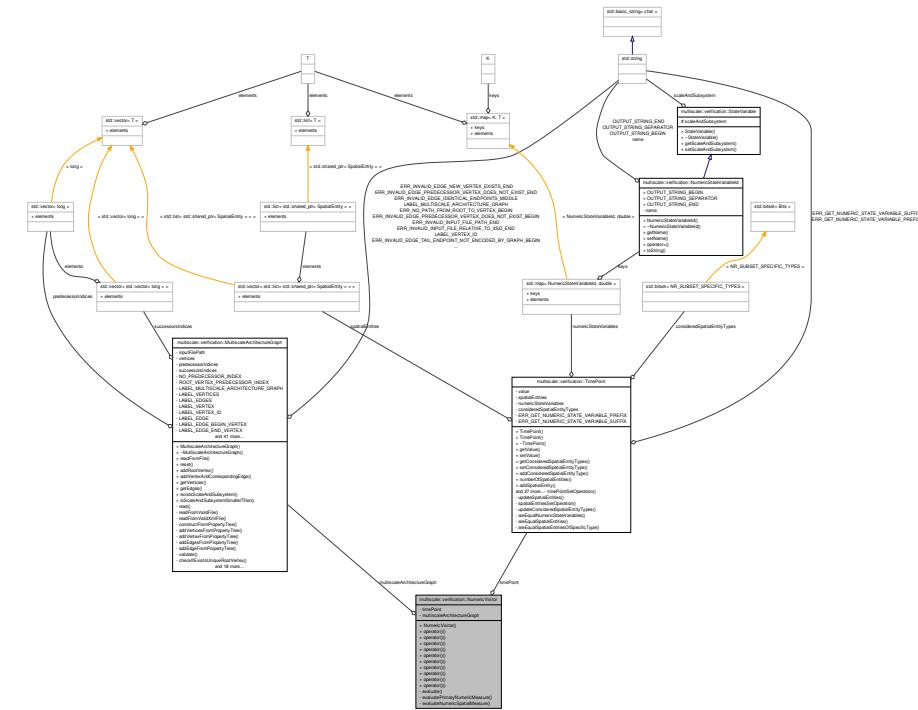
- NumericStatisticalMeasureAttribute.hpp

## 6.132 multiscale::verification::NumericVisitor Class Reference

Class for evaluating numeric measures.

```
#include <NumericVisitor.hpp>
```

## Collaboration diagram for multiscale::verification::NumericVisitor:



## Public Member Functions

- `NumericVisitor` (`TimePoint &timePoint, const MultiscaleArchitectureGraph &multiscaleArchitectureGraph`)
    - double `operator()` (`const NumericMeasureAttribute &numericMeasure`) const  
*Overloading the "()" operator for the `NumericMeasureAttribute` alternative.*
  - double `operator()` (`const PrimaryNumericMeasureAttribute &primaryNumericMeasure`) const  
*Overloading the "()" operator for the `PrimaryNumericMeasureAttribute` alternative.*
  - double `operator()` (`double realNumber`) const  
*Overloading the "()" operator for the real number alternative.*
  - double `operator()` (`const NumericStateVariableAttribute &numericStateVariable`) const  
*Overloading the "()" operator for the `NumericStateVariableAttribute` alternative.*
  - double `operator()` (`const NumericSpatialMeasureAttribute &numericSpatialMeasure`) const  
*Overloading the "()" operator for the `NumericSpatialAttribute` alternative.*
  - double `operator()` (`const UnaryNumericNumericAttribute &unaryNumericNumericMeasure`) const  
*Overloading the "()" operator for the `UnaryNumericNumericAttribute` alternative.*

- double `operator()` (const `BinaryNumericNumericAttribute` &`binaryNumericNumericMeasure`) const  
*Overloading the "(" operator for the `BinaryNumericNumericAttribute` alternative.*
- double `operator()` (const `UnaryStatisticalSpatialAttribute` &`unaryStatisticalSpatialAttribute`) const  
*Overloading the "(" operator for the `UnaryStatisticalSpatialAttribute` alternative.*
- double `operator()` (const `BinaryStatisticalSpatialAttribute` &`binaryStatisticalSpatialAttribute`) const  
*Overloading the "(" operator for the `BinaryStatisticalSpatialAttribute` alternative.*
- double `operator()` (const `BinaryStatisticalQuantileSpatialAttribute` &`binaryStatisticalQuantileSpatialAttribute`) const  
*Overloading the "(" operator for the `BinaryStatisticalQuantileSpatialAttribute` alternative.*

### Private Member Functions

- double `evaluate` (const `NumericMeasureType` &`numericMeasure`) const  
*Evaluate the given numeric measure considering the timePoint field.*
- double `evaluatePrimaryNumericMeasure` (const `PrimaryNumericMeasure-AttributeType` &`primaryNumericMeasure`) const  
*Evaluate the given primary numeric measure considering the timePoint field.*
- double `evaluateNumericSpatialMeasure` (const `NumericSpatialMeasureType` &`numericSpatialMeasure`) const  
*Evaluate the given numeric spatial measure considering the timePoint field.*

### Private Attributes

- `TimePoint` & `timePoint`
- const `MultiscaleArchitectureGraph` & `multiscaleArchitectureGraph`

#### 6.132.1 Detailed Description

Class for evaluating numeric measures.

Definition at line 20 of file NumericVisitor.hpp.

#### 6.132.2 Constructor & Destructor Documentation

##### 6.132.2.1 `multiscale::verification::NumericVisitor::NumericVisitor` ( `TimePoint` & `timePoint`, const `MultiscaleArchitectureGraph` & `multiscaleArchitectureGraph` ) [inline]

Definition at line 31 of file NumericVisitor.hpp.

Referenced by `evaluate()`, `evaluateNumericSpatialMeasure()`, and `evaluatePrimaryNumericMeasure()`.

### 6.132.3 Member Function Documentation

6.132.3.1 `double multiscale::verification::NumericVisitor::evaluate ( const NumericMeasureType & numericMeasure ) const [inline, private]`

Evaluate the given numeric measure considering the timePoint field.

#### Parameters

|                                               |                           |
|-----------------------------------------------|---------------------------|
| <code>numeric-</code><br><code>Measure</code> | The given numeric measure |
|-----------------------------------------------|---------------------------|

Definition at line 158 of file NumericVisitor.hpp.

References multiscaleArchitectureGraph, NumericVisitor(), and timePoint.

Referenced by operator()().

6.132.3.2 `double multiscale::verification::NumericVisitor::evaluateNumericSpatialMeasure ( const NumericSpatialMeasureType & numericSpatialMeasure ) const [inline, private]`

Evaluate the given numeric spatial measure considering the timePoint field.

#### Parameters

|                                                                        |                                   |
|------------------------------------------------------------------------|-----------------------------------|
| <code>numeric-</code><br><code>Spatial-</code><br><code>Measure</code> | The given numeric spatial measure |
|------------------------------------------------------------------------|-----------------------------------|

Definition at line 192 of file NumericVisitor.hpp.

References multiscaleArchitectureGraph, NumericVisitor(), and timePoint.

Referenced by operator()().

6.132.3.3 `double multiscale::verification::NumericVisitor::evaluatePrimaryNumericMeasure ( const PrimaryNumericMeasureAttributeType & primaryNumericMeasure ) const [inline, private]`

Evaluate the given primary numeric measure considering the timePoint field.

#### Parameters

|                                                                        |                                   |
|------------------------------------------------------------------------|-----------------------------------|
| <code>primary-</code><br><code>Numeric-</code><br><code>Measure</code> | The given primary numeric measure |
|------------------------------------------------------------------------|-----------------------------------|

Definition at line 175 of file NumericVisitor.hpp.

References multiscaleArchitectureGraph, NumericVisitor(), and timePoint.

Referenced by operator()().

**6.132.3.4 double multiscale::verification::NumericVisitor::operator()** ( **const NumericMeasureAttribute & numericMeasure** ) **const [inline]**

Overloading the "(" operator for the [NumericMeasureAttribute](#) alternative.

**Parameters**

|                             |                     |
|-----------------------------|---------------------|
| <b>numeric-<br/>Measure</b> | The numeric measure |
|-----------------------------|---------------------|

Definition at line 40 of file NumericVisitor.hpp.

References [evaluate\(\)](#), and [multiscale::verification::NumericMeasureAttribute::numericMeasure](#).

**6.132.3.5 double multiscale::verification::NumericVisitor::operator()** ( **const PrimaryNumericMeasureAttribute & primaryNumericMeasure** ) **const [inline]**

Overloading the "(" operator for the [PrimaryNumericMeasureAttribute](#) alternative.

**Parameters**

|                                          |                             |
|------------------------------------------|-----------------------------|
| <b>primary-<br/>Numeric-<br/>Measure</b> | The primary numeric measure |
|------------------------------------------|-----------------------------|

Definition at line 49 of file NumericVisitor.hpp.

References [evaluatePrimaryNumericMeasure\(\)](#), and [multiscale::verification::PrimaryNumericMeasureAttribute::primaryNumericMeasure](#).

**6.132.3.6 double multiscale::verification::NumericVisitor::operator()** ( **double realNumber** ) **const [inline]**

Overloading the "(" operator for the real number alternative.

**Parameters**

|                   |                 |
|-------------------|-----------------|
| <b>realNumber</b> | The real number |
|-------------------|-----------------|

Definition at line 58 of file NumericVisitor.hpp.

**6.132.3.7 double multiscale::verification::NumericVisitor::operator() ( const NumericStateVariableAttribute & *numericStateVariable* ) const [inline]**

Overloading the "(") operator for the [NumericStateVariableAttribute](#) alternative.

**Parameters**

|                                         |                            |
|-----------------------------------------|----------------------------|
| <i>numeric-<br/>State-<br/>Variable</i> | The numeric state variable |
|-----------------------------------------|----------------------------|

Definition at line 67 of file NumericVisitor.hpp.

References [multiscale::verification::NumericStateVariableEvaluator::evaluate\(\)](#), [multiscaleArchitectureGraph](#), and [timePoint](#).

**6.132.3.8 double multiscale::verification::NumericVisitor::operator() ( const NumericSpatialMeasureAttribute & *numericSpatialMeasure* ) const [inline]**

Overloading the "(") operator for the [NumericSpatialAttribute](#) alternative.

**Parameters**

|                                          |                                       |
|------------------------------------------|---------------------------------------|
| <i>numeric-<br/>Spatial-<br/>Measure</i> | The numeric spatial measure attribute |
|------------------------------------------|---------------------------------------|

Definition at line 83 of file NumericVisitor.hpp.

References [evaluateNumericSpatialMeasure\(\)](#), and [multiscale::verification::NumericSpatialMeasureAttribute::numericSpatialMeasure](#).

**6.132.3.9 double multiscale::verification::NumericVisitor::operator() ( const UnaryNumericNumericAttribute & *unaryNumericNumericMeasure* ) const [inline]**

Overloading the "(") operator for the [UnaryNumericNumericAttribute](#) alternative.

**Parameters**

|                                                     |                                   |
|-----------------------------------------------------|-----------------------------------|
| <i>unary-<br/>Numeric-<br/>Numeric-<br/>Measure</i> | The unary numeric numeric measure |
|-----------------------------------------------------|-----------------------------------|

Definition at line 92 of file NumericVisitor.hpp.

References [multiscale::verification::NumericEvaluator::evaluate\(\)](#), [evaluate\(\)](#), [multiscale::verification::UnaryNumericNumericAttribute::numericMeasure](#), [multiscale::verification-](#)

::UnaryNumericNumericAttribute::unaryNumericMeasure, and multiscale::verification::UnaryNumericMeasureAttribute::unaryNumericMeasureType.

```
6.132.3.10 double multiscale::verification::NumericVisitor::operator() ( const
    BinaryNumericNumericAttribute & binaryNumericNumericMeasure ) const
    [inline]
```

Overloading the "(" operator for the [BinaryNumericNumericAttribute](#) alternative.

#### Parameters

|                                                                        |                                    |
|------------------------------------------------------------------------|------------------------------------|
| <i>binary-</i><br><i>Numeric-</i><br><i>Numeric-</i><br><i>Measure</i> | The binary numeric numeric measure |
|------------------------------------------------------------------------|------------------------------------|

Definition at line 111 of file NumericVisitor.hpp.

References multiscale::verification::BinaryNumericNumericAttribute::binaryNumericMeasure, multiscale::verification::BinaryNumericMeasureAttribute::binaryNumericMeasureType, multiscale::verification::NumericEvaluator::evaluate(), evaluate(), multiscale::verification::BinaryNumericNumericAttribute::firstNumericMeasure, and multiscale::verification::BinaryNumericNumericAttribute::secondNumericMeasure.

```
6.132.3.11 double multiscale::verification::NumericVisitor::operator() ( const
    UnaryStatisticalSpatialAttribute & unaryStatisticalSpatialAttribute ) const
    [inline]
```

Overloading the "(" operator for the [UnaryStatisticalSpatialAttribute](#) alternative.

#### Parameters

|                                                                             |                                         |
|-----------------------------------------------------------------------------|-----------------------------------------|
| <i>unary-</i><br><i>Statistical-</i><br><i>Spatial-</i><br><i>Attribute</i> | The unary statistical spatial attribute |
|-----------------------------------------------------------------------------|-----------------------------------------|

Definition at line 220 of file NumericVisitor.hpp.

References multiscale::verification::NumericEvaluator::evaluate(), multiscale::verification::NumericMeasureCollectionEvaluator::evaluateSpatialMeasureCollection(), multiscale::ArchitectureGraph, multiscale::verification::UnaryStatisticalSpatialAttribute::spatialMeasureCollection, timePoint, multiscale::verification::UnaryStatisticalSpatialAttribute::unaryStatisticalMeasure, and multiscale::verification::UnaryStatisticalSpatialAttribute::unaryStatisticalMeasureType.

```
6.132.3.12 double multiscale::verification::NumericVisitor::operator() ( const
    BinaryStatisticalSpatialAttribute & binaryStatisticalSpatialAttribute ) const
    [inline]
```

Overloading the "(") operator for the [BinaryStatisticalSpatialAttribute](#) alternative.

#### Parameters

|                                                            |                                          |
|------------------------------------------------------------|------------------------------------------|
| <i>binary-<br/>Statistical-<br/>Spatial-<br/>Attribute</i> | The binary statistical spatial attribute |
|------------------------------------------------------------|------------------------------------------|

Definition at line 238 of file NumericVisitor.hpp.

References [multiscale::verification::BinaryStatisticalSpatialAttribute::binaryStatisticalMeasure](#), [multiscale::verification::BinaryStatisticalMeasureAttribute::binaryStatisticalMeasureType](#), [multiscale::verification::NumericEvaluator::evaluate\(\)](#), [multiscale::verification::NumericMeasureCollectionEvaluator::evaluateSpatialMeasureCollection\(\)](#), [multiscale::verification::BinaryStatisticalSpatialAttribute::firstSpatialMeasureCollection](#), and [multiscale::verification::BinaryStatisticalSpatialAttribute::secondSpatialMeasureCollection](#).

```
6.132.3.13 double multiscale::verification::NumericVisitor::operator() ( const Binary-
    StatisticalQuantileSpatialAttribute & binaryStatisticalQuantileSpatialAttribute
    ) const [inline]
```

Overloading the "(") operator for the [BinaryStatisticalQuantileSpatialAttribute](#) alternative.

#### Parameters

|                                                                          |                                                   |
|--------------------------------------------------------------------------|---------------------------------------------------|
| <i>binary-<br/>Statistical-<br/>Quantile-<br/>Spatial-<br/>Attribute</i> | The binary statistical quantile spatial attribute |
|--------------------------------------------------------------------------|---------------------------------------------------|

Definition at line 263 of file NumericVisitor.hpp.

References [multiscale::verification::BinaryStatisticalQuantileSpatialAttribute::binaryStatisticalQuantileMeasure](#), [multiscale::verification::NumericEvaluator::evaluate\(\)](#), [multiscale::verification::NumericMeasureCollectionEvaluator::evaluateSpatialMeasureCollection\(\)](#), [multiscale::verification::BinaryStatisticalQuantileSpatialAttribute::parameter](#), and [multiscale::verification::BinaryStatisticalQuantileSpatialAttribute::spatialMeasureCollection](#).

## 6.132.4 Member Data Documentation

**6.132.4.1 const MultiscaleArchitectureGraph& multiscale-  
::verification::NumericVisitor::multiscaleArchitectureGraph  
[private]**

The considered multiscale architecture graph

Definition at line 27 of file NumericVisitor.hpp.

Referenced by evaluate(), evaluateNumericSpatialMeasure(), evaluatePrimary-NumericMeasure(), and operator()().

**6.132.4.2 TimePoint& multiscale::verification::NumericVisitor::timePoint  
[private]**

The considered timepoint

Definition at line 25 of file NumericVisitor.hpp.

Referenced by evaluate(), evaluateNumericSpatialMeasure(), evaluatePrimary-NumericMeasure(), and operator()().

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

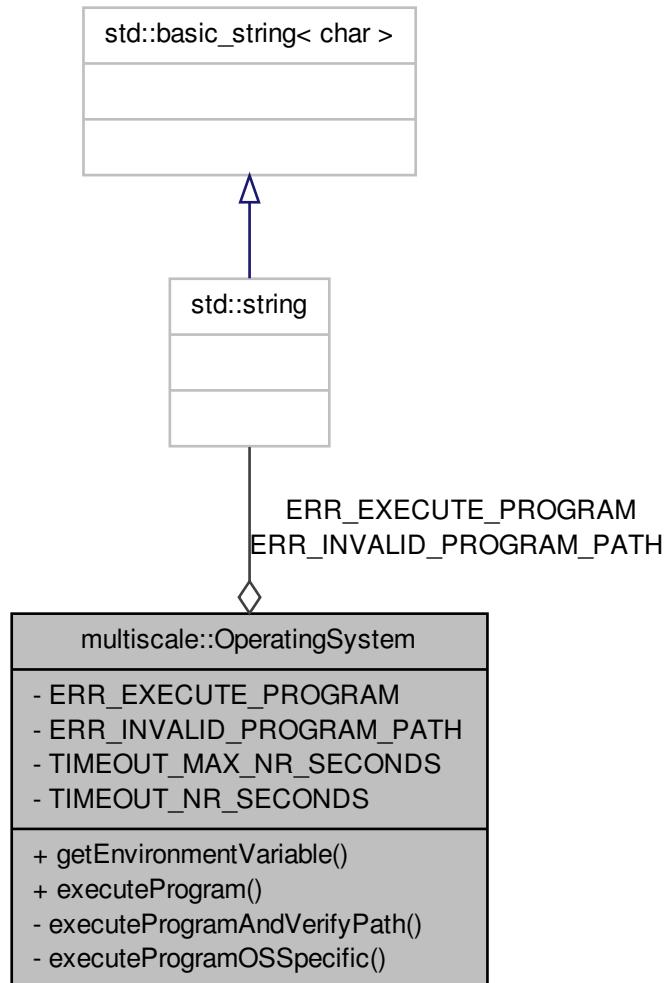
- NumericVisitor.hpp

## 6.133 multiscale::OperatingSystem Class Reference

Class for executing operating system related functions.

```
#include <OperatingSystem.hpp>
```

Collaboration diagram for multiscale::OperatingSystem:



## Static Public Member Functions

- static std::string [getEnvironmentVariable](#) (const std::string &name)  
*Get the value of the environment variable having the given name.*
- static void [executeProgram](#) (const std::string &path)  
*Create a child process and execute the program with the given path.*

## Static Private Member Functions

- static void [executeProgramAndVerifyPath](#) (const std::string &path)  
*Create a child process and execute the program with the given path if the provided path is valid.*
- static void [executeProgramOSSpecific](#) (const std::string &path)  
*Create a child process and execute the program with the given path considering the specific Operating system.*

## Static Private Attributes

- static const std::string [ERR\\_EXECUTE\\_PROGRAM](#) = "The process executing the program located at the following path could not be created: "
- static const std::string [ERR\\_INVALID\\_PROGRAM\\_PATH](#) = "The process was not created because the provided program path is invalid: "
- static const unsigned int [TIMEOUT\\_MAX\\_NR\\_SECONDS](#) = 100
- static const unsigned int [TIMEOUT\\_NR\\_SECONDS](#) = 1

### 6.133.1 Detailed Description

Class for executing operating system related functions.

Definition at line 23 of file OperatingSystem.hpp.

### 6.133.2 Member Function Documentation

#### 6.133.2.1 void OperatingSystem::executeProgram ( const std::string & path ) [static]

Create a child process and execute the program with the given path.

##### Parameters

|             |                                                |
|-------------|------------------------------------------------|
| <i>path</i> | The path to the program which will be executed |
|-------------|------------------------------------------------|

Definition at line 24 of file OperatingSystem.cpp.

References [executeProgramAndVerifyPath\(\)](#), [multiscale::ExceptionHandler::printDetailedErrorMessage\(\)](#), [multiscale::ConsolePrinter::printWarningMessage\(\)](#), and [multiscale::MultiscaleException::rawMessage\(\)](#).

Referenced by [multiscale::verification::ModelCheckingManager::executeExtraEvaluationProgramAndPrintMessage\(\)](#).

6.133.2.2 void OperatingSystem::executeProgramAndVerifyPath ( const std::string & *path* ) [static, private]

Create a child process and execute the program with the given path if the provided path is valid.

Parameters

|             |                                                |
|-------------|------------------------------------------------|
| <i>path</i> | The path to the program which will be executed |
|-------------|------------------------------------------------|

Definition at line 34 of file OperatingSystem.cpp.

References ERR\_INVALID\_PROGRAM\_PATH, executeProgramOSSpecific(), and multiscale::Filesystem::isValidFilePath().

Referenced by executeProgram().

6.133.2.3 static void multiscale::OperatingSystem::executeProgramOSSpecific ( const std::string & *path* ) [static, private]

Create a child process and execute the program with the given path considering the specific Operating system.

Parameters

|             |                                                |
|-------------|------------------------------------------------|
| <i>path</i> | The path to the program which will be executed |
|-------------|------------------------------------------------|

Referenced by executeProgramAndVerifyPath().

6.133.2.4 std::string OperatingSystem::getEnvironmentVariable ( const std::string & *name* ) [static]

Get the value of the environment variable having the given name.

Parameters

|             |                                      |
|-------------|--------------------------------------|
| <i>name</i> | The name of the environment variable |
|-------------|--------------------------------------|

Definition at line 14 of file OperatingSystem.cpp.

Referenced by multiscale::ConsolePrinter::terminalSupportsColour().

### 6.133.3 Member Data Documentation

6.133.3.1 const std::string OperatingSystem::ERR\_EXECUTE\_PROGRAM = "The process executing the program located at the following path could not be created: " [static, private]

Definition at line 114 of file OperatingSystem.hpp.

```
6.133.3.2 const std::string OperatingSystem::ERR_INVALID_PROGRAM_PATH  
= "The process was not created because the provided program path is invalid: "  
[static, private]
```

Definition at line 115 of file OperatingSystem.hpp.

Referenced by executeProgramAndVerifyPath().

```
6.133.3.3 const unsigned int OperatingSystem::TIMEOUT_MAX_NR_SECONDS = 100  
[static, private]
```

Definition at line 117 of file OperatingSystem.hpp.

```
6.133.3.4 const unsigned int OperatingSystem::TIMEOUT_NR_SECONDS = 1  
[static, private]
```

Definition at line 118 of file OperatingSystem.hpp.

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

- OperatingSystem.hpp
- OperatingSystem.cpp

## 6.134 multiscale::verification::OrConstraintAttribute Class Reference

Class for representing an "or" constraint attribute.

```
#include <OrConstraintAttribute.hpp>
```

### Public Attributes

- [ConstraintAttributeType constraint](#)

#### 6.134.1 Detailed Description

Class for representing an "or" constraint attribute.

Definition at line 14 of file OrConstraintAttribute.hpp.

#### 6.134.2 Member Data Documentation

##### 6.134.2.1 ConstraintAttributeType multiscale::verification::OrConstraintAttribute- ::constraint

The constraint following the "or" operator

Definition at line 18 of file OrConstraintAttribute.hpp.

Referenced by multiscale::verification::ConstraintVisitor::operator()().

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

- [OrConstraintAttribute.hpp](#)

## **6.135 multiscale::verification::OrLogicPropertyAttribute Class Reference**

Class for representing an "or" logic property attribute.

```
#include <OrLogicPropertyAttribute.hpp>
```

### **Public Attributes**

- [LogicPropertyAttributeType logicProperty](#)

#### **6.135.1 Detailed Description**

Class for representing an "or" logic property attribute.

Definition at line 14 of file OrLogicPropertyAttribute.hpp.

#### **6.135.2 Member Data Documentation**

##### **6.135.2.1 LogicPropertyAttributeType multiscale::verification::OrLogicPropertyAttribute::logicProperty**

The logical property following the "or" operator

Definition at line 18 of file OrLogicPropertyAttribute.hpp.

Referenced by multiscale::verification::LogicPropertyVisitor::operator()().

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

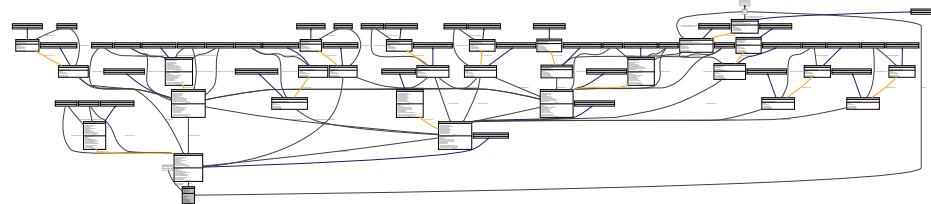
- [OrLogicPropertyAttribute.hpp](#)

## **6.136 multiscale::verification::Parser Class Reference**

Class used for parsing (P)BLSTL logical queries.

```
#include <Parser.hpp>
```

Collaboration diagram for multiscale::verification::Parser:



## Public Member Functions

- `Parser (const std::string &logicalQuery)`
- `~Parser ()`
- `void setLogicalQuery (const std::string &logicalQuery)`  
*Set the value of the logical query.*
- `bool parse (AbstractSyntaxTree &parseResult)`  
*Parse the logical query.*

## Private Member Functions

- `void initialise ()`  
*Initialisation function.*
- `bool parseLogicalQuery (AbstractSyntaxTree &parseResult)`  
*Parse the logical query and wrap the ProbabilisticLogicProperty into an AbstractSyntaxTree instance.*
- `bool parseLogicalQuery (ProbabilisticLogicPropertyAttribute &parseResult)`  
*Parse the logical query and construct the abstract syntax tree.*
- `void checkIfErrorCase (bool isSuccessfulParse)`  
*Check if an error case was encountered.*
- `bool isStringParsedCompletely ()`  
*Check if the string was parsed completely.*

## Private Attributes

- `std::string logicalQuery`
- `std::string::const_iterator logicalQueryIterator`
- `std::string::const_iterator logicalQueryEnd`
- `LogicPropertyGrammar < std::string::const_iterator > grammar`

### 6.136.1 Detailed Description

Class used for parsing (P)BLSTL logical queries.

Definition at line 17 of file Parser.hpp.

## 6.136.2 Constructor & Destructor Documentation

### 6.136.2.1 Parser::Parser ( const std::string & *logicalQuery* )

Definition at line 11 of file Parser.cpp.

References initialise(), and logicalQuery.

### 6.136.2.2 Parser::~Parser ( )

Definition at line 17 of file Parser.cpp.

## 6.136.3 Member Function Documentation

### 6.136.3.1 void Parser::checkIfErrorCase ( bool *isSuccessfulParse* ) [private]

Check if an error case was encountered.

#### Parameters

|                            |                                 |
|----------------------------|---------------------------------|
| <i>is-Successful-Parse</i> | The parse was successful or not |
|----------------------------|---------------------------------|

Definition at line 82 of file Parser.cpp.

References isStringParsedCompletely(), logicalQueryEnd, and logicalQueryIterator.

Referenced by parseLogicalQuery().

### 6.136.3.2 void Parser::initialise ( ) [private]

Initialisation function.

Definition at line 57 of file Parser.cpp.

References logicalQuery, logicalQueryEnd, and logicalQueryIterator.

Referenced by Parser(), and setLogicalQuery().

### 6.136.3.3 bool Parser::isStringParsedCompletely ( ) [private]

Check if the string was parsed completely.

Definition at line 92 of file Parser.cpp.

References logicalQueryEnd, and logicalQueryIterator.

Referenced by checkIfErrorCase().

#### 6.136.3.4 bool Parser::parse ( AbstractSyntaxTree & parseResult )

Parse the logical query.

##### Parameters

|                    |                                     |
|--------------------|-------------------------------------|
| <i>parseResult</i> | The result of the parsing procedure |
|--------------------|-------------------------------------|

Definition at line 25 of file Parser.cpp.

References multiscale::verification::ParserGrammarExtraInputException::getErrorString(), multiscale::verification::ParserGrammarUnparseableInputException::getErrorString(), multiscale::verification::ParserGrammarProbabilityException::getErrorString(), multiscale::verification::ParserGrammarUnexpectedTokenException::getErrorString(), multiscale::verification::ParserGrammarProbabilityException::getExpectedToken(), multiscale::verification::ParserGrammarUnexpectedTokenException::getExpectedToken(), multiscale::verification::ParserGrammarExceptionHandler::handleExtraInputException(), multiscale::verification::ParserGrammarExceptionHandler::handleProbabilityException(), multiscale::verification::ParserGrammarExceptionHandler::handleUnexpectedTokenException(), multiscale::verification::ParserGrammarExceptionHandler::handleUnparseableInputException(), logicalQuery, logicalQueryEnd, logicalQueryIterator, and parseLogicalQuery().

Referenced by multiscaletest::ModelCheckerTest::InitialiseAbstractSyntaxTree(), multiscale::verification::ModelCheckingManager::isValidLogicProperty(), multiscaletest::verification::parseInputString(), and multiscaletest::TraceEvaluationTest::RunTest().

#### 6.136.3.5 bool Parser::parseLogicalQuery ( AbstractSyntaxTree & parseResult ) [private]

Parse the logical query and wrap the ProbabilisticLogicProperty into an [AbstractSyntaxTree](#) instance.

##### Parameters

|                    |                                     |
|--------------------|-------------------------------------|
| <i>parseResult</i> | The result of the parsing procedure |
|--------------------|-------------------------------------|

Definition at line 62 of file Parser.cpp.

References multiscale::verification::AbstractSyntaxTree::initialiseTree().

Referenced by parse().

#### 6.136.3.6 bool Parser::parseLogicalQuery ( ProbabilisticLogicPropertyAttribute & parseResult ) [private]

Parse the logical query and construct the abstract syntax tree.

**Parameters**

|                          |                                     |
|--------------------------|-------------------------------------|
| <code>parseResult</code> | The result of the parsing procedure |
|--------------------------|-------------------------------------|

Definition at line 72 of file Parser.cpp.

References checkIfErrorCase(), logicalQueryEnd, and logicalQueryIterator.

**6.136.3.7 void Parser::setLogicalQuery ( const std::string & *logicalQuery* )**

Set the value of the logical query.

Definition at line 19 of file Parser.cpp.

References initialise(), and logicalQuery.

Referenced by multiscale::verification::ModelCheckingManager::isValidLogicProperty().

**6.136.4 Member Data Documentation****6.136.4.1 LogicPropertyGrammar<std::string::const\_iterator>  
multiscale::verification::Parser::grammar [private]**

The grammar used for parsing logic properties

Definition at line 27 of file Parser.hpp.

**6.136.4.2 std::string multiscale::verification::Parser::logicalQuery [private]**

The logical query to be parsed

Definition at line 21 of file Parser.hpp.

Referenced by initialise(), parse(), Parser(), and setLogicalQuery().

**6.136.4.3 std::string::const\_iterator multiscale::verification::Parser::logicalQueryEnd  
[private]**

Iterator pointing at the end of the logical query

Definition at line 24 of file Parser.hpp.

Referenced by checkIfErrorCase(), initialise(), isStringParsedCompletely(), parse(), and parseLogicalQuery().

**6.136.4.4 std::string::const\_iterator multiscale::verification::Parser::logicalQuery-  
Iterator [private]**

Iterator of the logical query

Definition at line 23 of file Parser.hpp.

Referenced by checkIfErrorCase(), initialise(), isStringParsedCompletely(), parse(), and parseLogicalQuery().

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

- Parser.hpp
- Parser.cpp

## 6.137 multiscale::verification::ParserGrammarExceptionHandler Class Reference

Class for handling parser grammar exceptions.

```
#include <ParserGrammarExceptionHandler.hpp>
```

### Static Public Member Functions

- static void **handleUnexpectedTokenException** (const std::string &initialString, const std::string &errorString, const std::string &expectedToken)  
*Handle the exception when an unexpected token was encountered.*
- static void **handleProbabilityException** (const std::string &initialString, const std::string &errorString, const std::string &expectedToken)  
*Handle the exception when an invalid probability was encountered.*
- static void **handleUnparseableInputException** (const std::string &initialString, const std::string &errorString)  
*Handle the exception when wrong input is provided.*
- static void **handleExtraInputException** (const std::string &initialString, const std::string &extraInput)  
*Handle the exception when extra input is provided.*

### Static Private Member Functions

- static std::string **handleUnexpectedTokenInString** (const std::string &initialString, const std::string &errorString, const std::string &expectedToken)  
*Handle the case where an unexpected token was found in the string.*
- static std::string **handleExpectedTokenAtEndOfString** (const std::string &initialString, const std::string &expectedToken)  
*Handle the case where an expected token was not encountered at the end of the string.*
- static std::string **trimRight** (const std::string &inputString)  
*Remove the trailing "new line" characters from the end of the string.*
- static std::string **getIntroductoryErrorMessage** ()  
*Return the generic introductory error message.*

### 6.137.1 Detailed Description

Class for handling parser grammar exceptions.

Definition at line 16 of file ParserGrammarExceptionHandler.hpp.

### 6.137.2 Member Function Documentation

#### 6.137.2.1 std::string ParserGrammarExceptionHandler::getIntroductoryErrorMessage( ) [static, private]

Return the generic introductory error message.

Definition at line 125 of file ParserGrammarExceptionHandler.cpp.

#### 6.137.2.2 std::string ParserGrammarExceptionHandler::handleExpectedTokenAtEndOfString( const std::string & initialString, const std::string & expectedToken ) [static, private]

Handle the case where an expected token was not encountered at the end of the string.

##### Parameters

|                            |                                                |
|----------------------------|------------------------------------------------|
| <i>initialString</i>       | The initial string                             |
| <i>expected-<br/>Token</i> | The token which should replace the error token |

Definition at line 107 of file ParserGrammarExceptionHandler.cpp.

#### 6.137.2.3 void ParserGrammarExceptionHandler::handleExtraInputException( const std::string & initialString, const std::string & extraInput ) [static]

Handle the exception when extra input is provided.

##### Parameters

|                      |                    |
|----------------------|--------------------|
| <i>initialString</i> | The initial string |
| <i>extraInput</i>    | Extra input        |

Definition at line 65 of file ParserGrammarExceptionHandler.cpp.

Referenced by multiscale::verification::Parser::parse().

#### 6.137.2.4 void ParserGrammarExceptionHandler::handleProbabilityException( const std::string & initialString, const std::string & errorString, const std::string & expectedToken ) [static]

Handle the exception when an invalid probability was encountered.

**Parameters**

|                      |                                                                        |
|----------------------|------------------------------------------------------------------------|
| <i>initialString</i> | The initial string                                                     |
| <i>errorString</i>   | A substring of the initial string which starts from the error position |
| <i>expectedToken</i> | The token which should replace the error token                         |

Definition at line 27 of file ParserGrammarExceptionHandler.cpp.

Referenced by multiscale::verification::Parser::parse().

```
6.137.2.5 void ParserGrammarExceptionHandler::handleUnexpectedToken-
Exception ( const std::string & initialString, const std::string & errorString, const
std::string & expectedToken ) [static]
```

Handle the exception when an unexpected token was encountered.

**Parameters**

|                      |                                                                        |
|----------------------|------------------------------------------------------------------------|
| <i>initialString</i> | The initial string                                                     |
| <i>errorString</i>   | A substring of the initial string which starts from the error position |
| <i>expectedToken</i> | The token which should replace the error token                         |

Definition at line 13 of file ParserGrammarExceptionHandler.cpp.

Referenced by multiscale::verification::Parser::parse().

```
6.137.2.6 std::string ParserGrammarExceptionHandler::handleUnexpectedToken-
InString ( const std::string & initialString, const std::string & errorString, const
std::string & expectedToken ) [static, private]
```

Handle the case where an unexpected token was found in the string.

**Parameters**

|                      |                                                                        |
|----------------------|------------------------------------------------------------------------|
| <i>initialString</i> | The initial string                                                     |
| <i>errorString</i>   | A substring of the initial string which starts from the error position |
| <i>expectedToken</i> | The token which should replace the error token                         |

Definition at line 84 of file ParserGrammarExceptionHandler.cpp.

```
6.137.2.7 void ParserGrammarExceptionHandler::handleUnparseableInput-
Exception ( const std::string & initialString, const std::string & errorString )
[static]
```

Handle the exception when wrong input is provided.

## 6.138 multiscale::verification::ParserGrammarExtraInputException Class

### Reference

767

#### Parameters

|                      |                    |
|----------------------|--------------------|
| <i>initialString</i> | The initial string |
| <i>errorString</i>   | Error string       |

Definition at line 47 of file ParserGrammarExceptionHandler.cpp.

Referenced by multiscale::verification::Parser::parse().

### 6.137.2.8 std::string ParserGrammarExceptionHandler::trimRight ( const std::string & *inputString* ) [static, private]

Remove the trailing "new line" characters from the end of the string.

#### Parameters

|                    |                        |
|--------------------|------------------------|
| <i>inputString</i> | The given input string |
|--------------------|------------------------|

Definition at line 121 of file ParserGrammarExceptionHandler.cpp.

References multiscale::StringManipulator::trimRight().

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

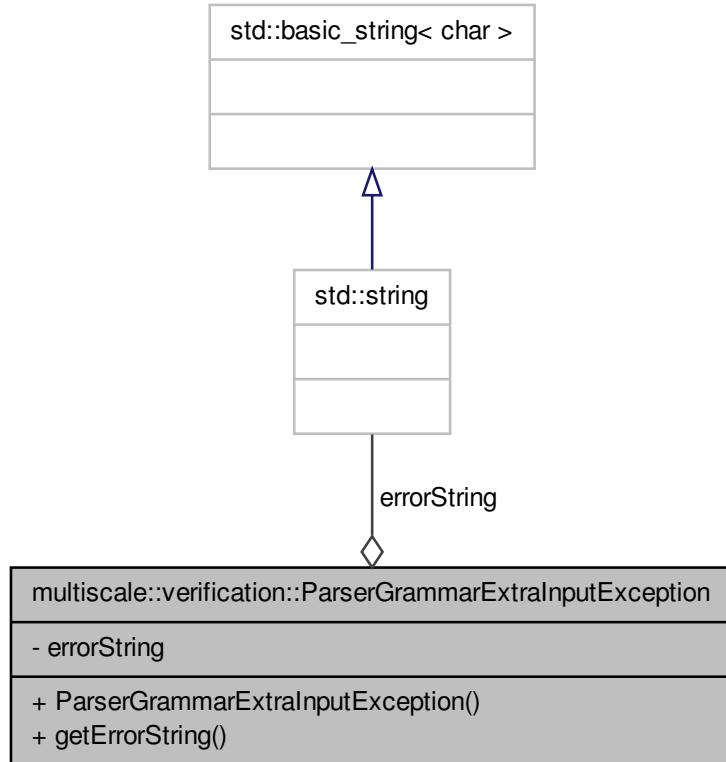
- ParserGrammarExceptionHandler.hpp
- ParserGrammarExceptionHandler.cpp

## 6.138 multiscale::verification::ParserGrammarExtraInputException Class Reference

Class for representing "extra input" exceptions in the parsing process.

```
#include <ParserGrammarExtraInputException.hpp>
```

Collaboration diagram for multiscale::verification::ParserGrammarExtraInputException:



### Public Member Functions

- `ParserGrammarExtraInputException (const std::string &errorString)`
- `std::string getErrorString () const`  
*Get the error string.*

### Private Attributes

- `std::string errorString`

#### 6.138.1 Detailed Description

Class for representing "extra input" exceptions in the parsing process.

Definition at line 14 of file ParserGrammarExtraInputException.hpp.

## 6.138.2 Constructor & Destructor Documentation

6.138.2.1 **multiscale::verification::ParserGrammarExtraInputException::-ParserGrammarExtraInputException ( const std::string & errorString )**  
[inline]

Definition at line 23 of file ParserGrammarExtraInputException.hpp.

## 6.138.3 Member Function Documentation

6.138.3.1 **std::string multiscale::verification::ParserGrammarExtraInputException::getErrorString ( ) const**  
[inline]

Get the error string.

Definition at line 28 of file ParserGrammarExtraInputException.hpp.

Referenced by multiscale::verification::Parser::parse().

## 6.138.4 Member Data Documentation

6.138.4.1 **std::string multiscale::verification::ParserGrammarExtraInputException::errorString [private]**

The substring from the original string starting with the index of the error token

Definition at line 18 of file ParserGrammarExtraInputException.hpp.

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

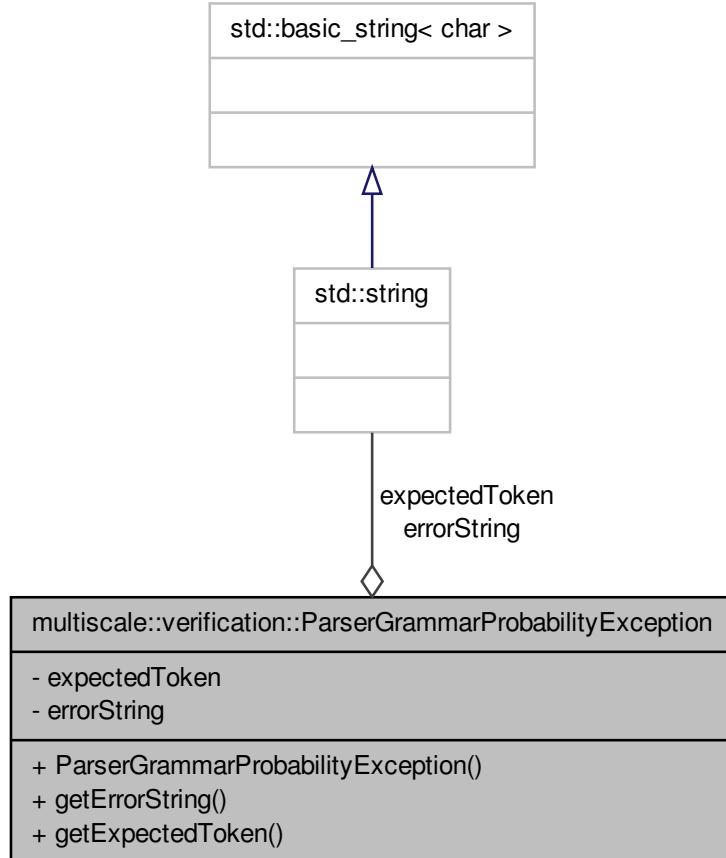
- ParserGrammarExtraInputException.hpp

## 6.139 multiscale::verification::ParserGrammarProbabilityException Class Reference

Class for representing "probability" exceptions in the parsing process.

```
#include <ParserGrammarProbabilityException.hpp>
```

Collaboration diagram for multiscale::verification::ParserGrammarProbabilityException:



## Public Member Functions

- `ParserGrammarProbabilityException` (const `std::string` &`expectedToken`, const `std::string` &`errorString`)
- `std::string getErrorString () const`  
*Get the error string.*
- `std::string getExpectedToken () const`  
*Get the expected token.*

**Private Attributes**

- std::string [expectedToken](#)
- std::string [errorString](#)

**6.139.1 Detailed Description**

Class for representing "probability" exceptions in the parsing process.

Definition at line 12 of file ParserGrammarProbabilityException.hpp.

**6.139.2 Constructor & Destructor Documentation****6.139.2.1 multiscale::verification::ParserGrammarProbabilityException::ParserGrammarProbabilityException ( const std::string & *expectedToken*, const std::string & *errorString* ) [inline]**

Definition at line 22 of file ParserGrammarProbabilityException.hpp.

References [errorString](#), and [expectedToken](#).

**6.139.3 Member Function Documentation****6.139.3.1 std::string multiscale::verification::ParserGrammarProbabilityException::getErrorString ( ) const [inline]**

Get the error string.

Definition at line 29 of file ParserGrammarProbabilityException.hpp.

References [errorString](#).

Referenced by [multiscale::verification::Parser::parse\(\)](#).

**6.139.3.2 std::string multiscale::verification::ParserGrammarProbabilityException::getExpectedToken ( ) const [inline]**

Get the expected token.

Definition at line 34 of file ParserGrammarProbabilityException.hpp.

References [expectedToken](#).

Referenced by [multiscale::verification::Parser::parse\(\)](#).

**6.139.4 Member Data Documentation**

**6.139.4.1 std::string multiscale::verification::ParserGrammarProbabilityException-  
::errorString [private]**

The substring from the original string starting with the index of the error token

Definition at line 17 of file ParserGrammarProbabilityException.hpp.

Referenced by getErrorString(), and ParserGrammarProbabilityException().

**6.139.4.2 std::string multiscale::verification::ParserGrammarProbabilityException-  
::expectedToken [private]**

The token which was expected and was not found during parsing

Definition at line 16 of file ParserGrammarProbabilityException.hpp.

Referenced by getExpectedToken(), and ParserGrammarProbabilityException().

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

- ParserGrammarProbabilityException.hpp

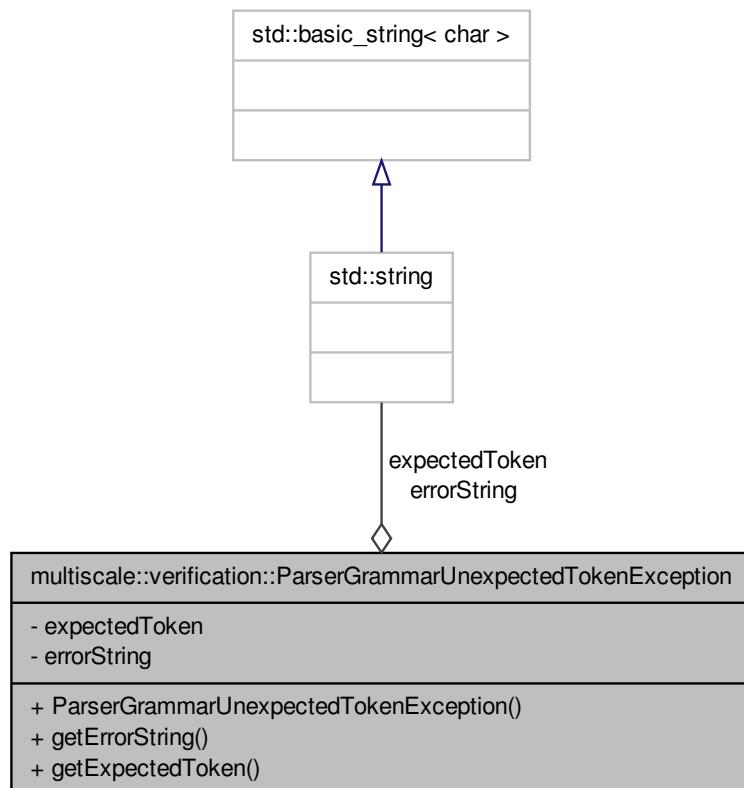
## **6.140 multiscale::verification::ParserGrammarUnexpectedToken- Exception Class Reference**

Class for representing "unexpected token" exceptions in the parsing process.

```
#include <ParserGrammarUnexpectedTokenException.hpp>
```

Collaboration diagram for multiscale::verification::ParserGrammarUnexpectedToken-

Exception:



## Public Member Functions

- `ParserGrammarUnexpectedTokenException` (const `std::string &expectedToken,`  
  `const std::string &errorString`)
- `std::string getErrorString () const`  
    *Get the error string.*
- `std::string getExpectedToken () const`  
    *Get the expected token.*

## Private Attributes

- `std::string expectedToken`
- `std::string errorString`

### 6.140.1 Detailed Description

Class for representing "unexpected token" exceptions in the parsing process.

Definition at line 12 of file ParserGrammarUnexpectedTokenException.hpp.

### 6.140.2 Constructor & Destructor Documentation

6.140.2.1 `multiscale::verification::ParserGrammarUnexpectedTokenException-  
::ParserGrammarUnexpectedTokenException ( const std::string &  
expectedToken, const std::string & errorString ) [inline]`

Definition at line 22 of file ParserGrammarUnexpectedTokenException.hpp.

References errorString, and expectedToken.

### 6.140.3 Member Function Documentation

6.140.3.1 `std::string multiscale::verification::ParserGrammar-  
UnexpectedTokenException::getErrorString ( ) const  
[inline]`

Get the error string.

Definition at line 29 of file ParserGrammarUnexpectedTokenException.hpp.

References errorString.

Referenced by multiscale::verification::Parser::parse().

6.140.3.2 `std::string multiscale::verification::ParserGrammar-  
UnexpectedTokenException::getExpectedToken ( ) const  
[inline]`

Get the expected token.

Definition at line 34 of file ParserGrammarUnexpectedTokenException.hpp.

References expectedToken.

Referenced by multiscale::verification::Parser::parse().

### 6.140.4 Member Data Documentation

6.140.4.1 `std::string multiscale::verification::ParserGrammarUnexpectedToken-  
Exception::errorString [private]`

The substring from the original string starting with the index of the error token

Definition at line 17 of file ParserGrammarUnexpectedTokenException.hpp.

Referenced by `getErrorMessage()`, and `ParserGrammarUnexpectedTokenException()`.

**6.140.4.2 std::string multiscale::verification::ParserGrammarUnexpectedToken-Exception::expectedToken [private]**

The token which was expected and was not found during parsing

Definition at line 16 of file `ParserGrammarUnexpectedTokenException.hpp`.

Referenced by `getExpectedToken()`, and `ParserGrammarUnexpectedTokenException()`.

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

- `ParserGrammarUnexpectedTokenException.hpp`

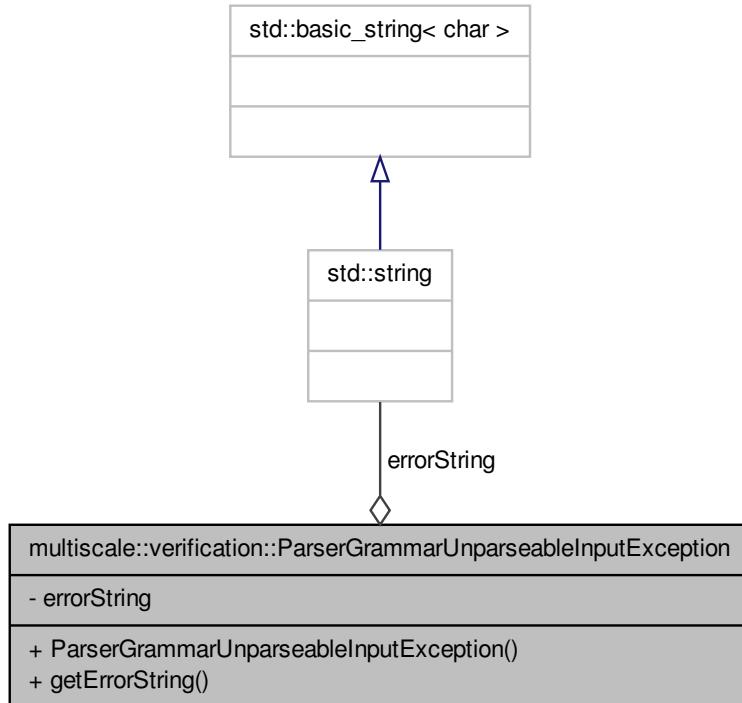
**6.141 multiscale::verification::ParserGrammarUnparseableInput-Exception Class Reference**

Class for representing "unparseable input" exceptions in the parsing process.

```
#include <ParserGrammarUnparseableInputException.hpp>
```

Collaboration diagram for `multiscale::verification::ParserGrammarUnparseableInput-`

Exception:



## Public Member Functions

- [ParserGrammarUnparseableInputException](#) (const std::string &[errorString](#))
- std::string [getErrorString](#) () const  
*Get the error string.*

## Private Attributes

- std::string [errorString](#)

### 6.141.1 Detailed Description

Class for representing "unparseable input" exceptions in the parsing process.

Definition at line 14 of file ParserGrammarUnparseableInputException.hpp.

## **6.142 multiscale::visualisation::PolarCsvToInputFilesConverter Class Reference**

---

### **6.141.2 Constructor & Destructor Documentation**

**6.141.2.1 multiscale::verification::ParserGrammarUnparseableInputException::ParserGrammarUnparseableInputException ( const std::string & *errorString* ) [inline]**

Definition at line 23 of file ParserGrammarUnparseableInputException.hpp.

### **6.141.3 Member Function Documentation**

**6.141.3.1 std::string multiscale::verification::ParserGrammarUnparseableInputException::getErrorString ( ) const [inline]**

Get the error string.

Definition at line 28 of file ParserGrammarUnparseableInputException.hpp.

Referenced by multiscale::verification::Parser::parse().

### **6.141.4 Member Data Documentation**

**6.141.4.1 std::string multiscale::verification::ParserGrammarUnparseableInputException::errorString [private]**

The substring from the original string starting with the index of the error token

Definition at line 18 of file ParserGrammarUnparseableInputException.hpp.

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

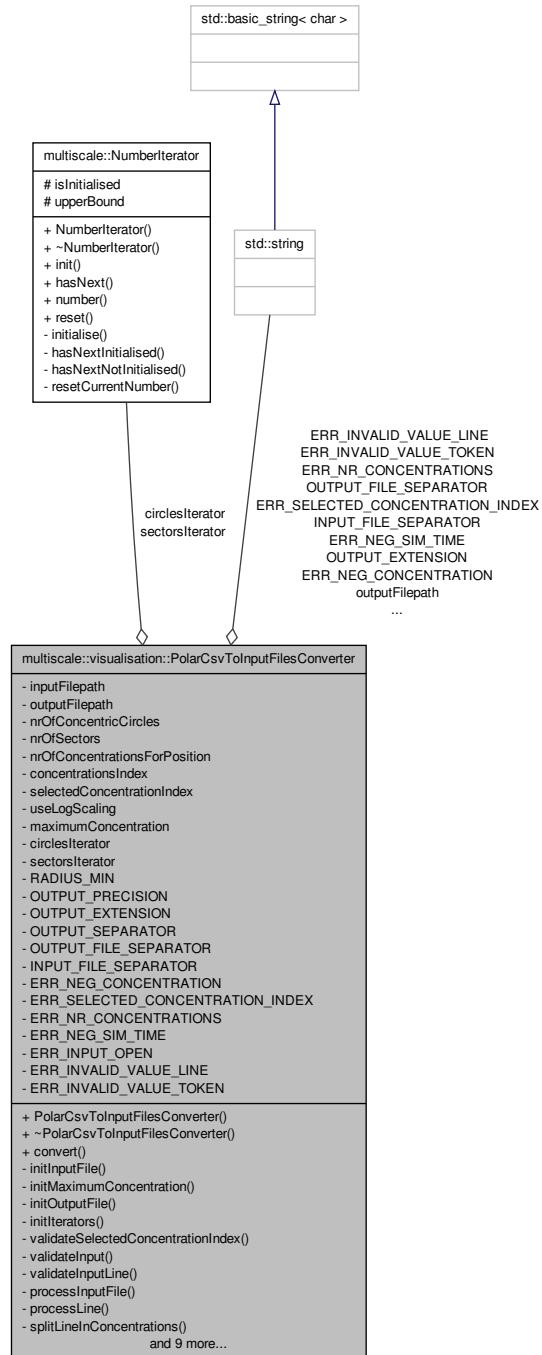
- ParserGrammarUnparseableInputException.hpp

## **6.142 multiscale::visualisation::PolarCsvToInputFilesConverter - Class Reference**

Csv file to input file converter considering polar coordinates.

```
#include <PolarCsvToInputFilesConverter.hpp>
```

Collaboration diagram for multiscale::visualisation::PolarCsvToInputFilesConverter:



## Public Member Functions

- `PolarCsvToInputFilesConverter (const std::string &inputfilepath, const std::string &outputfilepath, unsigned int nrOfConcentricCircles, unsigned int nrOfSectors, unsigned int nrOfConcentrationsForPosition, unsigned int selectedConcentrationIndex, bool useLogScaling, NumberIteratorType numberIteratorType)`
- `~PolarCsvToInputFilesConverter ()`
- `void convert ()`

*Start the conversion.*

## Private Member Functions

- `void initInputModule (std::ifstream &fin)`  
*Initialise the input file stream over the given input file.*
- `void initMaximumConcentration (std::ifstream &fin)`  
*Compute the value of member maximum concentration.*
- `void initOutputFile (std::ofstream &fout, unsigned int index, double &simulationTime)`  
*Initialise the output file with the given index and simulation time.*
- `void initIterators (const NumberIteratorType &numberIteratorType)`  
*Initialise the iterators considering the given number iterator type.*
- `void validateSelectedConcentrationIndex ()`  
*Validate the selected concentration index in case of more than one concentration for each position.*
- `void validateInput (std::ifstream &fin)`  
*Validate the input.*
- `void validateInputLine (const std::string &line, unsigned int lineNumber)`  
*Validate the provided line identified by a line number.*
- `void processInputModule (std::ifstream &fin)`  
*Process the input file.*
- `void processLine (const std::string &line, unsigned int outputIndex)`  
*Process the provided line.*
- `std::vector< double > splitLineInConcentrations (const std::string &line, double &simulationTime)`  
*Split the line in concentrations.*
- `void splitFirstPartInConcentrations (std::vector< double > &concentrations, const std::vector< std::string > &tokens, unsigned int circleIndex)`  
*Split first part of the line (i.e. part representing the origin) into concentrations.*
- `void splitOtherPartsInConcentrations (std::vector< double > &concentrations, const std::vector< std::string > &tokens, unsigned int circleIndex)`  
*Split other parts of the line (i.e. non-first part) into concentrations.*
- `double computeSimulationTime (const std::string &token)`  
*Compute the simulation time from the given token and check if it is valid.*
- `double computeNextPositionConcentration (unsigned int circleIndex, int concentrationIndex, const std::vector< std::string > &tokens)`

- `double computeConcentration (const std::string &concentration, int circleIndex)`

*Compute the concentration for the next position.*
- `double computeNonScaledConcentration (const std::string &concentration, int circleIndex)`

*Compute the concentration from the given string.*
- `double computeScaledConcentration (const std::string &concentration, int circleIndex)`

*Compute the non-scaled concentration from the given string.*
- `double computeConcentrationWrtArea (double amount, int circleIndex)`

*Compute the scaled concentration from the given string.*
- `double computeNormalisedConcentration (double concentration, int circleIndex)`

*Compute the concentration wrt. the area of the annular sector.*
- `void updateMaximumConcentration (const std::string &line, double &maximumConcentration)`

*Normalise the concentration considering the index of the current concentric circle.*
- `Update the maximum concentration if the values from the given line are greater than it.`

### Private Attributes

- `std::string inputfilepath`
- `std::string outputfilepath`
- `unsigned int nrOfConcentricCircles`
- `unsigned int nrOfSectors`
- `unsigned int nrOfConcentrationsForPosition`
- `unsigned int concentrationsIndex`
- `unsigned int selectedConcentrationIndex`
- `bool useLogScaling`
- `double maximumConcentration`
- `NumberIterator * circlesIterator`
- `NumberIterator * sectorsIterator`

### Static Private Attributes

- `static const int RADIUS_MIN = 1`
- `static const int OUTPUT_PRECISION = std::numeric_limits<double>::max_digits10`
- `static const std::string OUTPUT_EXTENSION = ".in"`
- `static const std::string OUTPUT_SEPARATOR = " "`
- `static const std::string OUTPUT_FILE_SEPARATOR = "_"`
- `static const std::string INPUT_FILE_SEPARATOR = ","`
- `static const std::string ERR_NEG_CONCENTRATION = "All concentrations must be non-negative."`

## **6.142 multiscale::visualisation::PolarCsvToInputFilesConverter Class Reference**

- static const std::string **ERR\_SELECTED\_CONCENTRATION\_INDEX** = "The selected concentration index (0-based indexing) should be smaller than the number of concentrations."
- static const std::string **ERR\_NR\_CONCENTRATIONS** = "The number of concentrations in the input file does not match the values of the input parameters height and width."
- static const std::string **ERR\_NEG\_SIM\_TIME** = "The simulation time must be non-negative."
- static const std::string **ERR\_INPUT\_OPEN** = "The input file could not be opened."
- static const std::string **ERR\_INVALID\_VALUE\_LINE** = "Invalid value on line: "
- static const std::string **ERR\_INVALID\_VALUE\_TOKEN** = ", value: "

### **6.142.1 Detailed Description**

Csv file to input file converter considering polar coordinates.

Definition at line 16 of file PolarCsvToInputFilesConverter.hpp.

### **6.142.2 Constructor & Destructor Documentation**

**6.142.2.1 PolarCsvToInputFilesConverter::PolarCsvToInputFilesConverter**  
( const std::string & *inputFilepath*, const std::string & *outputFilepath*,  
unsigned int *nrOfConcentricCircles*, unsigned int *nrOfSectors*, unsigned int  
*nrOfConcentrationsForPosition*, unsigned int *selectedConcentrationIndex*, bool  
*useLogScaling*, NumberIteratorType *numberIteratorType* )

Definition at line 22 of file PolarCsvToInputFilesConverter.cpp.

**6.142.2.2 PolarCsvToInputFilesConverter::~PolarCsvToInputFilesConverter( )**

Definition at line 46 of file PolarCsvToInputFilesConverter.cpp.

### **6.142.3 Member Function Documentation**

**6.142.3.1 double PolarCsvToInputFilesConverter::computeConcentration ( const**  
std::string & *concentration*, int *circleIndex* ) [private]

Compute the concentration from the given string.

Compute the concentration from the given string considering the index of the current concentric circle

#### **Parameters**

|                                  |                                       |
|----------------------------------|---------------------------------------|
| <i>concentra-</i><br><i>tion</i> | String representing the concentration |
| <i>circleIndex</i>               | Index of the concentric circle        |

Definition at line 314 of file PolarCsvToInputFilesConverter.cpp.

**6.142.3.2 double PolarCsvToInputFilesConverter::computeConcentrationWrtArea ( double *amount*, int *circleIndex* ) [private]**

Compute the concentration wrt. the area of the annular sector.

**Parameters**

|                    |                                                                         |
|--------------------|-------------------------------------------------------------------------|
| <i>amount</i>      | Amount in annular sector                                                |
| <i>circleIndex</i> | Index of the concentric circle which will be used to determine the area |

Definition at line 343 of file PolarCsvToInputFilesConverter.cpp.

**6.142.3.3 double PolarCsvToInputFilesConverter::computeNextPosition- Concentration ( unsigned int *circleIndex*, int *concentrationIndex*, const std::vector< std::string > & *tokens* ) [private]**

Compute the concentration for the next position.

**Parameters**

|                           |                                                              |
|---------------------------|--------------------------------------------------------------|
| <i>circleIndex</i>        | Index of the current concentric circle                       |
| <i>concentrationIndex</i> | Index of the current concentration from the vector of tokens |
| <i>tokens</i>             | Vector of tokens                                             |

Definition at line 285 of file PolarCsvToInputFilesConverter.cpp.

References multiscale::Numeric::division().

**6.142.3.4 double PolarCsvToInputFilesConverter::computeNonScaled- Concentration ( const std::string & *concentration*, int *circleIndex* ) [private]**

Compute the non-scaled concentration from the given string.

Compute the non-scaled concentration from the given string considering the index of the current concentric circle

**Parameters**

|                      |                                       |
|----------------------|---------------------------------------|
| <i>concentration</i> | String representing the concentration |
| <i>circleIndex</i>   | Index of the concentric circle        |

Definition at line 321 of file PolarCsvToInputFilesConverter.cpp.

## **6.142 multiscale::visualisation::PolarCsvToInputFilesConverter Class Reference**

**6.142.3.5 double PolarCsvToInputFilesConverter::computeNormalisedConcentration ( double *concentration*, int *circleIndex* ) [private]**

Normalise the concentration considering the index of the current concentric circle.

Normalise the concentration considering the index of the current concentric circle by dividing it to the maximum concentration

### Parameters

|                      |                                |
|----------------------|--------------------------------|
| <i>concentration</i> | The concentration              |
| <i>circleIndex</i>   | Index of the concentric circle |

Definition at line 347 of file PolarCsvToInputFilesConverter.cpp.

**6.142.3.6 double PolarCsvToInputFilesConverter::computeScaledConcentration ( const std::string & *concentration*, int *circleIndex* ) [private]**

Compute the scaled concentration from the given string.

Compute the scaled concentration from the given string considering the index of the current concentric circle by applying a logit transformation to it

### Parameters

|                      |                                       |
|----------------------|---------------------------------------|
| <i>concentration</i> | String representing the concentration |
| <i>circleIndex</i>   | Index of the concentric circle        |

Definition at line 328 of file PolarCsvToInputFilesConverter.cpp.

**6.142.3.7 double PolarCsvToInputFilesConverter::computeSimulationTime ( const std::string & *token* ) [private]**

Compute the simulation time from the given token and check if it is valid.

### Parameters

|              |                |
|--------------|----------------|
| <i>token</i> | Token (string) |
|--------------|----------------|

Definition at line 275 of file PolarCsvToInputFilesConverter.cpp.

**6.142.3.8 void PolarCsvToInputFilesConverter::convert ( )**

Start the conversion.

Definition at line 51 of file PolarCsvToInputFilesConverter.cpp.

---

**6.142.3.9 void PolarCsvToInputFilesConverter::initInputFile ( std::ifstream & *fin* ) [private]**

Initialise the input file stream over the given input file.

**Parameters**

|            |                   |
|------------|-------------------|
| <i>fin</i> | Input file stream |
|------------|-------------------|

Definition at line 64 of file PolarCsvToInputFilesConverter.cpp.

**6.142.3.10 void PolarCsvToInputFilesConverter::initIterators ( const NumberIteratorType & *numberIteratorType* ) [private]**

Initialise the iterators considering the given number iterator type.

**Parameters**

|                                 |                                 |
|---------------------------------|---------------------------------|
| <i>number-<br/>IteratorType</i> | The type of the number iterator |
|---------------------------------|---------------------------------|

Definition at line 114 of file PolarCsvToInputFilesConverter.cpp.

References multiscale::LEXICOGRAPHIC, and multiscale::STANDARD.

**6.142.3.11 void PolarCsvToInputFilesConverter::initMaximumConcentration ( std::ifstream & *fin* ) [private]**

Compute the value of member maximum concentration.

**Parameters**

|            |                   |
|------------|-------------------|
| <i>fin</i> | Input file stream |
|------------|-------------------|

Definition at line 72 of file PolarCsvToInputFilesConverter.cpp.

**6.142.3.12 void PolarCsvToInputFilesConverter::initOutputFile ( std::ofstream & *fout*, unsigned int *index*, double & *simulationTime* ) [private]**

Initialise the output file with the given index and simulation time.

**Parameters**

|                             |                          |
|-----------------------------|--------------------------|
| <i>fout</i>                 | Output file stream       |
| <i>index</i>                | Index of the output file |
| <i>simulation-<br/>Time</i> | Simulation time          |

## **6.142 multiscale::visualisation::PolarCsvToInputFilesConverter Class Reference**

Definition at line 96 of file PolarCsvToInputFilesConverter.cpp.

References multiscale::StringManipulator::toString().

**6.142.3.13 void PolarCsvToInputFilesConverter::processInputFile ( std::ifstream & *fin* ) [private]**

Process the input file.

Read the concentrations and normalise them if it is the case.

### Parameters

|            |                   |
|------------|-------------------|
| <i>fin</i> | Input file stream |
|------------|-------------------|

Definition at line 182 of file PolarCsvToInputFilesConverter.cpp.

**6.142.3.14 void PolarCsvToInputFilesConverter::processLine ( const std::string & *line*, unsigned int *outputIndex* ) [private]**

Process the provided line.

### Parameters

|                    |                                                 |
|--------------------|-------------------------------------------------|
| <i>line</i>        | Line                                            |
| <i>outputIndex</i> | Index integrated in the name of the output file |

Definition at line 197 of file PolarCsvToInputFilesConverter.cpp.

**6.142.3.15 void PolarCsvToInputFilesConverter::splitFirstPartInConcentrations ( std::vector< double > & *concentrations*, const std::vector< std::string > & *tokens*, unsigned int *circleIndex* ) [private]**

Split first part of the line (i.e. part representing the origin) into concentrations.

### Parameters

|                       |                                        |
|-----------------------|----------------------------------------|
| <i>concentrations</i> | Concentrations extracted from tokens   |
| <i>tokens</i>         | Tokens representing the line           |
| <i>circleIndex</i>    | Index of the current concentric circle |

Definition at line 245 of file PolarCsvToInputFilesConverter.cpp.

---

**6.142.3.16 std::vector< double > PolarCsvToInputFilesConverter::splitLineInConcentrations ( const std::string & *line*, double & *simulationTime* ) [private]**

Split the line in concentrations.

**Parameters**

|                        |                                          |
|------------------------|------------------------------------------|
| <i>line</i>            | Line                                     |
| <i>simulation-Time</i> | Simulation time associated with the line |

Definition at line 219 of file PolarCsvToInputFilesConverter.cpp.

References multiscale::StringManipulator::split().

**6.142.3.17 void PolarCsvToInputFilesConverter::splitOtherPartsInConcentrations ( std::vector< double > & *concentrations*, const std::vector< std::string > & *tokens*, unsigned int *circleIndex* ) [private]**

Split other parts of the line (i.e. non-first part) into concentrations.

**Parameters**

|                       |                                        |
|-----------------------|----------------------------------------|
| <i>concentrations</i> | Concentrations extracted from tokens   |
| <i>tokens</i>         | Tokens representing the line           |
| <i>circleIndex</i>    | Index of the current concentric circle |

Definition at line 259 of file PolarCsvToInputFilesConverter.cpp.

**6.142.3.18 void PolarCsvToInputFilesConverter::updateMaximumConcentration ( const std::string & *line*, double & *maximumConcentration* ) [private]**

Update the maximum concentration if the values from the given line are greater than it.

**Parameters**

|                              |                           |
|------------------------------|---------------------------|
| <i>line</i>                  | Line from input file      |
| <i>maximum-Concentration</i> | The maximum concentration |

Definition at line 351 of file PolarCsvToInputFilesConverter.cpp.

**6.142.3.19 void PolarCsvToInputFilesConverter::validateInput ( std::ifstream & *fin* ) [private]**

Validate the input.

## **6.142 multiscale::visualisation::PolarCsvToInputFilesConverter Class Reference**

### **Parameters**

|            |                   |
|------------|-------------------|
| <i>fin</i> | Input file stream |
|------------|-------------------|

Definition at line 137 of file PolarCsvToInputFilesConverter.cpp.

**6.142.3.20 void PolarCsvToInputFilesConverter::validateInputLine ( const std::string & *line*, unsigned int *lineNumber* ) [private]**

Validate the provided line identified by a line number.

### **Parameters**

|                   |                      |
|-------------------|----------------------|
| <i>line</i>       | Line from input file |
| <i>lineNumber</i> | Number of the line   |

Definition at line 161 of file PolarCsvToInputFilesConverter.cpp.

References multiscale::StringManipulator::split().

**6.142.3.21 void PolarCsvToInputFilesConverter::validateSelectedConcentration-Index( ) [private]**

Validate the selected concentration index in case of more than one concentration for each position.

Definition at line 131 of file PolarCsvToInputFilesConverter.cpp.

## **6.142.4 Member Data Documentation**

**6.142.4.1 NumberIterator\* multiscale::visualisation::PolarCsvToInputFiles-Converter::circlesIterator [private]**

Iterator over the number of concentric circles

Definition at line 41 of file PolarCsvToInputFilesConverter.hpp.

**6.142.4.2 unsigned int multiscale::visualisation::PolarCsvToInputFilesConverter-::concentrationsIndex [private]**

Index of the current concentration

Definition at line 27 of file PolarCsvToInputFilesConverter.hpp.

**6.142.4.3 const std::string PolarCsvToInputFilesConverter::ERR\_INPUT\_OPEN = "The input file could not be opened." [static, private]**

Definition at line 227 of file PolarCsvToInputFilesConverter.hpp.

```
6.142.4.4 const std::string PolarCsvToInputFilesConverter::ERR_IN-
    VALID_VALUE_LINE = "Invalid value on line: " [static,
    private]
```

Definition at line 228 of file PolarCsvToInputFilesConverter.hpp.

```
6.142.4.5 const std::string PolarCsvToInputFilesConverter::ERR-
    _INVALID_VALUE_TOKEN = ", value: " [static,
    private]
```

Definition at line 229 of file PolarCsvToInputFilesConverter.hpp.

```
6.142.4.6 const std::string PolarCsvToInputFilesConverter::ERR_NEG_CONC-
    ENTRATION = "All concentrations must be non-negative." [static,
    private]
```

Definition at line 223 of file PolarCsvToInputFilesConverter.hpp.

```
6.142.4.7 const std::string PolarCsvToInputFilesConverter::ERR_NEG_SIM_TIME =
    "The simulation time must be non-negative." [static, private]
```

Definition at line 226 of file PolarCsvToInputFilesConverter.hpp.

```
6.142.4.8 const std::string PolarCsvToInputFilesConverter::ERR_NR_CONCENTRA-
    TIONS = "The number of concentrations in the input file does not match the values of
    the input parameters height and width." [static, private]
```

Definition at line 225 of file PolarCsvToInputFilesConverter.hpp.

```
6.142.4.9 const std::string PolarCsvToInputFilesConverter::ERR_SELECTED_CON-
    CENTRATION_INDEX = "The selected concentration index (0-based indexing)
    should be smaller than the number of concentrations." [static, private]
```

Definition at line 224 of file PolarCsvToInputFilesConverter.hpp.

```
6.142.4.10 const std::string PolarCsvToInputFilesConverter-
    ::INPUT_FILE_SEPARATOR = "," [static,
    private]
```

Definition at line 221 of file PolarCsvToInputFilesConverter.hpp.

## **6.142 multiscale::visualisation::PolarCsvToInputFilesConverter Class Reference**

---

**6.142.4.11 std::string multiscale::visualisation::PolarCsvToInputFilesConverter-  
::filepath [private]**

Path to the input file

Definition at line 20 of file PolarCsvToInputFilesConverter.hpp.

**6.142.4.12 double multiscale::visualisation::PolarCsvToInputFilesConverter-  
::maximumConcentration [private]**

The maximum concentration in the input file

Definition at line 39 of file PolarCsvToInputFilesConverter.hpp.

**6.142.4.13 unsigned int multiscale::visualisation::PolarCsvTo-  
InputFilesConverter::nrOfConcentrationsForPosition  
[private]**

Number of concentrations for each position

Definition at line 25 of file PolarCsvToInputFilesConverter.hpp.

**6.142.4.14 unsigned int multiscale::visualisation::PolarCsv-  
ToInputFilesConverter::nrOfConcentricCircles  
[private]**

Number of concentric circles

Definition at line 23 of file PolarCsvToInputFilesConverter.hpp.

**6.142.4.15 unsigned int multiscale::visualisation::PolarCsvToInputFilesConverter-  
::nrOfSectors [private]**

Number of sectors

Definition at line 24 of file PolarCsvToInputFilesConverter.hpp.

**6.142.4.16 const std::string PolarCsvToInputFilesConverter::OUTPUT\_EXTENSION =  
".in" [static, private]**

Definition at line 218 of file PolarCsvToInputFilesConverter.hpp.

**6.142.4.17 const std::string PolarCsvToInputFilesConverter::-  
OUTPUT\_FILE\_SEPARATOR = "\_" [static,  
private]**

Definition at line 220 of file PolarCsvToInputFilesConverter.hpp.

```
6.142.4.18 const int PolarCsvToInputFilesConverter::OUTPUT_PRECISION =
    std::numeric_limits<double>::max_digits10 [static, private]
```

Definition at line 216 of file PolarCsvToInputFilesConverter.hpp.

```
6.142.4.19 const std::string PolarCsvToInputFilesConverter::OUTPUT_SEPARATOR
    = "" [static, private]
```

Definition at line 219 of file PolarCsvToInputFilesConverter.hpp.

```
6.142.4.20 std::string multiscale::visualisation::PolarCsvToInputFilesConverter-
    ::outputFilepath [private]
```

Path to the output file

Definition at line 21 of file PolarCsvToInputFilesConverter.hpp.

```
6.142.4.21 const int PolarCsvToInputFilesConverter::RADIUS_MIN = 1 [static,
    private]
```

Definition at line 214 of file PolarCsvToInputFilesConverter.hpp.

```
6.142.4.22 NumberIterator* multiscale::visualisation::PolarCsvToInputFiles-
    Converter::sectorsIterator [private]
```

Iterator over the number of sectors

Definition at line 42 of file PolarCsvToInputFilesConverter.hpp.

```
6.142.4.23 unsigned int multiscale::visualisation::PolarCsvTo-
    InputFilesConverter::selectedConcentrationIndex
    [private]
```

Index of the concentration A in case the number of concentrations for each position is greater than 1

finalConcentration = A / (A1 + A2 + ... + AN), where N is the number of concentrations for each position

Definition at line 29 of file PolarCsvToInputFilesConverter.hpp.

```
6.142.4.24 bool multiscale::visualisation::PolarCsvToInputFilesConverter::use-
    LogScaling [private]
```

Flag for using logarithmic scaling for concentrations

Definition at line 36 of file PolarCsvToInputFilesConverter.hpp.

## **6.143 multiscale::visualisation::PolarGnuplotScriptGenerator Class Reference**

---

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

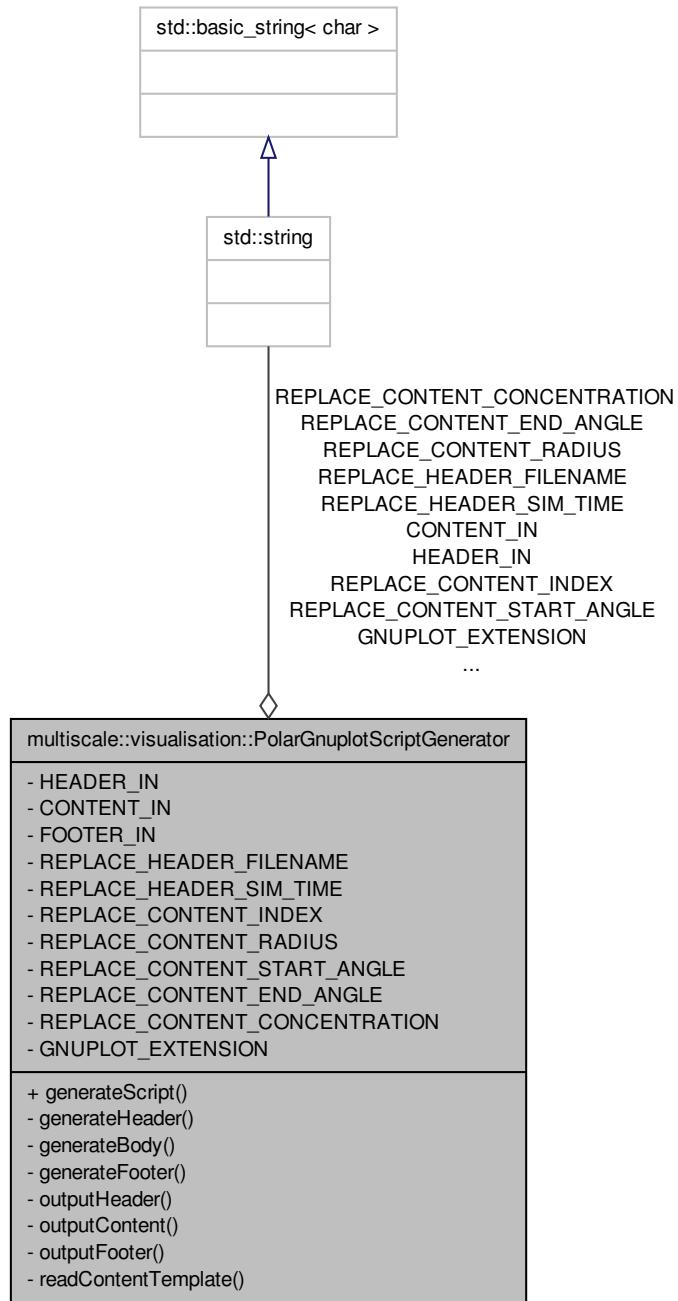
- `PolarCsvToInputFilesConverter.hpp`
  
- `PolarCsvToInputFilesConverter.cpp`

## **6.143 multiscale::visualisation::PolarGnuplotScriptGenerator - Class Reference**

Gnuplot script generator from the provided annular sectors.

```
#include <PolarGnuplotScriptGenerator.hpp>
```

Collaboration diagram for multiscale::visualisation::PolarGnuplotScriptGenerator:



### Static Public Member Functions

- static void `generateScript` (const std::vector< `AnnularSector` > &annularSectors, double simulationTime, const std::string &outputFilepath)  
*Generate the script.*

### Static Private Member Functions

- static void `generateHeader` (std::ofstream &fout, const std::string &outputFilepath, double simulationTime)  
*Generate the header of the script.*
- static void `generateBody` (const std::vector< `AnnularSector` > &annularSectors, std::ofstream &fout)  
*Generate the body/content of the script.*
- static void `generateFooter` (std::ofstream &fout)  
*Generate the footer of the script.*
- static void `outputHeader` (std::ifstream &fin, const std::string &outputFilename, double simulationTime, std::ofstream &fout)  
*Output the header of the script.*
- static void `outputContent` (const std::vector< `AnnularSector` > &annularSectors, const std::string &contentTemplate, std::ofstream &fout)  
*Output the content of the script.*
- static void `outputFooter` (std::ifstream &fin, std::ofstream &fout)  
*Output the footer of the script.*
- static std::string `readContentTemplate` (std::ifstream &fin)  
*Read content template.*

### Static Private Attributes

- static const std::string `HEADER_IN` = "/usr/local/share/mule/config/visualisation/circular/header.-in"
- static const std::string `CONTENT_IN` = "/usr/local/share/mule/config/visualisation/circular/content.-in"
- static const std::string `FOOTER_IN` = "/usr/local/share/mule/config/visualisation/circular/footer.-in"
- static const std::string `REPLACE_HEADER_FILENAME` = "OUTPUT\_FILENAME-E"
- static const std::string `REPLACE_HEADER_SIM_TIME` = "OUTPUT\_SIM\_TIME"
- static const std::string `REPLACE_CONTENT_INDEX` = "OBJ\_INDEX"
- static const std::string `REPLACE_CONTENT_RADIUS` = "OBJ\_END\_RADIUS"
- static const std::string `REPLACE_CONTENT_START_ANGLE` = "OBJ\_START\_ANGLE"
- static const std::string `REPLACE_CONTENT_END_ANGLE` = "OBJ\_END\_ANGLE"
- static const std::string `REPLACE_CONTENT_CONCENTRATION` = "OBJ\_CO-NCENTRATION"
- static const std::string `GNUPLOT_EXTENSION` = ".plt"

### 6.143.1 Detailed Description

Gnuplot script generator from the provided annular sectors.

Definition at line 14 of file PolarGnuplotScriptGenerator.hpp.

### 6.143.2 Member Function Documentation

**6.143.2.1 void PolarGnuplotScriptGenerator::generateBody ( const std::vector<AnnularSector > & *annularSectors*, std::ofstream & *fout* ) [static, private]**

Generate the body/content of the script.

#### Parameters

|                        |                    |
|------------------------|--------------------|
| <i>annular-Sectors</i> | Annular sectors    |
| <i>fout</i>            | Output file stream |

Definition at line 40 of file PolarGnuplotScriptGenerator.cpp.

References CONTENT\_IN, outputContent(), and readContentTemplate().

Referenced by generateScript().

**6.143.2.2 void PolarGnuplotScriptGenerator::generateFooter ( std::ofstream & *fout* ) [static, private]**

Generate the footer of the script.

#### Parameters

|             |                    |
|-------------|--------------------|
| <i>fout</i> | Output file stream |
|-------------|--------------------|

Definition at line 53 of file PolarGnuplotScriptGenerator.cpp.

References FOOTER\_IN, and outputFooter().

Referenced by generateScript().

**6.143.2.3 void PolarGnuplotScriptGenerator::generateHeader ( std::ofstream & *fout*, const std::string & *outputFilepath*, double *simulationTime* ) [static, private]**

Generate the header of the script.

#### Parameters

|             |                    |
|-------------|--------------------|
| <i>fout</i> | Output file stream |
|-------------|--------------------|

|                        |                         |
|------------------------|-------------------------|
| <i>output-Filepath</i> | Path to the output file |
| <i>simulation-Time</i> | Simulation time         |

Definition at line 27 of file PolarGnuplotScriptGenerator.cpp.

References multiscale::StringManipulator::filenameFromPath(), HEADER\_IN, and outputHeader().

Referenced by generateScript().

**6.143.2.4 void PolarGnuplotScriptGenerator::generateScript ( const std::vector<AnnularSector > & annularSectors, double simulationTime, const std::string & outputfilepath ) [static]**

Generate the script.

**Parameters**

|                        |                         |
|------------------------|-------------------------|
| <i>annular-Sectors</i> | Annular sectors         |
| <i>simulation-Time</i> | Simulation time         |
| <i>output-Filepath</i> | Path of the output file |

Definition at line 13 of file PolarGnuplotScriptGenerator.cpp.

References generateBody(), generateFooter(), generateHeader(), and GNUPLOT\_EXTENSION.

Referenced by multiscale::visualisation::CartesianToPolarConverter::outputResultsAsScript().

**6.143.2.5 void PolarGnuplotScriptGenerator::outputContent ( const std::vector<AnnularSector > & annularSectors, const std::string & contentTemplate, std::ofstream & fout ) [static, private]**

Output the content of the script.

**Parameters**

|                         |                                                             |
|-------------------------|-------------------------------------------------------------|
| <i>annular-Sectors</i>  | Annular sectors                                             |
| <i>content-Template</i> | Template used for generating output for each annular sector |
| <ifout< i=""></ifout<>  | Output file stream                                          |

Definition at line 85 of file PolarGnuplotScriptGenerator.cpp.

References multiscale::StringManipulator::replace(), REPLACE\_CONTENT\_CONCENTRATION, REPLACE\_CONTENT\_END\_ANGLE, REPLACE\_CONTENT\_INDEX, REPLACE\_CONTENT\_RADIUS, and REPLACE\_CONTENT\_START\_ANGLE.

Referenced by generateBody().

**6.143.2.6 void PolarGnuplotScriptGenerator::outputFooter ( std::ifstream & *fin*, std::ofstream & *fout* ) [static, private]**

Output the footer of the script.

**Parameters**

|             |                    |
|-------------|--------------------|
| <i>fin</i>  | Input file stream  |
| <i>fout</i> | Output file stream |

Definition at line 124 of file PolarGnuplotScriptGenerator.cpp.

Referenced by generateFooter().

**6.143.2.7 void PolarGnuplotScriptGenerator::outputHeader ( std::ifstream & *fin*, const std::string & *outputFilename*, double *simulationTime*, std::ofstream & *fout* ) [static, private]**

Output the header of the script.

**Parameters**

|                       |                         |
|-----------------------|-------------------------|
| <i>fin</i>            | Input file stream       |
| <i>outputFilename</i> | Name of the output file |
| <i>simulationTime</i> | Simulation time         |
| <i>fout</i>           | Output file stream      |

Definition at line 63 of file PolarGnuplotScriptGenerator.cpp.

References multiscale::StringManipulator::replace(), REPLACE\_HEADER\_FILENAME, and REPLACE\_HEADER\_SIM\_TIME.

Referenced by generateHeader().

**6.143.2.8 std::string PolarGnuplotScriptGenerator::readContentTemplate ( std::ifstream & *fin* ) [static, private]**

Read content template.

**Parameters**

|            |                   |
|------------|-------------------|
| <i>fin</i> | Input file stream |
|------------|-------------------|

Definition at line 134 of file PolarGnuplotScriptGenerator.cpp.

Referenced by generateBody().

### 6.143.3 Member Data Documentation

**6.143.3.1 const std::string PolarGnuplotScriptGenerator::CONTENT\_IN =  
"/usr/local/share/mule/config/visualisation/circular/content.in" [static,  
private]**

Definition at line 90 of file PolarGnuplotScriptGenerator.hpp.

Referenced by generateBody().

**6.143.3.2 const std::string PolarGnuplotScriptGenerator::FOOTER\_IN =  
"/usr/local/share/mule/config/visualisation/circular/footer.in" [static,  
private]**

Definition at line 91 of file PolarGnuplotScriptGenerator.hpp.

Referenced by generateFooter().

**6.143.3.3 const std::string PolarGnuplotScriptGenerator::GNUPLOT\_EXTENSION =  
.plt" [static, private]**

Definition at line 102 of file PolarGnuplotScriptGenerator.hpp.

Referenced by generateScript().

**6.143.3.4 const std::string PolarGnuplotScriptGenerator::HEADER\_IN =  
"/usr/local/share/mule/config/visualisation/circular/header.in" [static,  
private]**

Definition at line 89 of file PolarGnuplotScriptGenerator.hpp.

Referenced by generateHeader().

**6.143.3.5 const std::string PolarGnuplotScriptGenerator::REPLACE\_CON-  
CENT\_CONCENTRATION = "OBJ\_CONCENTRATION" [static,  
private]**

Definition at line 100 of file PolarGnuplotScriptGenerator.hpp.

Referenced by outputContent().

```
6.143.3.6 const std::string PolarGnuplotScriptGenerator::REPLACE_-  
CONTENT_END_ANGLE = "OBJ_END_ANGLE" [static,  
private]
```

Definition at line 99 of file PolarGnuplotScriptGenerator.hpp.

Referenced by outputContent().

```
6.143.3.7 const std::string PolarGnuplotScriptGenerator::REPL-  
ACE_CONTENT_INDEX = "OBJ_INDEX" [static,  
private]
```

Definition at line 96 of file PolarGnuplotScriptGenerator.hpp.

Referenced by outputContent().

```
6.143.3.8 const std::string PolarGnuplotScriptGenerator::REPLAC-  
E_CONTENT_RADIUS = "OBJ_END_RADIUS" [static,  
private]
```

Definition at line 97 of file PolarGnuplotScriptGenerator.hpp.

Referenced by outputContent().

```
6.143.3.9 const std::string PolarGnuplotScriptGenerator::REPLACE_C-  
ONTENT_START_ANGLE = "OBJ_START_ANGLE" [static,  
private]
```

Definition at line 98 of file PolarGnuplotScriptGenerator.hpp.

Referenced by outputContent().

```
6.143.3.10 const std::string PolarGnuplotScriptGenerator::REPLACE-  
_HEADER_FILENAME = "OUTPUT_FILENAME" [static,  
private]
```

Definition at line 93 of file PolarGnuplotScriptGenerator.hpp.

Referenced by outputHeader().

```
6.143.3.11 const std::string PolarGnuplotScriptGenerator::REPLAC-  
E_HEADER_SIM_TIME = "OUTPUT_SIM_TIME" [static,  
private]
```

Definition at line 94 of file PolarGnuplotScriptGenerator.hpp.

Referenced by outputHeader().

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

- [PolarGnuplotScriptGenerator.hpp](#)
- [PolarGnuplotScriptGenerator.cpp](#)

## **6.144 multiscale::verification::PrimaryConstraintAttribute Class Reference**

Class for representing a primary constraint attribute.

```
#include <PrimaryConstraintAttribute.hpp>
```

### **Public Attributes**

- [PrimaryConstraintAttributeType primaryConstraint](#)

#### **6.144.1 Detailed Description**

Class for representing a primary constraint attribute.

Definition at line 32 of file PrimaryConstraintAttribute.hpp.

#### **6.144.2 Member Data Documentation**

##### **6.144.2.1 PrimaryConstraintAttributeType multiscale::verification::PrimaryConstraintAttribute::primaryConstraint**

The primary constraint

Definition at line 36 of file PrimaryConstraintAttribute.hpp.

Referenced by multiscale::verification::ConstraintVisitor::operator()().

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

- [PrimaryConstraintAttribute.hpp](#)

## **6.145 multiscale::verification::PrimaryLogicPropertyAttribute Class Reference** -

Class for representing a primary logic property attribute.

```
#include <PrimaryLogicPropertyAttribute.hpp>
```

### **Public Attributes**

- [PrimaryLogicPropertyAttributeType primaryLogicProperty](#)

### 6.145.1 Detailed Description

Class for representing a primary logic property attribute.

Definition at line 40 of file PrimaryLogicPropertyAttribute.hpp.

### 6.145.2 Member Data Documentation

#### 6.145.2.1 PrimaryLogicPropertyAttributeType multiscale::verification::PrimaryLogicPropertyAttribute::primaryLogicProperty

The primary logic property

Definition at line 44 of file PrimaryLogicPropertyAttribute.hpp.

Referenced by multiscale::verification::LogicPropertyVisitor::operator()().

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

- PrimaryLogicPropertyAttribute.hpp

## 6.146 multiscale::verification::PrimaryNumericMeasureAttribute Class Reference

Class for representing a primary numeric measure attribute.

```
#include <PrimaryNumericMeasureAttribute.hpp>
```

### Public Attributes

- PrimaryNumericMeasureAttributeType primaryNumericMeasure

### 6.146.1 Detailed Description

Class for representing a primary numeric measure attribute.

Definition at line 28 of file PrimaryNumericMeasureAttribute.hpp.

### 6.146.2 Member Data Documentation

#### 6.146.2.1 PrimaryNumericMeasureAttributeType multiscale::verification::PrimaryNumericMeasureAttribute::primaryNumericMeasure

The primary numeric measure

Definition at line 32 of file PrimaryNumericMeasureAttribute.hpp.

Referenced by multiscale::verification::FilterNumericVisitor::operator()(), and multiscale-::verification::NumericVisitor::operator()().

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

- PrimaryNumericMeasureAttribute.hpp

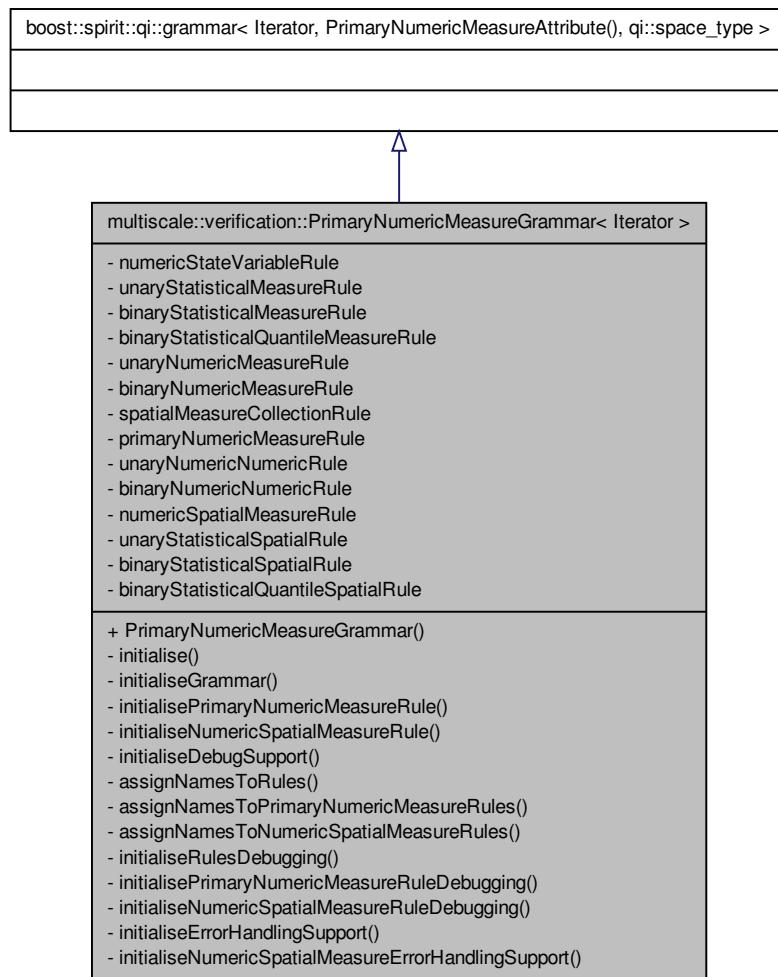
## **6.147 multiscale::verification::PrimaryNumericMeasureGrammar< Iterator > Class Template Reference**

The grammar for parsing primary numeric measure statements.

```
#include <PrimaryNumericMeasureGrammar.hpp>
```

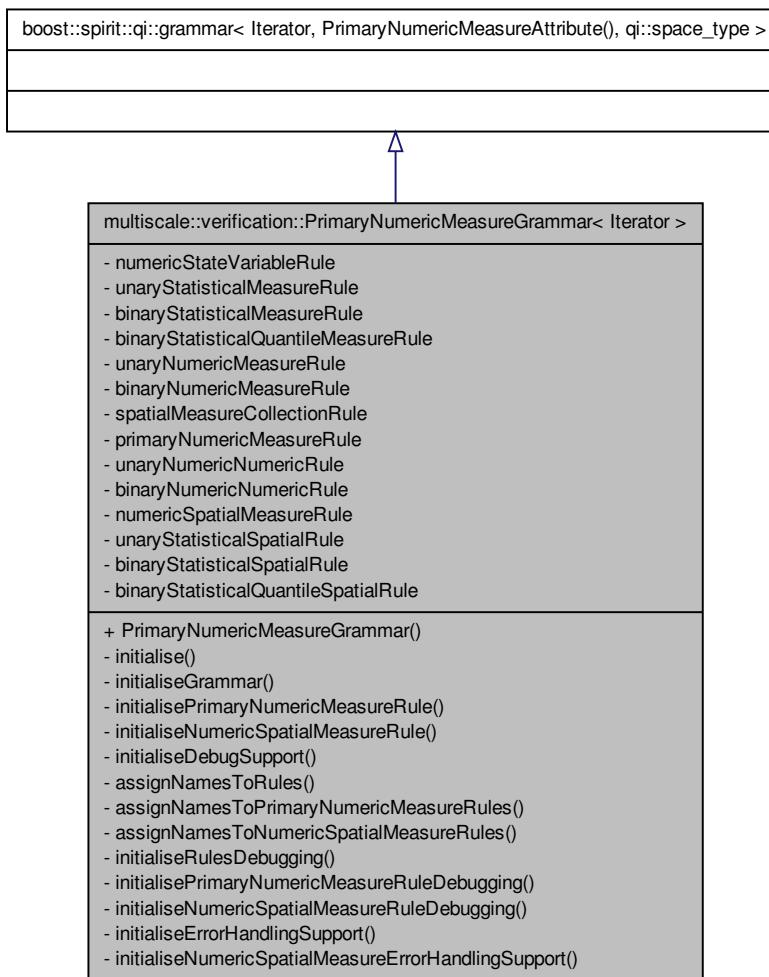
Inheritance diagram for multiscale::verification::PrimaryNumericMeasureGrammar< -

Iterator >:



Collaboration diagram for multiscale::verification::PrimaryNumericMeasureGrammar<

Iterator >:



## Public Member Functions

- [PrimaryNumericMeasureGrammar \(SpatialMeasureCollectionGrammar< Iterator > \\*spatialMeasureCollectionGrammar\)](#)

## Private Member Functions

- void [initialise \(\)](#)  
*Initialisation function.*

- void `initialiseGrammar ()`  
*Initialise the grammar.*
- void `initialisePrimaryNumericMeasureRule ()`  
*Initialise the primary numeric measure rule.*
- void `initialiseNumericSpatialMeasureRule ()`  
*Initialise the numeric spatial measure rule.*
- void `initialiseDebugSupport ()`  
*Initialise debug support.*
- void `assignNamesToRules ()`  
*Assign names to the rules.*
- void `assignNamesToPrimaryNumericMeasureRules ()`  
*Assign names to the primary numeric measure rules.*
- void `assignNamesToNumericSpatialMeasureRules ()`  
*Assign names to the numeric spatial measure rules.*
- void `initialiseRulesDebugging ()`  
*Initialise the debugging of rules.*
- void `initialisePrimaryNumericMeasureRuleDebugging ()`  
*Initialise debugging for the primary numeric measure rule.*
- void `initialiseNumericSpatialMeasureRuleDebugging ()`  
*Initialise debugging for the numeric spatial measure rule.*
- void `initialiseErrorHandlingSupport ()`  
*Initialise the error handling routines.*
- void `initialiseNumericSpatialMeasureErrorHandlingSupport ()`  
*Initialise the numeric spatial measure error handling support.*

### Private Attributes

- `NumericStateVariableGrammar < Iterator > numericStateVariableRule`
- `UnaryStatisticalMeasureGrammar < Iterator > unaryStatisticalMeasureRule`
- `BinaryStatisticalMeasureGrammar < Iterator > binaryStatisticalMeasureRule`
- `BinaryStatisticalQuantileMeasureGrammar < Iterator > binaryStatisticalQuantileMeasureRule`
- `UnaryNumericMeasureGrammar < Iterator > unaryNumericMeasureRule`
- `BinaryNumericMeasureGrammar < Iterator > binaryNumericMeasureRule`
- `SpatialMeasureCollectionGrammar < Iterator > * spatialMeasureCollectionRule`
- `qi::rule< Iterator, PrimaryNumericMeasureAttribute(), qi::space_type > primaryNumericMeasureRule`
- `qi::rule< Iterator, UnaryNumericNumericAttribute(), qi::space_type > unaryNumericNumericRule`
- `qi::rule< Iterator, BinaryNumericNumericAttribute(), qi::space_type > binaryNumericNumericRule`
- `qi::rule< Iterator, NumericSpatialMeasureAttribute(), qi::space_type > numericSpatialMeasureRule`
- `qi::rule< Iterator, UnaryStatisticalSpatialAttribute(), qi::space_type > unaryStatisticalSpatialRule`

- qi::rule< Iterator, BinaryStatisticalSpatialAttribute(), qi::space\_type > **binary-StatisticalSpatialRule**
- qi::rule< Iterator, BinaryStatisticalQuantileSpatialAttribute(), qi::space\_type > **binaryStatisticalQuantileSpatialRule**

### 6.147.1 Detailed Description

```
template<typename Iterator>class multiscale::verification::PrimaryNumericMeasureGrammar<Iterator>
```

The grammar for parsing primary numeric measure statements.

Definition at line 37 of file PrimaryNumericMeasureGrammar.hpp.

### 6.147.2 Constructor & Destructor Documentation

```
6.147.2.1 template<typename Iterator> multiscale::verification::PrimaryNumericMeasureGrammar<Iterator>::PrimaryNumericMeasureGrammar  
( SpatialMeasureCollectionGrammar< Iterator > *  
  spatialMeasureCollectionGrammar )
```

Definition at line 23 of file PrimaryNumericMeasureGrammarDefinition.hpp.

References multiscale::verification::PrimaryNumericMeasureGrammar< Iterator >::initialise().

### 6.147.3 Member Function Documentation

```
6.147.3.1 template<typename Iterator> void multiscale::verification-  
::PrimaryNumericMeasureGrammar< Iterator  
>::assignNamesToNumericSpatialMeasureRules( ) [private]
```

Assign names to the numeric spatial measure rules.

Definition at line 115 of file PrimaryNumericMeasureGrammarDefinition.hpp.

```
6.147.3.2 template<typename Iterator> void multiscale::verification-  
::PrimaryNumericMeasureGrammar< Iterator  
>::assignNamesToPrimaryNumericMeasureRules( ) [private]
```

Assign names to the primary numeric measure rules.

Definition at line 109 of file PrimaryNumericMeasureGrammarDefinition.hpp.

```
6.147.3.3 template<typename Iterator> void multiscale::verification::Primary-
    NumericMeasureGrammar< Iterator >::assignNamesToRules( )
    [private]
```

Assign names to the rules.

Definition at line 102 of file PrimaryNumericMeasureGrammarDefinition.hpp.

```
6.147.3.4 template<typename Iterator> void multiscale::verification::-
    PrimaryNumericMeasureGrammar< Iterator >::initialise( )
    [private]
```

Initialisation function.

Definition at line 32 of file PrimaryNumericMeasureGrammarDefinition.hpp.

Referenced by multiscale::verification::PrimaryNumericMeasureGrammar< Iterator >-
 ::PrimaryNumericMeasureGrammar().

```
6.147.3.5 template<typename Iterator> void multiscale::verification::Primary-
    NumericMeasureGrammar< Iterator >::initialiseDebugSupport( )
    [private]
```

Initialise debug support.

Definition at line 93 of file PrimaryNumericMeasureGrammarDefinition.hpp.

```
6.147.3.6 template<typename Iterator> void multiscale::verification::PrimaryNumeric-
    MeasureGrammar< Iterator >::initialiseErrorHandlingSupport( )
    [private]
```

Initialise the error handling routines.

Definition at line 146 of file PrimaryNumericMeasureGrammarDefinition.hpp.

```
6.147.3.7 template<typename Iterator> void multiscale::verification::Primary-
    NumericMeasureGrammar< Iterator >::initialiseGrammar( )
    [private]
```

Initialise the grammar.

Definition at line 40 of file PrimaryNumericMeasureGrammarDefinition.hpp.

```
6.147.3.8 template<typename Iterator> void multiscale::verification-
    ::PrimaryNumericMeasureGrammar< Iterator
    >::initialiseNumericSpatialMeasureErrorHandlingSupport( )
    [private]
```

Initialise the numeric spatial measure error handling support.

Definition at line 152 of file PrimaryNumericMeasureGrammarDefinition.hpp.

References multiscale::verification::handleUnexpectedTokenError.

**6.147.3.9 template<typename Iterator > void multiscale::verification::PrimaryNumericMeasureGrammar< Iterator >::initialiseNumericSpatialMeasureRule( ) [private]**

Initialise the numeric spatial measure rule.

Definition at line 56 of file PrimaryNumericMeasureGrammarDefinition.hpp.

**6.147.3.10 template<typename Iterator > void multiscale::verification::PrimaryNumericMeasureGrammar< Iterator >::initialiseNumericSpatialMeasureRuleDebugging( ) [private]**

Initialise debugging for the numeric spatial measure rule.

Definition at line 137 of file PrimaryNumericMeasureGrammarDefinition.hpp.

**6.147.3.11 template<typename Iterator > void multiscale::verification::PrimaryNumericMeasureGrammar< Iterator >::initialisePrimaryNumericMeasureRule( ) [private]**

Initialise the primary numeric measure rule.

Definition at line 47 of file PrimaryNumericMeasureGrammarDefinition.hpp.

**6.147.3.12 template<typename Iterator > void multiscale::verification::PrimaryNumericMeasureGrammar< Iterator >::initialisePrimaryNumericMeasureRuleDebugging( ) [private]**

Initialise debugging for the primary numeric measure rule.

Definition at line 131 of file PrimaryNumericMeasureGrammarDefinition.hpp.

**6.147.3.13 template<typename Iterator > void multiscale::verification::PrimaryNumericMeasureGrammar< Iterator >::initialiseRulesDebugging( ) [private]**

Initialise the debugging of rules.

Definition at line 124 of file PrimaryNumericMeasureGrammarDefinition.hpp.

#### 6.147.4 Member Data Documentation

---

```
6.147.4.1 template<typename Iterator > BinaryNumericMeasureGrammar<Iterator>
multiscale::verification::PrimaryNumericMeasureGrammar< Iterator
>::binaryNumericMeasureRule [private]
```

The grammar for parsing binary numeric measures

Definition at line 62 of file PrimaryNumericMeasureGrammar.hpp.

```
6.147.4.2 template<typename Iterator > qi::rule<Iterator,
BinaryNumericNumericAttribute(), qi::space_type>
multiscale::verification::PrimaryNumericMeasureGrammar< Iterator
>::binaryNumericNumericRule [private]
```

The rule for parsing a binary numeric numeric attribute

Definition at line 78 of file PrimaryNumericMeasureGrammar.hpp.

```
6.147.4.3 template<typename Iterator > BinaryStatisticalMeasureGrammar<Iterator>
multiscale::verification::PrimaryNumericMeasureGrammar< Iterator
>::binaryStatisticalMeasureRule [private]
```

The grammar for parsing binary statistical measures

Definition at line 52 of file PrimaryNumericMeasureGrammar.hpp.

```
6.147.4.4 template<typename Iterator > BinaryStatisticalQuantileMeasureGrammar<-
Iterator> multiscale::verification::PrimaryNumericMeasureGrammar<-
Iterator >::binaryStatisticalQuantileMeasureRule [private]
```

The grammar for parsing binary statistical quantile measures

Definition at line 55 of file PrimaryNumericMeasureGrammar.hpp.

```
6.147.4.5 template<typename Iterator > qi::rule<Iterator, Binary-
StatisticalQuantileSpatialAttribute(), qi::space_type>
multiscale::verification::PrimaryNumericMeasureGrammar< Iterator
>::binaryStatisticalQuantileSpatialRule [private]
```

The rule for parsing a binary statistical quantile spatial attribute

Definition at line 90 of file PrimaryNumericMeasureGrammar.hpp.

```
6.147.4.6 template<typename Iterator > qi::rule<Iterator,
BinaryStatisticalSpatialAttribute(), qi::space_type>
multiscale::verification::PrimaryNumericMeasureGrammar< Iterator
>::binaryStatisticalSpatialRule [private]
```

The rule for parsing a binary statistical spatial attribute

Definition at line 87 of file PrimaryNumericMeasureGrammar.hpp.

```
6.147.4.7 template<typename Iterator> qi::rule<Iterator,
    NumericSpatialMeasureAttribute(), qi::space_type>
multiscale::verification::PrimaryNumericMeasureGrammar< Iterator
>::numericSpatialMeasureRule [private]
```

The rule for parsing a numeric spatial measure

Definition at line 82 of file PrimaryNumericMeasureGrammar.hpp.

```
6.147.4.8 template<typename Iterator> NumericStateVariableGrammar<Iterator>
multiscale::verification::PrimaryNumericMeasureGrammar< Iterator
>::numericStateVariableRule [private]
```

The grammar for parsing numeric state variables

Definition at line 45 of file PrimaryNumericMeasureGrammar.hpp.

```
6.147.4.9 template<typename Iterator> qi::rule<Iterator,
    PrimaryNumericMeasureAttribute(), qi::space_type>
multiscale::verification::PrimaryNumericMeasureGrammar< Iterator
>::primaryNumericMeasureRule [private]
```

The rule for parsing a primary numeric numeric attribute

Definition at line 72 of file PrimaryNumericMeasureGrammar.hpp.

```
6.147.4.10 template<typename Iterator> SpatialMeasureCollectionGrammar<-
    Iterator>* multiscale::verification::PrimaryNumericMeasureGrammar<
    Iterator>::spatialMeasureCollectionRule [private]
```

The grammar for parsing spatial measure collections

Definition at line 66 of file PrimaryNumericMeasureGrammar.hpp.

```
6.147.4.11 template<typename Iterator> UnaryNumericMeasureGrammar<Iterator>
multiscale::verification::PrimaryNumericMeasureGrammar< Iterator
>::unaryNumericMeasureRule [private]
```

The grammar for parsing unary numeric measures

Definition at line 59 of file PrimaryNumericMeasureGrammar.hpp.

---

```
6.147.4.12 template<typename Iterator > qi::rule<Iterator,
    UnaryNumericNumericAttribute(), qi::space_type>
    multiscale::verification::PrimaryNumericMeasureGrammar< Iterator
    >::unaryNumericNumericRule [private]
```

The rule for parsing a unary numeric numeric attribute

Definition at line 75 of file PrimaryNumericMeasureGrammar.hpp.

---

```
6.147.4.13 template<typename Iterator > UnaryStatisticalMeasureGrammar<Iterator>
    multiscale::verification::PrimaryNumericMeasureGrammar< Iterator
    >::unaryStatisticalMeasureRule [private]
```

The grammar for parsing unary statistical measures

Definition at line 49 of file PrimaryNumericMeasureGrammar.hpp.

---

```
6.147.4.14 template<typename Iterator > qi::rule<Iterator,
    UnaryStatisticalSpatialAttribute(), qi::space_type>
    multiscale::verification::PrimaryNumericMeasureGrammar< Iterator
    >::unaryStatisticalSpatialRule [private]
```

The rule for parsing a unary statistical spatial attribute

Definition at line 84 of file PrimaryNumericMeasureGrammar.hpp.

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

- PrimaryNumericMeasureGrammar.hpp
- PrimaryNumericMeasureGrammarDefinition.hpp

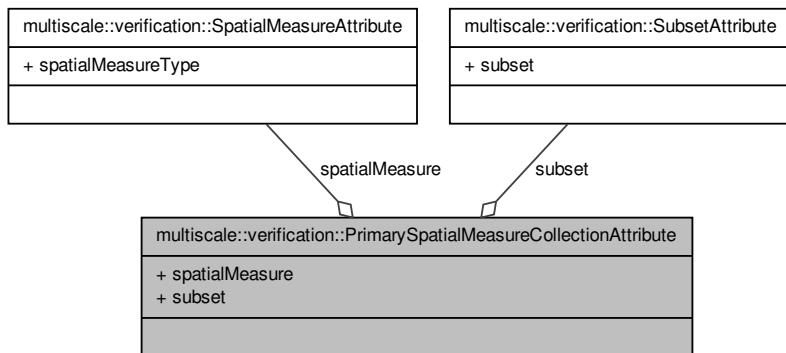
## 6.148 multiscale::verification::PrimarySpatialMeasureCollection-Attribute Class Reference

Class used to represent a primary spatial measure collection attribute.

```
#include <PrimarySpatialMeasureCollectionAttribute.hpp>
```

Collaboration diagram for multiscale::verification::PrimarySpatialMeasureCollection-

Attribute:



## Public Attributes

- `SpatialMeasureAttribute spatialMeasure`
- `SubsetAttribute subset`

### 6.148.1 Detailed Description

Class used to represent a primary spatial measure collection attribute.

Definition at line 15 of file `PrimarySpatialMeasureCollectionAttribute.hpp`.

### 6.148.2 Member Data Documentation

#### 6.148.2.1 SpatialMeasureAttribute multiscale::verification::PrimarySpatialMeasureCollectionAttribute::spatialMeasure

The spatial measure

Definition at line 19 of file `PrimarySpatialMeasureCollectionAttribute.hpp`.

Referenced by `multiscale::verification::SpatialMeasureCollectionVisitor::operator()()`.

#### 6.148.2.2 SubsetAttribute multiscale::verification::PrimarySpatialMeasureCollectionAttribute::subset

The considered subset

Definition at line 20 of file `PrimarySpatialMeasureCollectionAttribute.hpp`.

Referenced by multiscale::verification::SpatialMeasureCollectionVisitor::operator()().

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

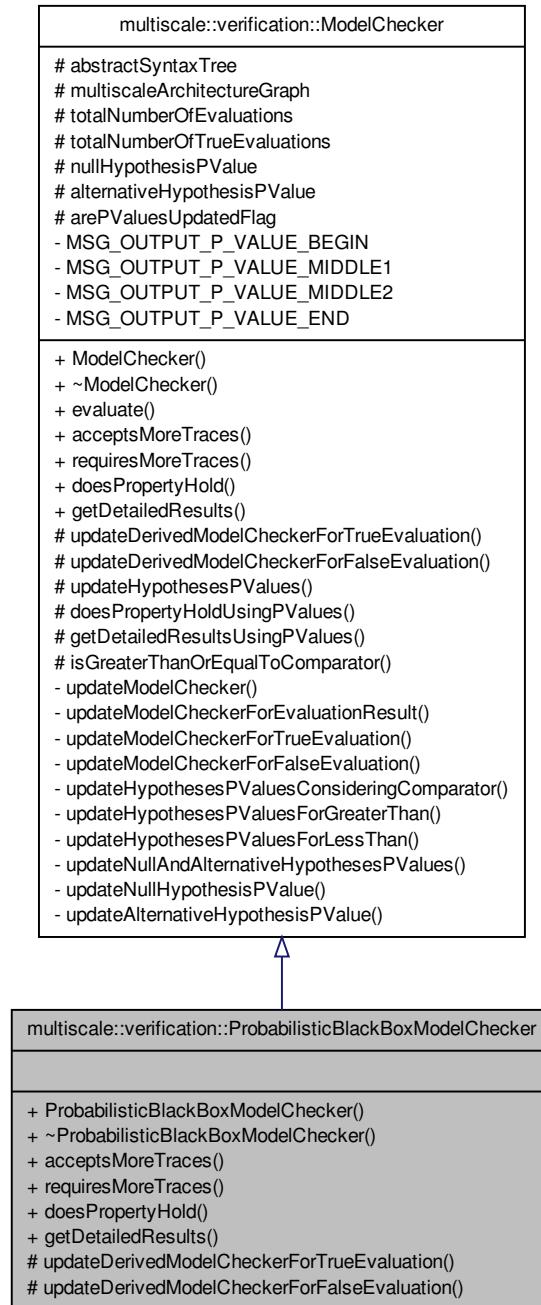
- PrimarySpatialMeasureCollectionAttribute.hpp

## 6.149 multiscale::verification::ProbabilisticBlackBoxModelChecker Class Reference

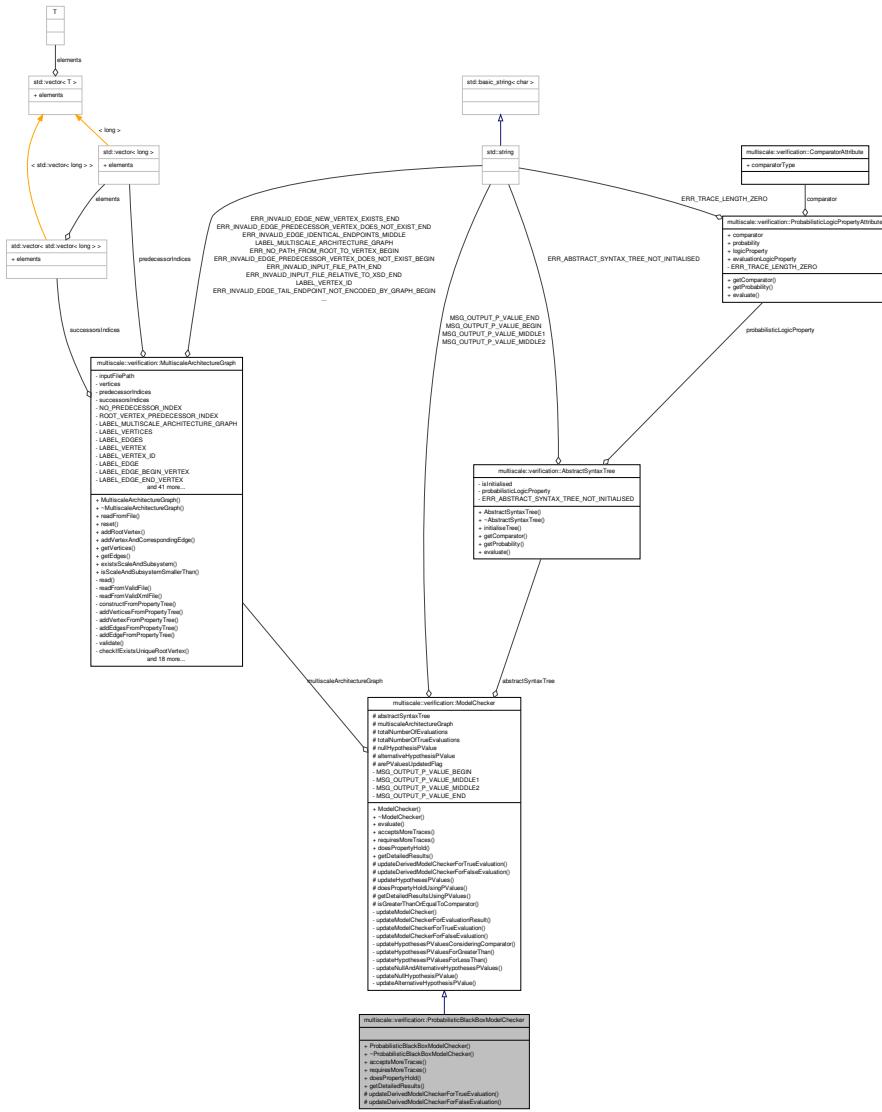
Class used to run probabilistic black-box model checking tasks.

```
#include <ProbabilisticBlackBoxModelChecker.hpp>
```

Inheritance diagram for multiscale::verification::ProbabilisticBlackBoxModelChecker:



Collaboration diagram for multiscale::verification::ProbabilisticBlackBoxModelChecker:



## Public Member Functions

- **ProbabilisticBlackBoxModelChecker** (const AbstractSyntaxTree &abstractSyntaxTree, const MultiscaleArchitectureGraph &multiscaleArchitectureGraph)
- **~ProbabilisticBlackBoxModelChecker** ()
- bool **acceptsMoreTraces** () override

*Check if more traces are accepted for evaluating the logic property.*

- bool **requiresMoreTraces** () override

*Check if more traces are required for evaluating the logic property.*

- bool [doesPropertyHold \(\)](#) override

*Check if the given property holds.*

- std::string [getDetailedResults \(\)](#) override

*Get the detailed description of the results.*

## Protected Member Functions

- void [updateDerivedModelCheckerForTrueEvaluation \(\)](#) override

*Update the results of the derived model checker type considering that the logic property was evaluated to true for the last trace.*

- void [updateDerivedModelCheckerForFalseEvaluation \(\)](#) override

*Update the results of the derived model checker type considering that the logic property was evaluated to false for the last trace.*

### 6.149.1 Detailed Description

Class used to run probabilistic black-box model checking tasks.

The implementation of this class is (partially) based on the algorithms described in the following paper:

H. L. S. Younes, ‘Probabilistic Verification for “Black-Box” Systems’, in Computer Aided Verification, K. Etessami and S. K. Rajamani, Eds. Springer Berlin Heidelberg, 2005, pp. 253–265.

Definition at line 21 of file ProbabilisticBlackBoxModelChecker.hpp.

### 6.149.2 Constructor & Destructor Documentation

#### 6.149.2.1 ProbabilisticBlackBoxModelChecker::ProbabilisticBlackBoxModelChecker ( const AbstractSyntaxTree & abstractSyntaxTree, const MultiscaleArchitectureGraph & multiscaleArchitectureGraph )

Definition at line 12 of file ProbabilisticBlackBoxModelChecker.cpp.

#### 6.149.2.2 ProbabilisticBlackBoxModelChecker::~ProbabilisticBlackBoxModelChecker ( )

Definition at line 20 of file ProbabilisticBlackBoxModelChecker.cpp.

### 6.149.3 Member Function Documentation

6.149.3.1 **bool ProbabilisticBlackBoxModelChecker::acceptsMoreTraces( )**  
[override, virtual]

Check if more traces are accepted for evaluating the logic property.

Implements [multiscale::verification::ModelChecker](#).

Definition at line 22 of file ProbabilisticBlackBoxModelChecker.cpp.

6.149.3.2 **bool ProbabilisticBlackBoxModelChecker::doesPropertyHold( )**  
[override, virtual]

Check if the given property holds.

Implements [multiscale::verification::ModelChecker](#).

Definition at line 30 of file ProbabilisticBlackBoxModelChecker.cpp.

References [multiscale::verification::ModelChecker::doesPropertyHoldUsingPValues\(\)](#).

6.149.3.3 **std::string ProbabilisticBlackBoxModelChecker::getDetailedResults( )**  
[override, virtual]

Get the detailed description of the results.

Implements [multiscale::verification::ModelChecker](#).

Definition at line 34 of file ProbabilisticBlackBoxModelChecker.cpp.

References [multiscale::verification::ModelChecker::getDetailedResultsUsingPValues\(\)](#).

6.149.3.4 **bool ProbabilisticBlackBoxModelChecker::requiresMoreTraces( )**  
[override, virtual]

Check if more traces are required for evaluating the logic property.

Implements [multiscale::verification::ModelChecker](#).

Definition at line 26 of file ProbabilisticBlackBoxModelChecker.cpp.

6.149.3.5 **void ProbabilisticBlackBoxModelChecker::updateDerivedModel-  
CheckerForFalseEvaluation( )** [override, protected,  
virtual]

Update the results of the derived model checker type considering that the logic property was evaluated to false for the last trace.

Do not do anything

Implements [multiscale::verification::ModelChecker](#).

Definition at line 40 of file ProbabilisticBlackBoxModelChecker.cpp.

**6.149.3.6 void ProbabilisticBlackBoxModelChecker::updateDerivedModel-  
CheckerForTrueEvaluation( ) [override, protected,  
virtual]**

Update the results of the derived model checker type considering that the logic property was evaluated to true for the last trace.

Do not do anything

Implements [multiscale::verification::ModelChecker](#).

Definition at line 38 of file ProbabilisticBlackBoxModelChecker.cpp.

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

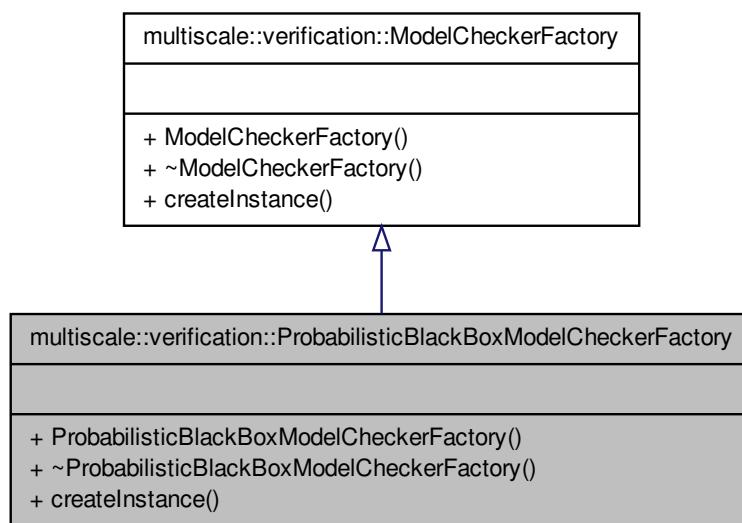
- ProbabilisticBlackBoxModelChecker.hpp
- ProbabilisticBlackBoxModelChecker.cpp

## **6.150 multiscale::verification::ProbabilisticBlackBoxModelChecker- Factory Class Reference**

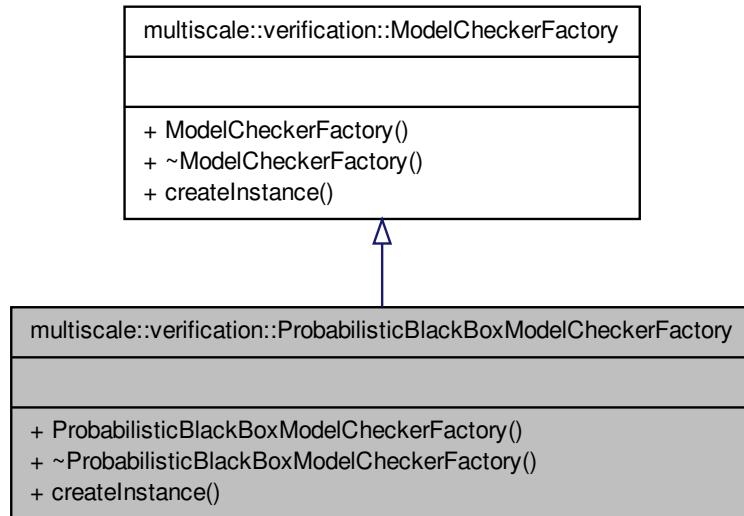
Class for creating [ProbabilisticBlackBoxModelChecker](#) instances.

```
#include <ProbabilisticBlackBoxModelCheckerFactory.hpp>
```

Inheritance diagram for multiscale::verification::ProbabilisticBlackBoxModelChecker-Factory:



Collaboration diagram for multiscale::verification::ProbabilisticBlackBoxModelCheckerFactory:



## Public Member Functions

- [ProbabilisticBlackBoxModelCheckerFactory \(\)](#)
- [~ProbabilisticBlackBoxModelCheckerFactory \(\)](#)
- [std::shared\\_ptr< ModelChecker > createInstance \(const AbstractSyntaxTree &abstractSyntaxTree, const MultiscaleArchitectureGraph &multiscaleArchitectureGraph\) override](#)

*Create an instance of ProbabilisticBlackBoxModelChecker.*

### 6.150.1 Detailed Description

Class for creating [ProbabilisticBlackBoxModelChecker](#) instances.

Definition at line 12 of file [ProbabilisticBlackBoxModelCheckerFactory.hpp](#).

### 6.150.2 Constructor & Destructor Documentation

**6.150.2.1 ProbabilisticBlackBoxModelCheckerFactory::ProbabilisticBlackBoxModelCheckerFactory( )**

Definition at line 7 of file ProbabilisticBlackBoxModelCheckerFactory.cpp.

**6.150.2.2 ProbabilisticBlackBoxModelCheckerFactory::~ProbabilisticBlackBoxModelCheckerFactory( )**

Definition at line 9 of file ProbabilisticBlackBoxModelCheckerFactory.cpp.

**6.150.3 Member Function Documentation**

**6.150.3.1 std::shared\_ptr< ModelChecker > ProbabilisticBlackBoxModelCheckerFactory::createInstance ( const AbstractSyntaxTree & *abstractSyntaxTree*, const MultiscaleArchitectureGraph & *multiscaleArchitectureGraph* ) [override, virtual]**

Create an instance of [ProbabilisticBlackBoxModelChecker](#).

**Parameters**

|                                    |                                                                                                   |
|------------------------------------|---------------------------------------------------------------------------------------------------|
| <i>abstractSyntaxTree</i>          | The abstract syntax tree representing the logic property to be checked                            |
| <i>multiscaleArchitectureGraph</i> | The multiscale architecture graph encoding the hierarchical organization of scales and subsystems |

Implements [multiscale::verification::ModelCheckerFactory](#).

Definition at line 12 of file ProbabilisticBlackBoxModelCheckerFactory.cpp.

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

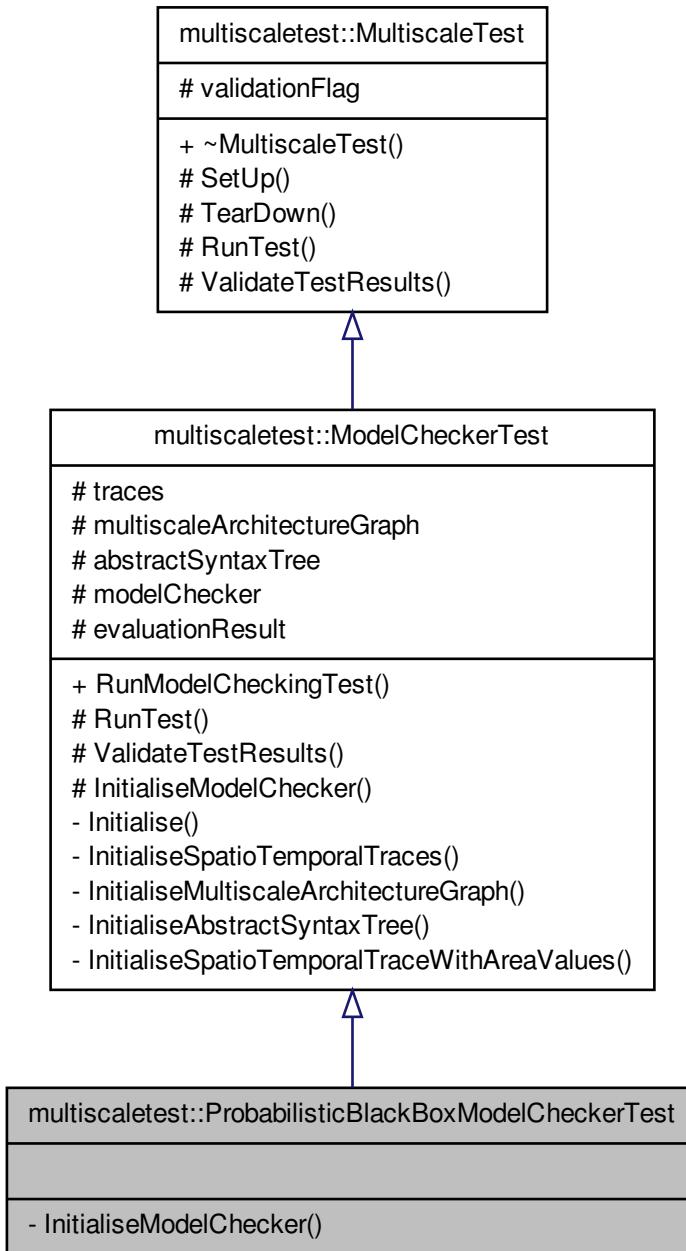
- ProbabilisticBlackBoxModelCheckerFactory.hpp
- ProbabilisticBlackBoxModelCheckerFactory.cpp

**6.151 multiscaletest::ProbabilisticBlackBoxModelCheckerTest - Class Reference**

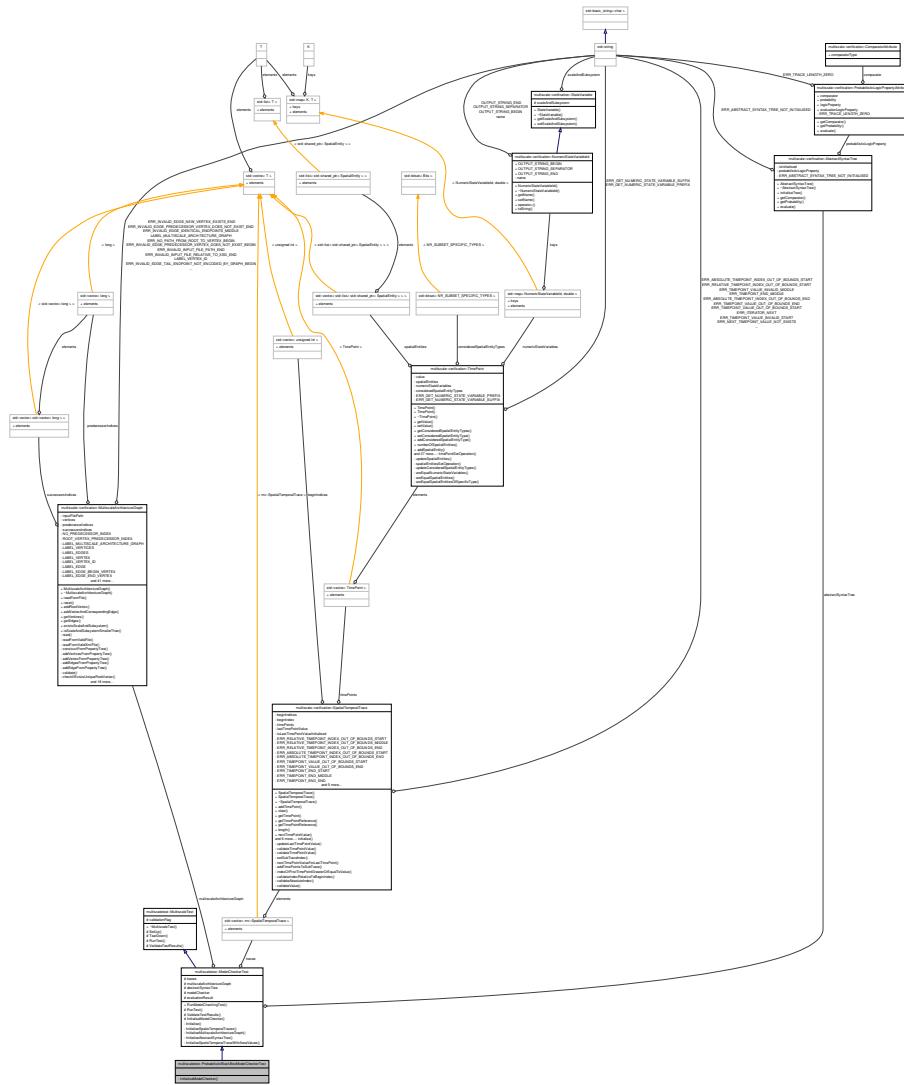
Class for testing the probabilistic black-box model checker.

```
#include <ProbabilisticBlackBoxModelCheckerTest.hpp>
```

Inheritance diagram for multiscaletest::ProbabilisticBlackBoxModelCheckerTest:



## Collaboration diagram for multiscaletest::ProbabilisticBlackBoxModelCheckerTest:



## Private Member Functions

- void `InitialiseModelChecker ()` override  
*Initialise the model checker.*

### **6.151.1 Detailed Description**

Class for testing the probabilistic black-box model checker.

Definition at line 15 of file ProbabilisticBlackBoxModelCheckerTest.hpp.

### 6.151.2 Member Function Documentation

```
6.151.2.1 void multiscaletest::ProbabilisticBlackBoxModelCheckerTest-
           ::InitialiseModelChecker( ) [override, private,
           virtual]
```

Initialise the model checker.

Implements [multiscaletest::ModelCheckerTest](#).

Definition at line 25 of file ProbabilisticBlackBoxModelCheckerTest.hpp.

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

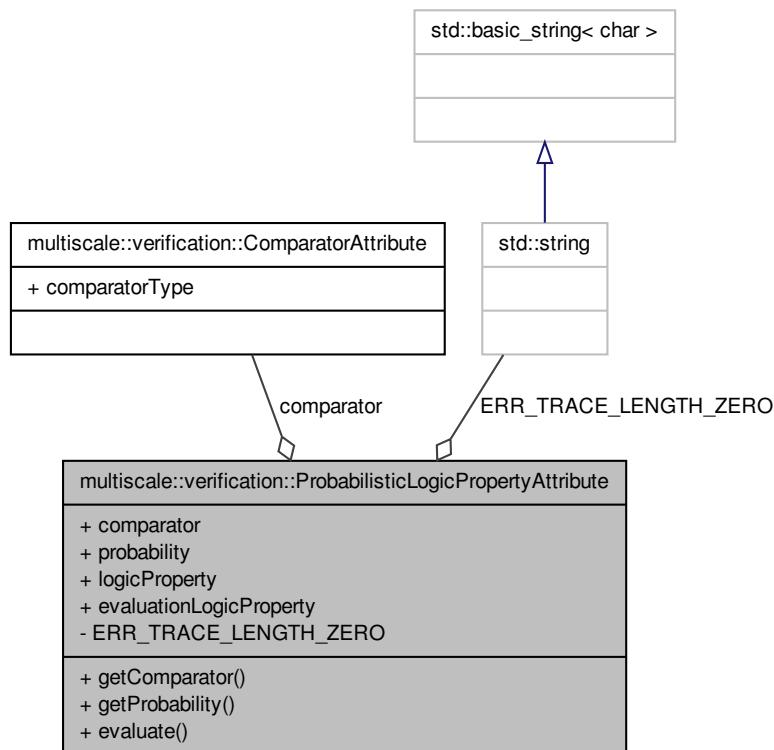
- ProbabilisticBlackBoxModelCheckerTest.hpp

## 6.152 multiscale::verification::ProbabilisticLogicPropertyAttribute Class Reference

Class for representing a probabilistic logic property attribute.

```
#include <ProbabilisticLogicPropertyAttribute.hpp>
```

Collaboration diagram for multiscale::verification::ProbabilisticLogicPropertyAttribute:



## Public Member Functions

- [ComparatorType getComparator \(\)](#)  
*Get the type of the comparator.*
- [double getProbability \(\)](#)  
*Get the probability.*
- [bool evaluate \(const SpatialTemporalTrace &trace, const MultiscaleArchitectureGraph &multiscaleArchitectureGraph\)](#)  
*Evaluate the logic property considering the given trace and multiscale architecture graph.*

## Public Attributes

- [ComparatorAttribute comparator](#)

- double probability
- [LogicPropertyAttributeType logicProperty](#)
- [LogicPropertyAttributeType evaluationLogicProperty](#)

### Static Private Attributes

- static const std::string [ERR\\_TRACE\\_LENGTH\\_ZERO](#) = "The length of the trace provided for evaluating the probabilistic logic property is zero. Please provide a trace which contains at least one timepoint."

#### 6.152.1 Detailed Description

Class for representing a probabilistic logic property attribute.

Definition at line 19 of file ProbabilisticLogicPropertyAttribute.hpp.

#### 6.152.2 Member Function Documentation

##### 6.152.2.1 bool ProbabilisticLogicPropertyAttribute::evaluate ( const SpatialTemporalTrace & trace, const MultiscaleArchitectureGraph & multiscaleArchitectureGraph )

Evaluate the logic property considering the given trace and multiscale architecture graph.

###### Parameters

|                                      |                                              |
|--------------------------------------|----------------------------------------------|
| <i>trace</i>                         | The considered trace (i.e. timeseries data)  |
| <i>multiscale-Architecture-Graph</i> | The considered multiscale architecture graph |

Definition at line 15 of file ProbabilisticLogicPropertyAttribute.cpp.

References [ERR\\_TRACE\\_LENGTH\\_ZERO](#), [evaluationLogicProperty](#), [multiscale::verification::SpatialTemporalTrace::length\(\)](#), and [logicProperty](#).

Referenced by [multiscale::verification::AbstractSyntaxTree::evaluate\(\)](#).

##### 6.152.2.2 ComparatorType ProbabilisticLogicPropertyAttribute::getComparator ( )

Get the type of the comparator.

Definition at line 7 of file ProbabilisticLogicPropertyAttribute.cpp.

References [comparator](#), and [multiscale::verification::ComparatorAttribute::comparatorType](#).

Referenced by multiscale::verification::AbstractSyntaxTree::getComparator().

**6.152.2.3 double ProbabilisticLogicPropertyAttribute::getProbability( )**

Get the probability.

Definition at line 11 of file ProbabilisticLogicPropertyAttribute.cpp.

References probability.

Referenced by multiscale::verification::AbstractSyntaxTree::getProbability().

**6.152.3 Member Data Documentation**

**6.152.3.1 ComparatorAttribute multiscale::verification::ProbabilisticLogicPropertyAttribute::comparator**

The comparator

Definition at line 23 of file ProbabilisticLogicPropertyAttribute.hpp.

Referenced by getComparator().

**6.152.3.2 const std::string ProbabilisticLogicPropertyAttribute::ERR\_TRACE\_LENGTH\_ZERO = "The length of the trace provided for evaluating the probabilistic logic property is zero. Please provide a trace which contains at least one timepoint."**  
[static, private]

Definition at line 49 of file ProbabilisticLogicPropertyAttribute.hpp.

Referenced by evaluate().

**6.152.3.3 LogicPropertyAttributeType multiscale::verification::ProbabilisticLogicPropertyAttribute::evaluationLogicProperty**

The logic property used only for evaluation purposes

Definition at line 27 of file ProbabilisticLogicPropertyAttribute.hpp.

Referenced by evaluate().

**6.152.3.4 LogicPropertyAttributeType multiscale::verification::ProbabilisticLogicPropertyAttribute::logicProperty**

The logic property

Definition at line 25 of file ProbabilisticLogicPropertyAttribute.hpp.

Referenced by evaluate().

### 6.152.3.5 double multiscale::verification::ProbabilisticLogicPropertyAttribute- ::probability

The probability

Definition at line 24 of file ProbabilisticLogicPropertyAttribute.hpp.

Referenced by getProbability().

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

- ProbabilisticLogicPropertyAttribute.hpp
- ProbabilisticLogicPropertyAttribute.cpp

## 6.153 multiscale::verification::ProbabilityErrorHandler Struct - Reference

Structure for defining the error handler for invalid probability errors.

```
#include <ProbabilityErrorHandler.hpp>
```

### Classes

- struct [result](#)

*Structure for specifying the type of the result.*

### Public Member Functions

- template<typename Iterator >  
void [operator\(\)](#) (qi::info const &expectedToken, Iterator errorPosition, Iterator last)  
const

*Overloaded operator.*

### Private Member Functions

- std::string [getExpectedTokenAsString](#) (qi::info const &expectedToken) const  
*Convert the expected token to a string.*

### 6.153.1 Detailed Description

Structure for defining the error handler for invalid probability errors.

Definition at line 17 of file ProbabilityErrorHandler.hpp.

### 6.153.2 Member Function Documentation

6.153.2.1 `std::string multiscale::verification::ProbabilityErrorHandler::get-ExpectedTokenAsString ( qi::info const & expectedToken ) const [inline, private]`

Convert the expected token to a string.

Convert the expected token to a string and remove enclosing quotes

#### Parameters

|                                  |                                   |
|----------------------------------|-----------------------------------|
| <code>expected-<br/>Token</code> | The expected token (not a string) |
|----------------------------------|-----------------------------------|

Definition at line 46 of file ProbabilityErrorHandler.hpp.

Referenced by operator()().

6.153.2.2 `template<typename Iterator > void multiscale::verification::ProbabilityErrorHandler-::operator() ( qi::info const & expectedToken, Iterator errorPosition, Iterator last ) const [inline]`

Overloaded operator.

#### Parameters

|                                  |                                           |
|----------------------------------|-------------------------------------------|
| <code>expected-<br/>Token</code> | The expected token                        |
| <code>errorPosition</code>       | Iterator pointing to the error position   |
| <code>last</code>                | Iterator pointing to the end of the query |

Definition at line 32 of file ProbabilityErrorHandler.hpp.

References getExpectedTokenAsString().

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

- ProbabilityErrorHandler.hpp

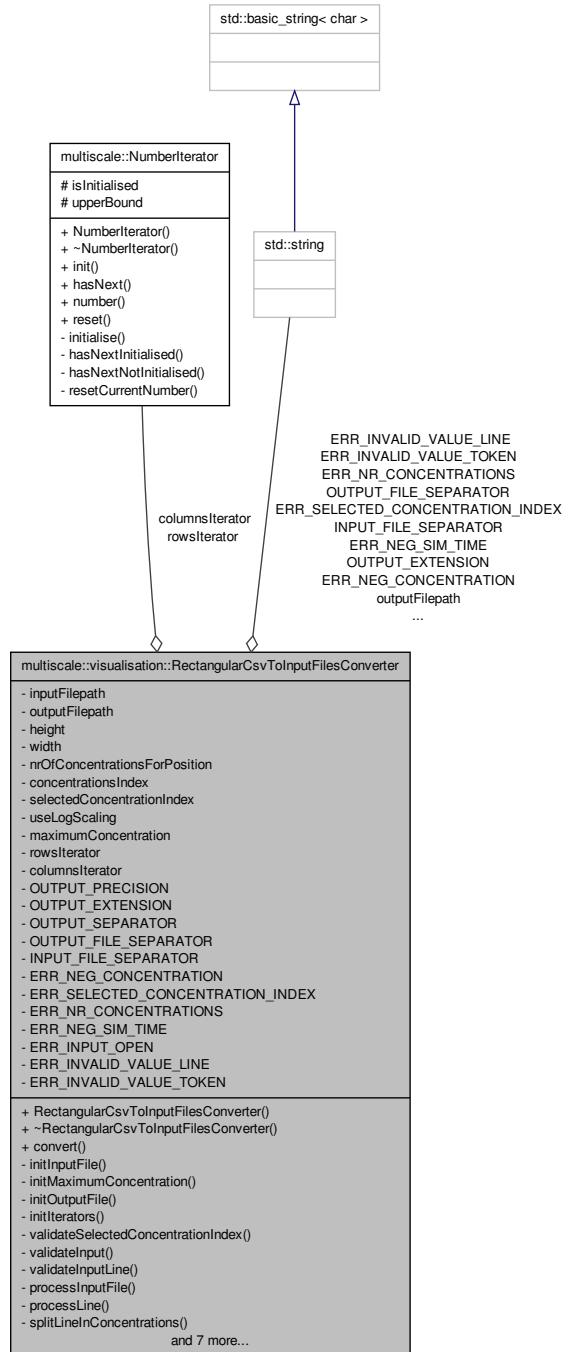
## 6.154 multiscale::visualisation::RectangularCsvToInputFiles-Converter Class Reference

Csv file to input file converter considering cartesian coordinates.

```
#include <RectangularCsvToInputFilesConverter.hpp>
```

Collaboration diagram for multiscale::visualisation::RectangularCsvToInputFiles-

Converter:



## Public Member Functions

- `RectangularCsvToInputFilesConverter` (const std::string &`inputfilepath`, const std::string &`outputfilepath`, unsigned int `height`, unsigned int `width`, unsigned int `nrOfConcentrationsForPosition`, unsigned int `selectedConcentrationIndex`, bool `useLogScaling`, NumberIteratorType `numberIteratorType`)
- `~RectangularCsvToInputFilesConverter` ()
- void `convert` ()

*Start the conversion.*

## Private Member Functions

- void `initInputFile` (std::ifstream &`fin`)  
*Initialise the input file stream over the given input file.*
- void `initMaximumConcentration` (std::ifstream &`fin`)  
*Compute the value of member maximum concentration.*
- void `initOutputFile` (std::ofstream &`fout`, unsigned int `index`, double &`simulationTime`)  
*Initialise the output file with the given index and simulation time.*
- void `initIterators` (const NumberIteratorType &`numberIteratorType`)  
*Initialise the iterators considering the given number iterator type.*
- void `validateSelectedConcentrationIndex` ()  
*Validate the selected concentration index in case of more than one concentration for each position.*
- void `validateInput` (std::ifstream &`fin`)  
*Validate the input.*
- void `validateInputLine` (const std::string &`line`, unsigned int `lineNumber`)  
*Validate the provided line identified by a line number.*
- void `processInputFile` (std::ifstream &`fin`)  
*Process the input file.*
- void `processLine` (const std::string &`line`, unsigned int `outputIndex`)  
*Process the provided line.*
- std::vector< double > `splitLineInConcentrations` (const std::string &`line`, double &`simulationTime`)  
*Split the line in concentrations.*
- void `splitLineInConcentrations` (std::vector< double > &`concentrations`, std::vector< std::string > &`tokens`, unsigned int `rowIndex`)  
*Split line into concentrations.*
- double `computeSimulationTime` (const std::string &`token`)  
*Compute the simulation time from the given token and check if it is valid.*
- double `computeNextPositionConcentration` (int `concentrationIndex`, std::vector< std::string > &`tokens`)  
*Compute the concentration for the next position.*
- double `computeConcentration` (const std::string &`concentration`)

- `double computeNonScaledConcentration (const std::string &concentration)`  
*Compute the concentration from the given string.*
- `double computeScaledConcentration (const std::string &concentration)`  
*Compute the non-scaled concentration from the given string.*
- `double computeNormalisedConcentration (double concentration)`  
*Compute the scaled concentration from the given string.*
- `void updateMaximumConcentration (const std::string &line, double &maximumConcentration)`  
*Normalise the given concentration by dividing it to the maximum concentration.*
- `Update the maximum concentration if the values from the given line are greater than it.`

### Private Attributes

- `std::string inputfilepath`
- `std::string outputfilepath`
- `unsigned int height`
- `unsigned int width`
- `unsigned int nrOfConcentrationsForPosition`
- `unsigned int concentrationsIndex`
- `unsigned int selectedConcentrationIndex`
- `bool useLogScaling`
- `double maximumConcentration`
- `NumberIterator * rowsIterator`
- `NumberIterator * columnsIterator`

### Static Private Attributes

- `static const int OUTPUT_PRECISION = std::numeric_limits<double>::max_digits10`
- `static const std::string OUTPUT_EXTENSION = ".in"`
- `static const std::string OUTPUT_SEPARATOR = "`
- `static const std::string OUTPUT_FILE_SEPARATOR = "_"`
- `static const std::string INPUT_FILE_SEPARATOR = ","`
- `static const std::string ERR_NEG_CONCENTRATION = "All concentrations must be non-negative."`
- `static const std::string ERR_SELECTED_CONCENTRATION_INDEX = "The selected concentration index (0-based indexing) should be smaller than the number of concentrations."`
- `static const std::string ERR_NR_CONCENTRATIONS = "The number of concentrations in the input file does not match the values of the input parameters height and width."`
- `static const std::string ERR_NEG_SIM_TIME = "The simulation time must be non-negative."`
- `static const std::string ERR_INPUT_OPEN = "The input file could not be opened."`
- `static const std::string ERR_INVALID_VALUE_LINE = "Invalid value on line: "`
- `static const std::string ERR_INVALID_VALUE_TOKEN = ", value: "`

### 6.154.1 Detailed Description

Csv file to input file converter considering cartesian coordinates.

Definition at line 16 of file RectangularCsvToInputFilesConverter.hpp.

### 6.154.2 Constructor & Destructor Documentation

**6.154.2.1 RectangularCsvToInputFilesConverter::RectangularCsvToInputFilesConverter ( const std::string & *inputFilepath*, const std::string & *outputFilepath*, unsigned int *height*, unsigned int *width*, unsigned int *nrOfConcentrationsForPosition*, unsigned int *selectedConcentrationIndex*, bool *useLogScaling*, NumberIteratorType *numberIteratorType* )**

Definition at line 21 of file RectangularCsvToInputFilesConverter.cpp.

**6.154.2.2 RectangularCsvToInputFilesConverter::~RectangularCsvToInputFilesConverter ( )**

Definition at line 45 of file RectangularCsvToInputFilesConverter.cpp.

### 6.154.3 Member Function Documentation

**6.154.3.1 double RectangularCsvToInputFilesConverter::computeConcentration ( const std::string & *concentration* ) [private]**

Compute the concentration from the given string.

#### Parameters

|                      |                                       |
|----------------------|---------------------------------------|
| <i>concentration</i> | String representing the concentration |
|----------------------|---------------------------------------|

Definition at line 288 of file RectangularCsvToInputFilesConverter.cpp.

**6.154.3.2 double RectangularCsvToInputFilesConverter::computeNextPositionConcentration ( int *concentrationIndex*, std::vector< std::string > & *tokens* ) [private]**

Compute the concentration for the next position.

#### Parameters

|                           |                                                              |
|---------------------------|--------------------------------------------------------------|
| <i>concentrationIndex</i> | Index of the current concentration from the vector of tokens |
| <i>tokens</i>             | Vector of tokens                                             |

Definition at line 262 of file RectangularCsvToInputFilesConverter.cpp.

References multiscale::Numeric::division().

**6.154.3.3 double RectangularCsvToInputFilesConverter::compute-  
NonScaledConcentration ( const std::string & *concentration* )  
[private]**

Compute the non-scaled concentration from the given string.

**Parameters**

|                            |                                       |
|----------------------------|---------------------------------------|
| <i>concentra-<br/>tion</i> | String representing the concentration |
|----------------------------|---------------------------------------|

Definition at line 294 of file RectangularCsvToInputFilesConverter.cpp.

**6.154.3.4 double RectangularCsvToInputFilesConverter::compute-  
NormalisedConcentration ( double *concentration* )  
[private]**

Normalise the given concentration by dividing it to the maximum concentration.

**Parameters**

|                            |                   |
|----------------------------|-------------------|
| <i>concentra-<br/>tion</i> | The concentration |
|----------------------------|-------------------|

Definition at line 310 of file RectangularCsvToInputFilesConverter.cpp.

**6.154.3.5 double RectangularCsvToInputFilesConverter::compute-  
ScaledConcentration ( const std::string & *concentration* )  
[private]**

Compute the scaled concentration from the given string.

Compute the scaled concentration from the given string by applying a logit transformation to it

**Parameters**

|                            |                                       |
|----------------------------|---------------------------------------|
| <i>concentra-<br/>tion</i> | String representing the concentration |
|----------------------------|---------------------------------------|

Definition at line 298 of file RectangularCsvToInputFilesConverter.cpp.

**6.154.3.6 double RectangularCsvToInputFilesConverter::computeSimulationTime ( const std::string & *token* ) [private]**

Compute the simulation time from the given token and check if it is valid.

**Parameters**

|              |                |
|--------------|----------------|
| <i>token</i> | Token (string) |
|--------------|----------------|

Definition at line 252 of file RectangularCsvToInputFilesConverter.cpp.

**6.154.3.7 void RectangularCsvToInputFilesConverter::convert ( )**

Start the conversion.

Definition at line 50 of file RectangularCsvToInputFilesConverter.cpp.

**6.154.3.8 void RectangularCsvToInputFilesConverter::initInputModule ( std::ifstream & *fin* ) [private]**

Initialise the input file stream over the given input file.

**Parameters**

|            |                   |
|------------|-------------------|
| <i>fin</i> | Input file stream |
|------------|-------------------|

Definition at line 63 of file RectangularCsvToInputFilesConverter.cpp.

**6.154.3.9 void RectangularCsvToInputFilesConverter::initIterators ( const NumberIteratorType & *numberIteratorType* ) [private]**

Initialise the iterators considering the given number iterator type.

**Parameters**

|                                 |                                 |
|---------------------------------|---------------------------------|
| <i>number-<br/>IteratorType</i> | The type of the number iterator |
|---------------------------------|---------------------------------|

Definition at line 113 of file RectangularCsvToInputFilesConverter.cpp.

References multiscale::LEXICOGRAPHIC, and multiscale::STANDARD.

**6.154.3.10 void RectangularCsvToInputFilesConverter::initMaximum-  
Concentration ( std::ifstream & *fin* ) [private]**

Compute the value of member maximum concentration.

**Parameters**

|            |                   |
|------------|-------------------|
| <i>fin</i> | Input file stream |
|------------|-------------------|

Definition at line 71 of file RectangularCsvToInputFilesConverter.cpp.

**6.154.3.11 void RectangularCsvToInputFilesConverter::initOutputFile ( std::ofstream & *fout*, unsigned int *index*, double & *simulationTime* ) [private]**

Initialise the output file with the given index and simulation time.

**Parameters**

|                             |                          |
|-----------------------------|--------------------------|
| <i>fout</i>                 | Output file stream       |
| <i>index</i>                | Index of the output file |
| <i>simulation-<br/>Time</i> | Simulation time          |

Definition at line 95 of file RectangularCsvToInputFilesConverter.cpp.

References multiscale::StringManipulator::toString().

**6.154.3.12 void RectangularCsvToInputFilesConverter::processInputFile ( std::ifstream & *fin* ) [private]**

Process the input file.

Read the concentrations and normalise them if it is the case.

**Parameters**

|            |                   |
|------------|-------------------|
| <i>fin</i> | Input file stream |
|------------|-------------------|

Definition at line 181 of file RectangularCsvToInputFilesConverter.cpp.

**6.154.3.13 void RectangularCsvToInputFilesConverter::processLine ( const std::string & *line*, unsigned int *outputIndex* ) [private]**

Process the provided line.

**Parameters**

|                    |                                                 |
|--------------------|-------------------------------------------------|
| <i>line</i>        | Line                                            |
| <i>outputIndex</i> | Index integrated in the name of the output file |

Definition at line 196 of file RectangularCsvToInputFilesConverter.cpp.

**6.154.3.14 std::vector< double > RectangularCsvToInputFilesConverter::splitLineInConcentrations ( const std::string & *line*, double & *simulationTime* ) [private]**

Split the line in concentrations.

**Parameters**

|                        |                                          |
|------------------------|------------------------------------------|
| <i>line</i>            | Line                                     |
| <i>simulation-Time</i> | Simulation time associated with the line |

Definition at line 215 of file RectangularCsvToInputFilesConverter.cpp.

References multiscale::StringManipulator::split().

**6.154.3.15 void RectangularCsvToInputFilesConverter::splitLineInConcentrations ( std::vector< double > & *concentrations*, std::vector< std::string > & *tokens*, unsigned int *rowIndex* ) [private]**

Split line into concentrations.

**Parameters**

|                       |                                      |
|-----------------------|--------------------------------------|
| <i>concentrations</i> | Concentrations extracted from tokens |
| <i>tokens</i>         | Tokens representing the line         |
| <i>rowIndex</i>       | Index of the current row             |

Definition at line 237 of file RectangularCsvToInputFilesConverter.cpp.

**6.154.3.16 void RectangularCsvToInputFilesConverter::updateMaximumConcentration ( const std::string & *line*, double & *maximumConcentration* ) [private]**

Update the maximum concentration if the values from the given line are greater than it.

**Parameters**

|                              |                           |
|------------------------------|---------------------------|
| <i>line</i>                  | Line from input file      |
| <i>maximum-Concentration</i> | The maximum concentration |

Definition at line 314 of file RectangularCsvToInputFilesConverter.cpp.

---

**6.154.3.17 void RectangularCsvToInputFilesConverter::validateInput ( std::ifstream & *fin* ) [private]**

Validate the input.

**Parameters**

|            |                   |
|------------|-------------------|
| <i>fin</i> | Input file stream |
|------------|-------------------|

Definition at line 136 of file RectangularCsvToInputFilesConverter.cpp.

**6.154.3.18 void RectangularCsvToInputFilesConverter::validateInputLine ( const std::string & *line*, unsigned int *lineNumber* ) [private]**

Validate the provided line identified by a line number.

**Parameters**

|                   |                      |
|-------------------|----------------------|
| <i>line</i>       | Line from input file |
| <i>lineNumber</i> | Number of the line   |

Definition at line 160 of file RectangularCsvToInputFilesConverter.cpp.

References multiscale::StringManipulator::split().

**6.154.3.19 void RectangularCsvToInputFilesConverter::validateSelectedConcentrationIndex ( ) [private]**

Validate the selected concentration index in case of more than one concentration for each position.

Definition at line 130 of file RectangularCsvToInputFilesConverter.cpp.

#### 6.154.4 Member Data Documentation

**6.154.4.1 NumberIterator\* multiscale::visualisation::RectangularCsvToInputFilesConverter::columnsIterator [private]**

Iterator over the number of columns

Definition at line 42 of file RectangularCsvToInputFilesConverter.hpp.

**6.154.4.2 unsigned int multiscale::visualisation::RectangularCsvToInputFilesConverter::concentrationsIndex [private]**

Index of the current concentration

Definition at line 27 of file RectangularCsvToInputFilesConverter.hpp.

```
6.154.4.3 const std::string RectangularCsvToInputFilesConverter::ERR_-  
    INPUT_OPEN = "The input file could not be opened." [static,  
    private]
```

Definition at line 198 of file RectangularCsvToInputFilesConverter.hpp.

```
6.154.4.4 const std::string RectangularCsvToInputFilesConverter::ERR_-  
    INVALID_VALUE_LINE = "Invalid value on line: " [static,  
    private]
```

Definition at line 199 of file RectangularCsvToInputFilesConverter.hpp.

```
6.154.4.5 const std::string RectangularCsvToInputFilesConverter::  
    ERR_INVALID_VALUE_TOKEN = ", value: " [static,  
    private]
```

Definition at line 200 of file RectangularCsvToInputFilesConverter.hpp.

```
6.154.4.6 const std::string RectangularCsvToInputFilesConverter::ERR_NEG_CO-  
    NCENTRATION = "All concentrations must be non-negative." [static,  
    private]
```

Definition at line 194 of file RectangularCsvToInputFilesConverter.hpp.

```
6.154.4.7 const std::string RectangularCsvToInputFilesConverter::ERR_NEG_-  
    _SIM_TIME = "The simulation time must be non-negative." [static,  
    private]
```

Definition at line 197 of file RectangularCsvToInputFilesConverter.hpp.

```
6.154.4.8 const std::string RectangularCsvToInputFilesConverter::ERR_NR_CONC-  
    ENTRATIONS = "The number of concentrations in the input file does not match the  
    values of the input parameters height and width." [static, private]
```

Definition at line 196 of file RectangularCsvToInputFilesConverter.hpp.

```
6.154.4.9 const std::string RectangularCsvToInputFilesConverter::ERR_SELECTE-  
    D_CONCENTRATION_INDEX = "The selected concentration index (0-based  
    indexing) should be smaller than the number of concentrations." [static,  
    private]
```

Definition at line 195 of file RectangularCsvToInputFilesConverter.hpp.

---

6.154.4.10 **unsigned int multiscale::visualisation::RectangularCsvToInputFiles-Converter::height** [private]

Height of the grid

Definition at line 23 of file RectangularCsvToInputFilesConverter.hpp.

6.154.4.11 **const std::string RectangularCsvToInputFilesConverter-::INPUT\_FILE\_SEPARATOR = ","** [static, private]

Definition at line 192 of file RectangularCsvToInputFilesConverter.hpp.

6.154.4.12 **std::string multiscale::visualisation::RectangularCsvToInputFiles-Converter::inputFilepath** [private]

Path to the input file

Definition at line 20 of file RectangularCsvToInputFilesConverter.hpp.

6.154.4.13 **double multiscale::visualisation::RectangularCsv-  
ToInputFilesConverter::maximumConcentration**  
[private]

The maximum concentration in the input file

Definition at line 39 of file RectangularCsvToInputFilesConverter.hpp.

6.154.4.14 **unsigned int multiscale::visualisation::RectangularCsv-  
ToInputFilesConverter::nrOfConcentrationsForPosition**  
[private]

Number of concentrations for each position

Definition at line 25 of file RectangularCsvToInputFilesConverter.hpp.

6.154.4.15 **const std::string RectangularCsvToInputFiles-  
Converter::OUTPUT\_EXTENSION = ".in"** [static, private]

Definition at line 189 of file RectangularCsvToInputFilesConverter.hpp.

6.154.4.16 **const std::string RectangularCsvToInputFilesConverter-  
::OUTPUT\_FILE\_SEPARATOR = "\_"** [static, private]

Definition at line 191 of file RectangularCsvToInputFilesConverter.hpp.

**6.154.4.17 const int RectangularCsvToInputFilesConverter::OUTPUT\_PRECISION = std::numeric\_limits<double>::max\_digits10 [static, private]**

Definition at line 187 of file RectangularCsvToInputFilesConverter.hpp.

**6.154.4.18 const std::string RectangularCsvToInputFiles- Converter::OUTPUT\_SEPARATOR = " " [static, private]**

Definition at line 190 of file RectangularCsvToInputFilesConverter.hpp.

**6.154.4.19 std::string multiscale::visualisation::RectangularCsvToInputFiles- Converter::outputFilepath [private]**

Path to the output file

Definition at line 21 of file RectangularCsvToInputFilesConverter.hpp.

**6.154.4.20 NumberIterator\* multiscale::visualisation::- RectangularCsvToInputFilesConverter::rowsIterator [private]**

Iterator over the number of rows

Definition at line 41 of file RectangularCsvToInputFilesConverter.hpp.

**6.154.4.21 unsigned int multiscale::visualisation::RectangularCsv- ToInputFilesConverter::selectedConcentrationIndex [private]**

Index of the concentration A in case the number of concentrations for each position is greater than 1

finalConcentration = A / (A1 + A2 + ... + AN), where N is the number of concentrations for each position

Definition at line 29 of file RectangularCsvToInputFilesConverter.hpp.

**6.154.4.22 bool multiscale::visualisation::RectangularCsvToInputFilesConverter- ::useLogScaling [private]**

Flag for using logarithmic scaling for concentrations

Definition at line 36 of file RectangularCsvToInputFilesConverter.hpp.

6.154.4.23 **unsigned int multiscale::visualisation::RectangularCsvToInputFiles-Converter::width** [private]

Width of the grid

Definition at line 24 of file RectangularCsvToInputFilesConverter.hpp.

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

- RectangularCsvToInputFilesConverter.hpp

- RectangularCsvToInputFilesConverter.cpp

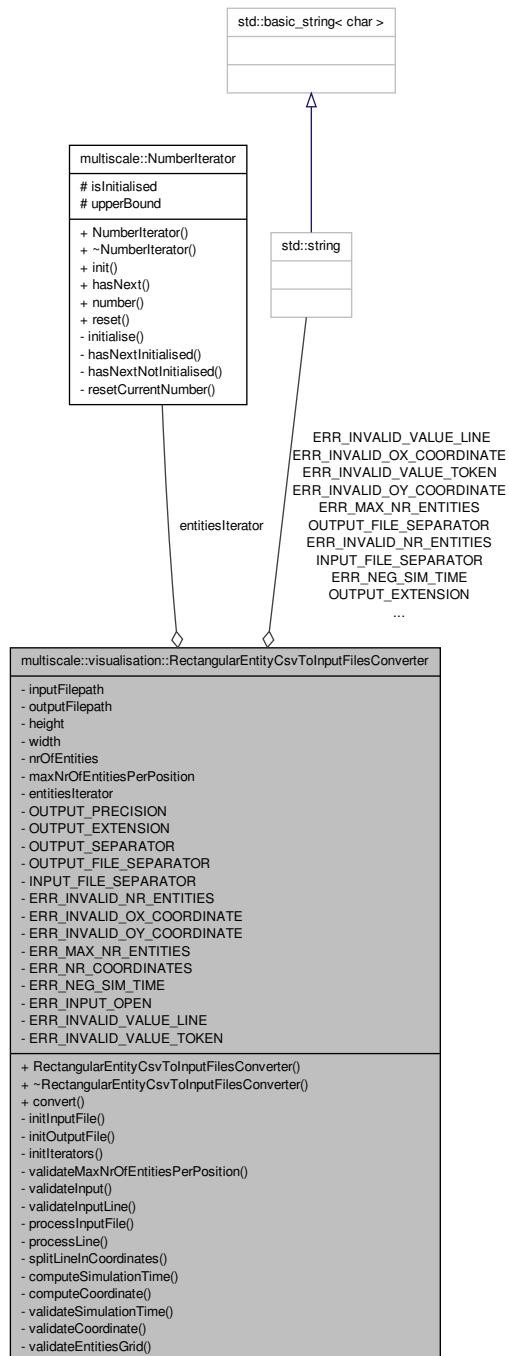
## 6.155 multiscale::visualisation::RectangularEntityCsvToInput-FilesConverter Class Reference

Csv entity file to input file converter considering cartesian coordinates.

```
#include <RectangularEntityCsvToInputFilesConverter.hpp>
```

Collaboration diagram for multiscale::visualisation::RectangularEntityCsvToInputFiles-

Converter:



## Public Member Functions

- `RectangularEntityCsvToInputFilesConverter` (const std::string &`inputFilepath`, const std::string &`outputFilepath`, unsigned int `height`, unsigned int `width`, unsigned int `nrOfEntities`, unsigned int `maxNrOfEntitiesPerPosition`, Number-`IteratorType` `numberIteratorType`)
- `~RectangularEntityCsvToInputFilesConverter` ()
- void `convert` ()

*Start the conversion.*

## Private Member Functions

- void `initInputFile` (std::ifstream &`fin`)
 

*Initialise the input file stream over the given input file.*
- void `initOutputFile` (std::ofstream &`fout`, unsigned int `index`, double &`simulationTime`)
 

*Initialise the output file with the given index and simulation time.*
- void `initIterators` (const Number-`IteratorType` &`numberIteratorType`)
 

*Initialise the iterators considering the given number iterator type.*
- void `validateMaxNrOfEntitiesPerPosition` ()
 

*Check if the maximum number of entities per position is a non-zero natural number.*
- void `validateInput` (std::ifstream &`fin`)
 

*Validate the input.*
- void `validateInputLine` (const std::string &`line`, unsigned int `lineNumber`)
 

*Validate the provided line identified by a line number.*
- void `processInputFile` (std::ifstream &`fin`)
 

*Process the input file.*
- void `processLine` (const std::string &`line`, unsigned int `outputIndex`)
 

*Process the provided line.*
- std::vector< double > `splitLineInCoordinates` (const std::string &`line`, double &`simulationTime`)
 

*Split the line in coordinates.*
- double `computeSimulationTime` (const std::string &`token`)
 

*Compute the simulation time from the given token and check if it is valid.*
- unsigned int `computeCoordinate` (const std::string &`token`, bool `isOxCoordinate`)
 

*Compute the coordinate from the given string and check if it is valid.*
- void `validateSimulationTime` (const std::string &`token`, unsigned int `lineNumber`)
 

*Check if the simulation time is valid.*
- void `validateCoordinate` (const std::string &`token`, unsigned int `lineNumber`, bool `isOxCoordinate`)
 

*Check if the coordinate is valid.*
- void `validateEntitiesGrid` (const std::vector< double > &`entitiesGrid`)
 

*Check if the entities grid contains only values between zero and one.*

### Private Attributes

- std::string `inputFilepath`
- std::string `outputFilepath`
- unsigned int `height`
- unsigned int `width`
- unsigned int `nrOfEntities`
- unsigned int `maxNrOfEntitiesPerPosition`
- `NumberIterator * entitiesIterator`

### Static Private Attributes

- static const int `OUTPUT_PRECISION` = std::numeric\_limits<double>::max\_digits10
- static const std::string `OUTPUT_EXTENSION` = ".in"
- static const std::string `OUTPUT_SEPARATOR` = " "
- static const std::string `OUTPUT_FILE_SEPARATOR` = "\_"
- static const std::string `INPUT_FILE_SEPARATOR` = ","
- static const std::string `ERR_INVALID_NR_ENTITIES` = "The number of entities at the given position is invalid."
- static const std::string `ERR_INVALID_OX_COORDINATE` = "The value of the Ox coordinate is invalid."
- static const std::string `ERR_INVALID_OY_COORDINATE` = "The value of the Oy coordinate is invalid."
- static const std::string `ERR_MAX_NR_ENTITIES` = "The maximum number of entities per grid position is equal to zero."
- static const std::string `ERR_NR_COORDINATES` = "The number of coordinates in the input file does not match the values of the input parameters `height`, `width` and `nrOfEntities`."
- static const std::string `ERR_NEG_SIM_TIME` = "The simulation time must be non-negative."
- static const std::string `ERR_INPUT_OPEN` = "The input file could not be opened."
- static const std::string `ERR_INVALID_VALUE_LINE` = "Invalid value on line: "
- static const std::string `ERR_INVALID_VALUE_TOKEN` = ", value: "

#### 6.155.1 Detailed Description

Csv entity file to input file converter considering cartesian coordinates.

Definition at line 16 of file RectangularEntityCsvToInputFilesConverter.hpp.

#### 6.155.2 Constructor & Destructor Documentation

---

**6.155.2.1 `RectangularEntityCsvToInputFilesConverter::RectangularEntityCsvToInputFilesConverter ( const std::string & inputfilepath, const std::string & outputfilepath, unsigned int height, unsigned int width, unsigned int nrOfEntities, unsigned int maxNrOfEntitiesPerPosition, NumberIteratorType numberIteratorType )`**

Definition at line 20 of file RectangularEntityCsvToInputFilesConverter.cpp.

**6.155.2.2 `RectangularEntityCsvToInputFilesConverter::~RectangularEntityCsvToInputFilesConverter ( )`**

Definition at line 37 of file RectangularEntityCsvToInputFilesConverter.cpp.

### 6.155.3 Member Function Documentation

**6.155.3.1 `unsigned int RectangularEntityCsvToInputFilesConverter::computeCoordinate ( const std::string & token, bool isOxCoordinate ) [private]`**

Compute the coordinate from the given string and check if it is valid.

#### Parameters

|                                   |                                                                      |
|-----------------------------------|----------------------------------------------------------------------|
| <i>token</i>                      | Token (string)                                                       |
| <i>isOx-</i><br><i>Coordinate</i> | Flag which indicates if the coordinate corresponds to Ox axis or not |

Definition at line 213 of file RectangularEntityCsvToInputFilesConverter.cpp.

**6.155.3.2 `double RectangularEntityCsvToInputFilesConverter::computeSimulationTime ( const std::string & token ) [private]`**

Compute the simulation time from the given token and check if it is valid.

#### Parameters

|              |                |
|--------------|----------------|
| <i>token</i> | Token (string) |
|--------------|----------------|

Definition at line 203 of file RectangularEntityCsvToInputFilesConverter.cpp.

**6.155.3.3 `void RectangularEntityCsvToInputFilesConverter::convert ( )`**

Start the conversion.

Definition at line 41 of file RectangularEntityCsvToInputFilesConverter.cpp.

**6.155.3.4 void RectangularEntityCsvToInputFilesConverter::initInputFile ( std::ifstream & *fin* ) [private]**

Initialise the input file stream over the given input file.

**Parameters**

|            |                   |
|------------|-------------------|
| <i>fin</i> | Input file stream |
|------------|-------------------|

Definition at line 53 of file RectangularEntityCsvToInputFilesConverter.cpp.

**6.155.3.5 void RectangularEntityCsvToInputFilesConverter::initIterators ( const NumberIteratorType & *numberIteratorType* ) [private]**

Initialise the iterators considering the given number iterator type.

**Parameters**

|                                 |                                 |
|---------------------------------|---------------------------------|
| <i>number-<br/>IteratorType</i> | The type of the number iterator |
|---------------------------------|---------------------------------|

Definition at line 79 of file RectangularEntityCsvToInputFilesConverter.cpp.

References multiscale::LEXICOGRAPHIC, and multiscale::STANDARD.

**6.155.3.6 void RectangularEntityCsvToInputFilesConverter::initOutputFile ( std::ofstream & *fout*, unsigned int *index*, double & *simulationTime* ) [private]**

Initialise the output file with the given index and simulation time.

**Parameters**

|                             |                          |
|-----------------------------|--------------------------|
| <i>fout</i>                 | Output file stream       |
| <i>index</i>                | Index of the output file |
| <i>simulation-<br/>Time</i> | Simulation time          |

Definition at line 61 of file RectangularEntityCsvToInputFilesConverter.cpp.

References multiscale::StringManipulator::toString().

**6.155.3.7 void RectangularEntityCsvToInputFilesConverter::processInputFile ( std::ifstream & *fin* ) [private]**

Process the input file.

Read the concentrations and normalise them if it is the case.

**Parameters**

|            |                   |
|------------|-------------------|
| <i>fin</i> | Input file stream |
|------------|-------------------|

Definition at line 143 of file RectangularEntityCsvToInputFilesConverter.cpp.

**6.155.3.8 void RectangularEntityCsvToInputFilesConverter::processLine ( const std::string & *line*, unsigned int *outputIndex* ) [private]**

Process the provided line.

**Parameters**

|                    |                                                 |
|--------------------|-------------------------------------------------|
| <i>line</i>        | Line                                            |
| <i>outputIndex</i> | Index integrated in the name of the output file |

Definition at line 158 of file RectangularEntityCsvToInputFilesConverter.cpp.

**6.155.3.9 std::vector< double > RectangularEntityCsvToInputFilesConverter::splitLineInCoordinates ( const std::string & *line*, double & *simulationTime* ) [private]**

Split the line in coordinates.

Split the line in coordinates and return the grid of size height \* width recording the position of the entities. The number of entities per grid position is normalised to the range [0, 1]

**Parameters**

|                        |                                          |
|------------------------|------------------------------------------|
| <i>line</i>            | Line                                     |
| <i>simulation-Time</i> | Simulation time associated with the line |

Definition at line 179 of file RectangularEntityCsvToInputFilesConverter.cpp.

References multiscale::StringManipulator::split().

**6.155.3.10 void RectangularEntityCsvToInputFilesConverter::validateCoordinate ( const std::string & *token*, unsigned int *lineNumber*, bool *isOxCoordinate* ) [private]**

Check if the coordinate is valid.

**Parameters**

|                        |                                                                      |
|------------------------|----------------------------------------------------------------------|
| <i>token</i>           | Token (string)                                                       |
| <i>lineNumber</i>      | Number of the line                                                   |
| <i>isOx-Coordinate</i> | Flag which indicates if the coordinate corresponds to Ox axis or not |

Definition at line 244 of file RectangularEntityCsvToInputFilesConverter.cpp.

**6.155.3.11 void RectangularEntityCsvToInputFilesConverter::validateEntitiesGrid ( const std::vector< double > & entitiesGrid ) [private]**

Check if the entities grid contains only values between zero and one.

Parameters

|                     |                      |
|---------------------|----------------------|
| <i>entitiesGrid</i> | The grid of entities |
|---------------------|----------------------|

Definition at line 260 of file RectangularEntityCsvToInputFilesConverter.cpp.

**6.155.3.12 void RectangularEntityCsvToInputFilesConverter::validateInput ( std::ifstream & fin ) [private]**

Validate the input.

Parameters

|            |                   |
|------------|-------------------|
| <i>fin</i> | Input file stream |
|------------|-------------------|

Definition at line 101 of file RectangularEntityCsvToInputFilesConverter.cpp.

**6.155.3.13 void RectangularEntityCsvToInputFilesConverter::validateInputLine ( const std::string & line, unsigned int lineNumber ) [private]**

Validate the provided line identified by a line number.

Parameters

|                   |                      |
|-------------------|----------------------|
| <i>line</i>       | Line from input file |
| <i>lineNumber</i> | Number of the line   |

Definition at line 125 of file RectangularEntityCsvToInputFilesConverter.cpp.

References multiscale::StringManipulator::split().

**6.155.3.14 void RectangularEntityCsvToInputFilesConverter::validateMaxNrOf- EntitiesPerPosition ( ) [private]**

Check if the maximum number of entities per position is a non-zero natural number.

Definition at line 95 of file RectangularEntityCsvToInputFilesConverter.cpp.

---

```
6.155.3.15 void RectangularEntityCsvToInputFilesConverter::validate-
SimulationTime ( const std::string & token, unsigned int lineNumber )
[private]
```

Check if the simulation time is valid.

**Parameters**

|                   |                    |
|-------------------|--------------------|
| <i>token</i>      | Token (string)     |
| <i>lineNumber</i> | Number of the line |

Definition at line 230 of file RectangularEntityCsvToInputFilesConverter.cpp.

#### 6.155.4 Member Data Documentation

```
6.155.4.1 NumberIterator* multiscale::visualisation::Rectangular-
EntityCsvToInputFilesConverter::entitiesIterator
[private]
```

Iterator over the number of rows

Definition at line 29 of file RectangularEntityCsvToInputFilesConverter.hpp.

```
6.155.4.2 const std::string RectangularEntityCsvToInputFilesConverter::ER-
R_INPUT_OPEN = "The input file could not be opened." [static,
private]
```

Definition at line 158 of file RectangularEntityCsvToInputFilesConverter.hpp.

```
6.155.4.3 const std::string RectangularEntityCsvToInputFilesConverter::ERR_INVA-
LID_NR_ENTITIES = "The number of entities at the given position is invalid."
[static, private]
```

Definition at line 152 of file RectangularEntityCsvToInputFilesConverter.hpp.

```
6.155.4.4 const std::string RectangularEntityCsvToInputFilesConverter::ERR_IN-
VALID_OX_COORDINATE = "The value of the Ox coordinate is invalid."
[static, private]
```

Definition at line 153 of file RectangularEntityCsvToInputFilesConverter.hpp.

```
6.155.4.5 const std::string RectangularEntityCsvToInputFilesConverter::ERR_IN-
VALID_OY_COORDINATE = "The value of the Oy coordinate is invalid."
[static, private]
```

Definition at line 154 of file RectangularEntityCsvToInputFilesConverter.hpp.

```
6.155.4.6 const std::string RectangularEntityCsvToInputFilesConverter::E-
RR_INVALID_VALUE_LINE = "Invalid value on line: " [static,
private]
```

Definition at line 159 of file RectangularEntityCsvToInputFilesConverter.hpp.

```
6.155.4.7 const std::string RectangularEntityCsvToInputFilesConverter-
::ERR_INVALID_VALUE_TOKEN = ", value: " [static,
private]
```

Definition at line 160 of file RectangularEntityCsvToInputFilesConverter.hpp.

```
6.155.4.8 const std::string RectangularEntityCsvToInputFilesConverter::ERR_MAX-
_NR_ENTITIES = "The maximum number of entities per grid position is equal to
zero." [static, private]
```

Definition at line 155 of file RectangularEntityCsvToInputFilesConverter.hpp.

```
6.155.4.9 const std::string RectangularEntityCsvToInputFilesConverter::ERR_N-
EG_SIM_TIME = "The simulation time must be non-negative." [static,
private]
```

Definition at line 157 of file RectangularEntityCsvToInputFilesConverter.hpp.

```
6.155.4.10 const std::string RectangularEntityCsvToInputFilesConverter::ERR_NR_-
COORDINATES = "The number of coordinates in the input file does not match the
values of the input parameters height, width and nrOfEntities." [static,
private]
```

Definition at line 156 of file RectangularEntityCsvToInputFilesConverter.hpp.

```
6.155.4.11 unsigned int multiscale::visualisation::RectangularEntityCsvToInput-
FilesConverter::height [private]
```

Height of the grid

Definition at line 23 of file RectangularEntityCsvToInputFilesConverter.hpp.

```
6.155.4.12 const std::string RectangularEntityCsvToInputFiles-
Converter::INPUT_FILE_SEPARATOR = "," [static,
private]
```

Definition at line 150 of file RectangularEntityCsvToInputFilesConverter.hpp.

---

```
6.155.4.13 std::string multiscale::visualisation::Rectangular-
EntityCsvToInputFilesConverter::filepath
[private]
```

Path to the input file

Definition at line 20 of file RectangularEntityCsvToInputFilesConverter.hpp.

```
6.155.4.14 unsigned int multiscale::visualisation::RectangularEntity-
CsvToInputFilesConverter::maxNrOfEntitiesPerPosition
[private]
```

The maximum number of entities per position

Definition at line 27 of file RectangularEntityCsvToInputFilesConverter.hpp.

```
6.155.4.15 unsigned int multiscale::visualisation::Rectangular-
EntityCsvToInputFilesConverter::nrOfEntities
[private]
```

Number of entities

Definition at line 25 of file RectangularEntityCsvToInputFilesConverter.hpp.

```
6.155.4.16 const std::string RectangularEntityCsvToInputFiles-
Converter::OUTPUT_EXTENSION = ".in" [static,
private]
```

Definition at line 147 of file RectangularEntityCsvToInputFilesConverter.hpp.

```
6.155.4.17 const std::string RectangularEntityCsvToInputFiles-
Converter::OUTPUT_FILE_SEPARATOR = "_" [static,
private]
```

Definition at line 149 of file RectangularEntityCsvToInputFilesConverter.hpp.

```
6.155.4.18 const int RectangularEntityCsvToInputFilesConverter::OUTPUT_P-
RECISION = std::numeric_limits<double>::max_digits10 [static,
private]
```

Definition at line 145 of file RectangularEntityCsvToInputFilesConverter.hpp.

```
6.155.4.19 const std::string RectangularEntityCsvToInputFiles-
Converter::OUTPUT_SEPARATOR = " " [static,
private]
```

Definition at line 148 of file RectangularEntityCsvToInputFilesConverter.hpp.

**6.155.4.20 std::string multiscale::visualisation::RectangularEntityCsvToInputFilesConverter::outputFilepath [private]**

Path to the output file

Definition at line 21 of file RectangularEntityCsvToInputFilesConverter.hpp.

**6.155.4.21 unsigned int multiscale::visualisation::RectangularEntityCsvToInputFilesConverter::width [private]**

Width of the grid

Definition at line 24 of file RectangularEntityCsvToInputFilesConverter.hpp.

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

- RectangularEntityCsvToInputFilesConverter.hpp
  
- RectangularEntityCsvToInputFilesConverter.cpp

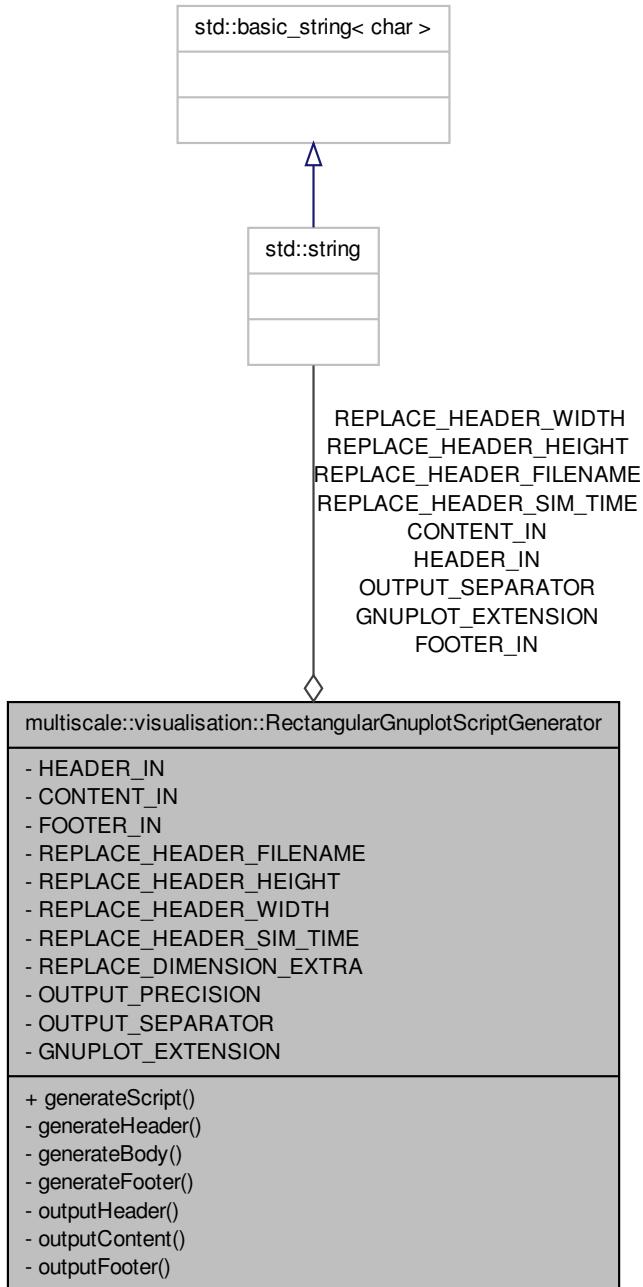
## **6.156 multiscale::visualisation::RectangularGnuplotScriptGenerator Class Reference**

Gnuplot script generator from the provided concentrations considering a rectangular geometry.

```
#include <RectangularGnuplotScriptGenerator.hpp>
```

Collaboration diagram for multiscale::visualisation::RectangularGnuplotScript-

Generator:



### Static Public Member Functions

- static void `generateScript` (const std::vector< double > &concentrations, double simulationTime, unsigned long height, unsigned long width, const std::string &outputFilepath)

*Generate the script.*

### Static Private Member Functions

- static void `generateHeader` (std::ofstream &fout, const std::string &outputFilepath, double simulationTime, unsigned long height, unsigned long width)

*Generate the header of the script.*

- static void `generateBody` (const std::vector< double > &concentrations, unsigned long height, unsigned long width, std::ofstream &fout)

*Generate the body/content of the script.*

- static void `generateFooter` (std::ofstream &fout)

*Generate the footer of the script.*

- static void `outputHeader` (std::ifstream &fin, const std::string &outputFilename, double simulationTime, unsigned long height, unsigned long width, std::ofstream &fout)

*Output the header of the script.*

- static void `outputContent` (const std::vector< double > &concentrations, unsigned long height, unsigned long width, std::ofstream &fout)

*Output the content of the script.*

- static void `outputFooter` (std::ifstream &fin, std::ofstream &fout)

*Output the footer of the script.*

### Static Private Attributes

- static const std::string `HEADER_IN` = "/usr/local/share/mule/config/visualisation/rectangular/header.in"
- static const std::string `CONTENT_IN` = "/usr/local/share/mule/config/visualisation/rectangular/content.in"
- static const std::string `FOOTER_IN` = "/usr/local/share/mule/config/visualisation/rectangular/footer.in"
- static const std::string `REPLACE_HEADER_FILENAME` = "OUTPUT\_FILENAME"
- static const std::string `REPLACE_HEADER_HEIGHT` = "OUTPUT\_DIMENSION\_N1"
- static const std::string `REPLACE_HEADER_WIDTH` = "OUTPUT\_DIMENSION2"
- static const std::string `REPLACE_HEADER_SIM_TIME` = "OUTPUT\_SIM\_TIME"
- static const double `REPLACE_DIMENSION_EXTRA` = 0.5
- static const int `OUTPUT_PRECISION` = std::numeric\_limits<double>::max\_digits10
- static const std::string `OUTPUT_SEPARATOR` = " "
- static const std::string `GNUPLOT_EXTENSION` = ".plt"

### 6.156.1 Detailed Description

Gnuplot script generator from the provided concentrations considering a rectangular geometry.

Definition at line 13 of file RectangularGnuplotScriptGenerator.hpp.

### 6.156.2 Member Function Documentation

**6.156.2.1 void RectangularGnuplotScriptGenerator::generateBody ( const std::vector< double > & concentrations, unsigned long height, unsigned long width, std::ofstream & fout ) [static, private]**

Generate the body/content of the script.

#### Parameters

|                                   |                        |
|-----------------------------------|------------------------|
| <i>concentra-</i><br><i>tions</i> | The concentrations     |
| <i>height</i>                     | The height of the grid |
| <i>width</i>                      | The width of the grid  |
| <i>fout</i>                       | Output file stream     |

Definition at line 45 of file RectangularGnuplotScriptGenerator.cpp.

References CONTENT\_IN, and outputContent().

Referenced by generateScript().

**6.156.2.2 void RectangularGnuplotScriptGenerator::generateFooter ( std::ofstream & fout ) [static, private]**

Generate the footer of the script.

#### Parameters

|             |                    |
|-------------|--------------------|
| <i>fout</i> | Output file stream |
|-------------|--------------------|

Definition at line 57 of file RectangularGnuplotScriptGenerator.cpp.

References FOOTER\_IN, and outputFooter().

Referenced by generateScript().

**6.156.2.3 void RectangularGnuplotScriptGenerator::generateHeader ( std::ofstream & fout, const std::string & outputPath, double simulationTime, unsigned long height, unsigned long width ) [static, private]**

Generate the header of the script.

**Parameters**

|                        |                         |
|------------------------|-------------------------|
| <i>fout</i>            | Output file stream      |
| <i>output-Filepath</i> | Path to the output file |
| <i>simulation-Time</i> | Simulation time         |
| <i>height</i>          | Height of the grid      |
| <i>width</i>           | Width of the grid       |

Definition at line 31 of file RectangularGnuplotScriptGenerator.cpp.

References multiscale::StringManipulator::filenameFromPath(), HEADER\_IN, and outputHeader().

Referenced by generateScript().

```
6.156.2.4 void RectangularGnuplotScriptGenerator::generateScript ( const
                     std::vector< double > & concentrations, double simulationTime, unsigned long
                     height, unsigned long width, const std::string & outputfilepath ) [static]
```

Generate the script.

**Parameters**

|                        |                         |
|------------------------|-------------------------|
| <i>concentra-tions</i> | Concentrations          |
| <i>simulation-Time</i> | Simulation time         |
| <i>height</i>          | Height of the grid      |
| <i>width</i>           | Width of the grid       |
| <i>output-Filepath</i> | Path of the output file |

Definition at line 15 of file RectangularGnuplotScriptGenerator.cpp.

References generateBody(), generateFooter(), generateHeader(), and GNUPLOT\_EXTENSION.

Referenced by multiscale::visualisation::CartesianToConcentrationsConverter::outputResults().

```
6.156.2.5 void RectangularGnuplotScriptGenerator::outputContent ( const
                     std::vector< double > & concentrations, unsigned long height, unsigned long width,
                     std::ofstream & fout ) [static, private]
```

Output the content of the script.

**Parameters**

|                                   |                        |
|-----------------------------------|------------------------|
| <i>concentra-</i><br><i>tions</i> | The concentrations     |
| <i>height</i>                     | The height of the grid |
| <i>width</i>                      | The width of the grid  |
| <i>fout</i>                       | Output file stream     |

Definition at line 102 of file RectangularGnuplotScriptGenerator.cpp.

References OUTPUT\_PRECISION, and OUTPUT\_SEPARATOR.

Referenced by generateBody().

**6.156.2.6 void RectangularGnuplotScriptGenerator::outputFooter ( std::ifstream & *fin*, std::ofstream & *fout* ) [static, private]**

Output the footer of the script.

**Parameters**

|             |                    |
|-------------|--------------------|
| <i>fin</i>  | Input file stream  |
| <i>fout</i> | Output file stream |

Definition at line 118 of file RectangularGnuplotScriptGenerator.cpp.

Referenced by generateFooter().

**6.156.2.7 void RectangularGnuplotScriptGenerator::outputHeader ( std::ifstream & *fin*, const std::string & *outputFilename*, double *simulationTime*, unsigned long *height*, unsigned long *width*, std::ofstream & *fout* ) [static, private]**

Output the header of the script.

**Parameters**

|                                               |                         |
|-----------------------------------------------|-------------------------|
| <i>fin</i>                                    | Input file stream       |
| <i>output-</i><br><i>Filename</i>             | Name of the output file |
| <i>simula-</i><br><i>tion-</i><br><i>Time</i> | Simulation time         |
| <i>height</i>                                 | The height of the grid  |
| <i>width</i>                                  | The width of the grid   |
| <i>fout</i>                                   | Output file stream      |

Definition at line 67 of file RectangularGnuplotScriptGenerator.cpp.

References multiscale::StringManipulator::replace(), REPLACE\_DIMENSION\_EXTRA, REPLACE\_HEADER\_FILENAME, REPLACE\_HEADER\_HEIGHT, REPLACE\_HEADER\_SIM\_TIME, and REPLACE\_HEADER\_WIDTH.

Referenced by generateHeader().

### 6.156.3 Member Data Documentation

**6.156.3.1 const std::string RectangularGnuplotScriptGenerator::CONTENT\_IN = "/usr/local/share/mule/config/visualisation/rectangular/content.in" [static, private]**

Definition at line 102 of file RectangularGnuplotScriptGenerator.hpp.

Referenced by generateBody().

**6.156.3.2 const std::string RectangularGnuplotScriptGenerator::FOOTER\_IN = "/usr/local/share/mule/config/visualisation/rectangular/footer.in" [static, private]**

Definition at line 103 of file RectangularGnuplotScriptGenerator.hpp.

Referenced by generateFooter().

**6.156.3.3 const std::string RectangularGnuplotScriptGenerator-::GNUPLOT\_EXTENSION = ".plt" [static, private]**

Definition at line 116 of file RectangularGnuplotScriptGenerator.hpp.

Referenced by generateScript().

**6.156.3.4 const std::string RectangularGnuplotScriptGenerator::HEADER\_IN = "/usr/local/share/mule/config/visualisation/rectangular/header.in" [static, private]**

Definition at line 101 of file RectangularGnuplotScriptGenerator.hpp.

Referenced by generateHeader().

**6.156.3.5 const int RectangularGnuplotScriptGenerator::OUTPUT\_PRECISION = std::numeric\_limits<double>::max\_digits10 [static, private]**

Definition at line 112 of file RectangularGnuplotScriptGenerator.hpp.

Referenced by outputContent().

**6.156.3.6 const std::string RectangularGnuplotScriptGenerator::OUTPUT\_SEPARATOR = " " [static, private]**

Definition at line 114 of file RectangularGnuplotScriptGenerator.hpp.

Referenced by outputContent().

```
6.156.3.7 const double RectangularGnuplotScriptGenerator::-
REPLACE_DIMENSION_EXTRA = 0.5 [static,
private]
```

Definition at line 110 of file RectangularGnuplotScriptGenerator.hpp.

Referenced by outputHeader().

```
6.156.3.8 const std::string RectangularGnuplotScriptGenerator::REPLA-
CE_HEADER_FILENAME = "OUTPUT_FILENAME" [static,
private]
```

Definition at line 105 of file RectangularGnuplotScriptGenerator.hpp.

Referenced by outputHeader().

```
6.156.3.9 const std::string RectangularGnuplotScriptGenerator::REPLA-
CE_HEADER_HEIGHT = "OUTPUT_DIMENSION1" [static,
private]
```

Definition at line 106 of file RectangularGnuplotScriptGenerator.hpp.

Referenced by outputHeader().

```
6.156.3.10 const std::string RectangularGnuplotScriptGenerator::REPLA-
CE_HEADER_SIM_TIME = "OUTPUT_SIM_TIME" [static,
private]
```

Definition at line 108 of file RectangularGnuplotScriptGenerator.hpp.

Referenced by outputHeader().

```
6.156.3.11 const std::string RectangularGnuplotScriptGenerator::REP-
LACE_HEADER_WIDTH = "OUTPUT_DIMENSION2" [static,
private]
```

Definition at line 107 of file RectangularGnuplotScriptGenerator.hpp.

Referenced by outputHeader().

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

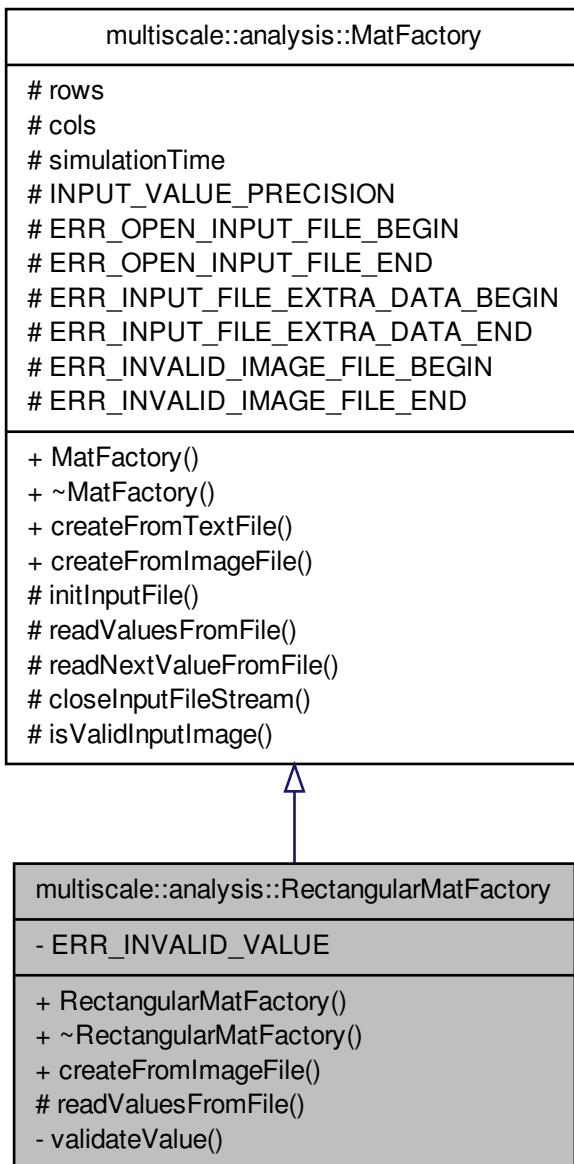
- RectangularGnuplotScriptGenerator.hpp
- RectangularGnuplotScriptGenerator.cpp

## **6.157 multiscale::analysis::RectangularMatFactory Class Reference**

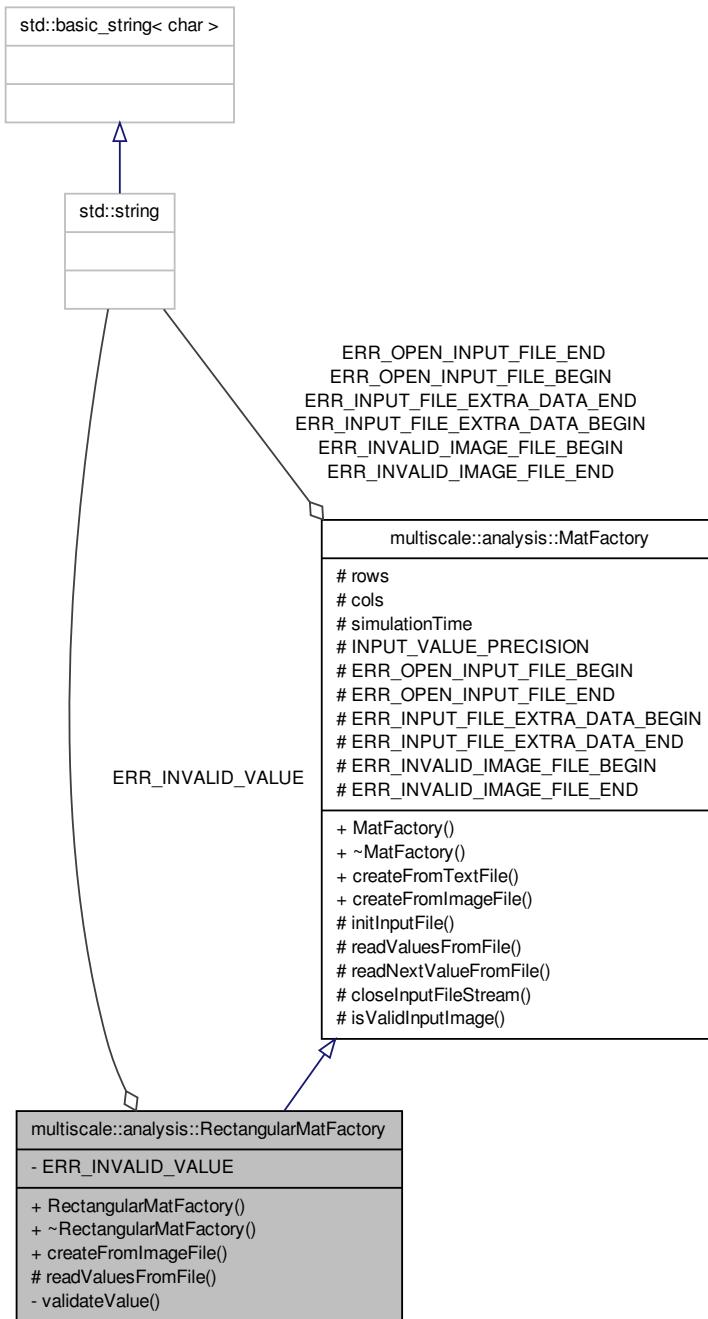
Class for creating a cv::Mat object considering a rectangular grid.

```
#include <RectangularMatFactory.hpp>
```

Inheritance diagram for multiscale::analysis::RectangularMatFactory:



Collaboration diagram for multiscale::analysis::RectangularMatFactory:



## Public Member Functions

- [RectangularMatFactory \(\)](#)
- [~RectangularMatFactory \(\)](#)
- [cv::Mat createFromImageFile \(const std::string &inputFilePath\) override](#)  
*Create a Mat object from the provided image file.*

## Protected Member Functions

- [void readValuesFromFile \(std::ifstream &fin, cv::Mat &image\) override](#)  
*Read the values from the given input file.*

## Private Member Functions

- [void validateValue \(float value\)](#)  
*Validate the provided image value.*

## Static Private Attributes

- static const std::string [ERR\\_INVALID\\_VALUE](#) = "All values have to be between 0 and 1."

### 6.157.1 Detailed Description

Class for creating a cv::Mat object considering a rectangular grid.

Definition at line 12 of file RectangularMatFactory.hpp.

### 6.157.2 Constructor & Destructor Documentation

#### 6.157.2.1 [RectangularMatFactory::RectangularMatFactory \( \)](#)

Definition at line 12 of file RectangularMatFactory.cpp.

#### 6.157.2.2 [RectangularMatFactory::~RectangularMatFactory \( \)](#)

Definition at line 14 of file RectangularMatFactory.cpp.

### 6.157.3 Member Function Documentation

#### 6.157.3.1 [cv::Mat RectangularMatFactory::createFromImageFile \( const std::string &inputFilePath \) \[override, virtual\]](#)

Create a Mat object from the provided image file.

Create a Mat instance from the given image file

**Parameters**

|                       |                            |
|-----------------------|----------------------------|
| <i>inputFile-Path</i> | The path to the image file |
|-----------------------|----------------------------|

Implements [multiscale::analysis::MatFactory](#).

Definition at line 16 of file RectangularMatFactory.cpp.

References multiscale::analysis::MatFactory::isValidInputImage().

**6.157.3.2 void RectangularMatFactory::readValuesFromFile ( std::ifstream & *fin*, cv::Mat & *image* ) [override, protected, virtual]**

Read the values from the given input file.

Read the values from the input file and return them as an array which can be used afterwards to create a cv::Mat object. Each floating point value is in the interval [0, 255].

REMARK: The constructor of cv::Mat does not copy the data. Therefore, DO NOT deallocate the data in this class.

**Parameters**

|              |                                                  |
|--------------|--------------------------------------------------|
| <i>fin</i>   | Input file stream from which the values are read |
| <i>image</i> | Image to which the values are written            |

Implements [multiscale::analysis::MatFactory](#).

Definition at line 33 of file RectangularMatFactory.cpp.

References multiscale::analysis::MatFactory::cols, multiscale::analysis::MatFactory::readNextValueFromFile(), multiscale::analysis::MatFactory::rows, and validateValue().

**6.157.3.3 void RectangularMatFactory::validateValue ( float *value* ) [private]**

Validate the provided image value.

The image value is valid if it is between 0 and 1. An exception is thrown if the image value is invalid.

**Parameters**

|              |                          |
|--------------|--------------------------|
| <i>value</i> | The provided image value |
|--------------|--------------------------|

Definition at line 48 of file RectangularMatFactory.cpp.

References ERR\_INVALID\_VALUE.

Referenced by [readValuesFromFile\(\)](#).

#### 6.157.4 Member Data Documentation

6.157.4.1 `const std::string RectangularMatFactory::ERR_INVALID_VALUE = "All values have to be between 0 and 1." [static, private]`

Definition at line 54 of file RectangularMatFactory.hpp.

Referenced by validateValue().

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

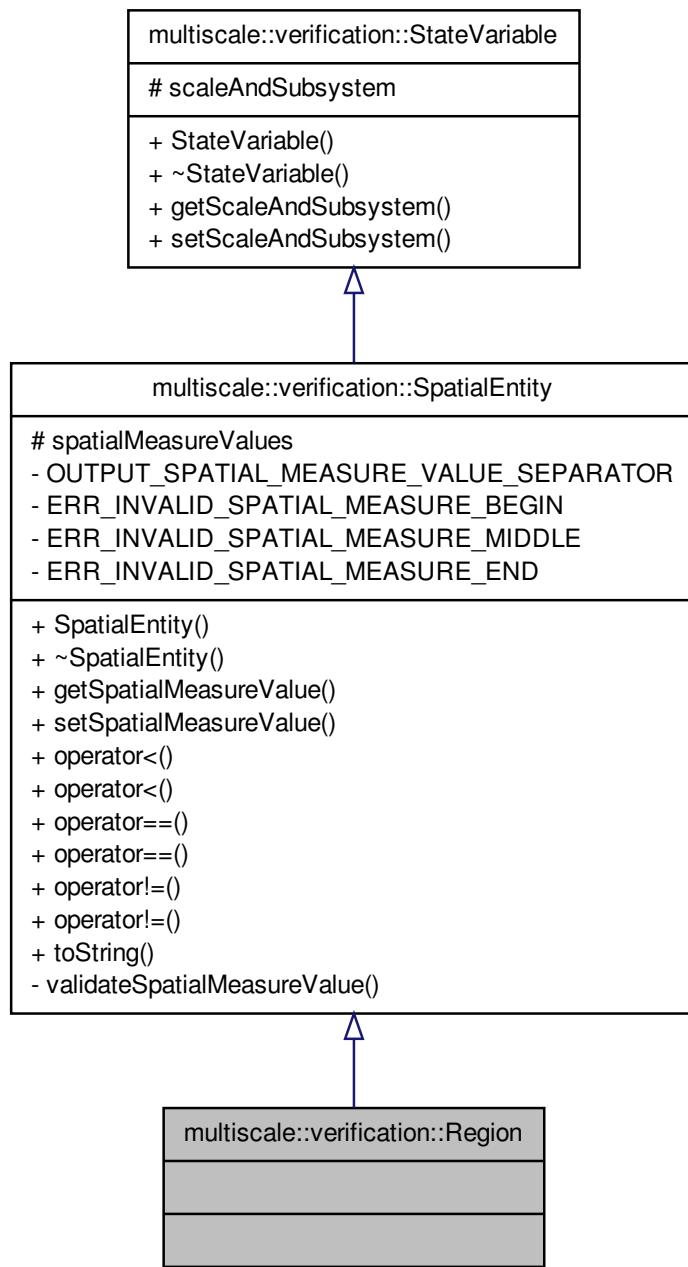
- RectangularMatFactory.hpp
  
- RectangularMatFactory.cpp

#### 6.158 multiscale::verification::Region Class Reference

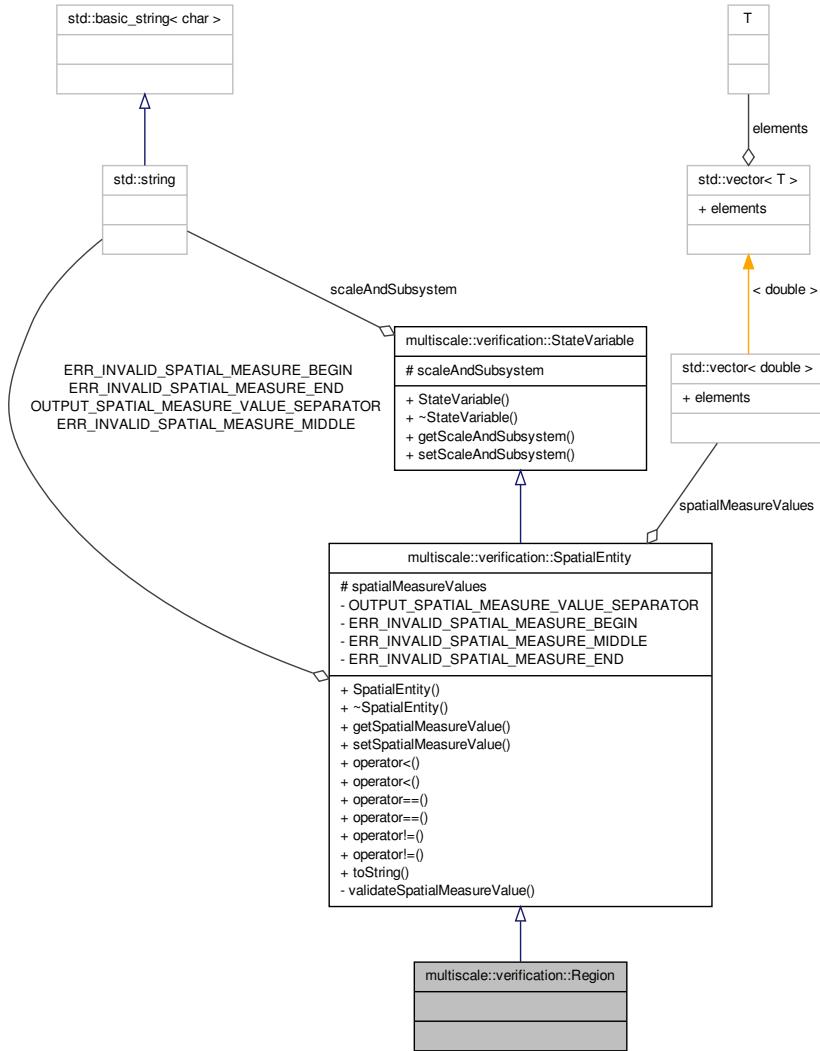
Class for representing a region.

```
#include <Region.hpp>
```

Inheritance diagram for multiscale::verification::Region:



Collaboration diagram for multiscale::verification::Region:



### 6.158.1 Detailed Description

Class for representing a region.

Definition at line 21 of file verification/spatial-temporal/include/multiscale/verification/spatial-temporal/model/Region.hpp.

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

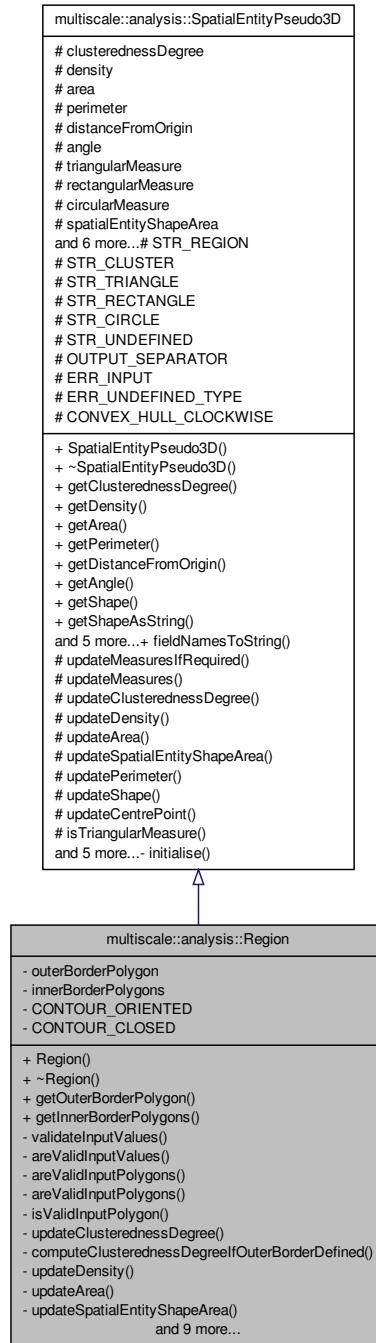
- verification/spatial-temporal/include/multiscale/verification/spatial-temporal/model/-Region.hpp

## 6.159 multiscale::analysis::Region Class Reference

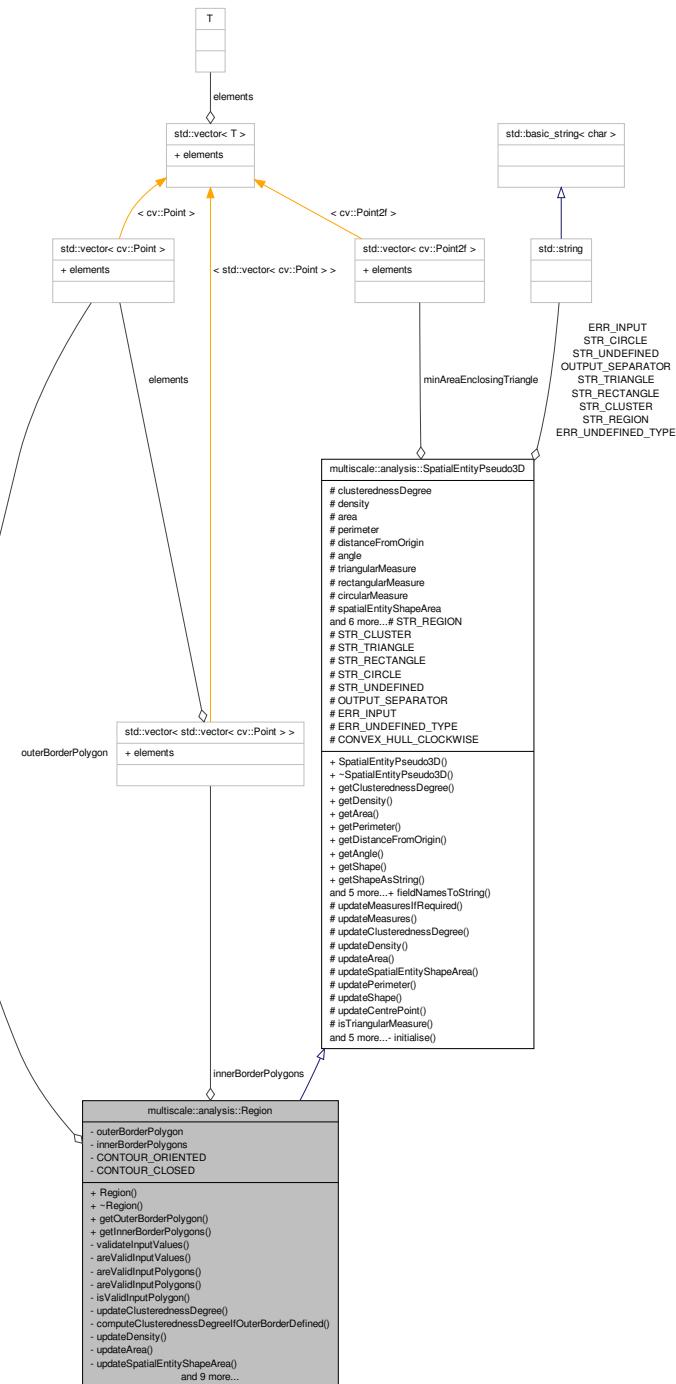
Class for representing a region.

```
#include <Region.hpp>
```

Inheritance diagram for multiscale::analysis::Region:



Collaboration diagram for multiscale::analysis::Region:



## Public Member Functions

- `Region` (double `density`, double `distanceFromOrigin`, double `angleWrtOrigin`, const std::vector< cv::Point > &`outerBorderPolygon`, const std::vector< cv::Point > > &`innerBorderPolygons`)  
*Get the polygon defining the outer border of the region.*
- `~Region` ()
- const std::vector< cv::Point > & `getOuterBorderPolygon` () const  
*Get the polygons defining the inner borders of the region.*

## Private Member Functions

- void `validateInputValues` (double `density`, double `distanceFromOrigin`, double `angleWrtOrigin`, const std::vector< cv::Point > &`outerBorderPolygon`, const std::vector< std::vector< cv::Point > > &`innerBorderPolygons`)  
*Validate the input values.*
- bool `isValidInputValues` (double `density`, double `distanceFromOrigin`, double `angleWrtOrigin`, const std::vector< cv::Point > &`outerBorderPolygon`, const std::vector< std::vector< cv::Point > > &`innerBorderPolygons`)  
*Check if the input values are valid or not.*
- bool `isValidInputPolygons` (const std::vector< cv::Point > &`outerBorderPolygon`, const std::vector< std::vector< cv::Point > > &`innerBorderPolygons`)  
*Check if the given input outer/inner border polygons are valid.*
- bool `isValidInputPolygons` (const std::vector< std::vector< cv::Point > > &`polygons`)  
*Check if the given input polygons are valid.*
- bool `isValidInputPolygon` (const std::vector< cv::Point > &`polygon`)  
*Check if the given input polygons are valid.*
- void `updateClusterednessDegree` () override  
*Update the value of the clusteredness degree.*
- double `computeClusterednessDegreeIfOuterBorderDefined` ()  
*Compute the value of the clusteredness degree if the outer border of the region is defined.*
- void `updateDensity` () override  
*Update the value of the density.*
- void `updateArea` () override  
*Update the area.*
- void `updateSpatialEntityShapeArea` () override  
*Update the spatial entity shape area.*
- double `computeAreaIfOuterBoderDefined` ()  
*Compute the value of the area if the outer border of the region is defined.*
- double `computeSpatialEntityShapeAreaIfOuterBoderDefined` ()

*Compute the value of the spatial entity shape area if the outer border of the region is defined.*

- void `updatePerimeter ()` override  
*Update the perimeter.*
- double `isTriangularMeasure ()` override  
*Get the measure that the cluster has a triangular shape.*
- double `isRectangularMeasure ()` override  
*Get the measure that the cluster has a rectangular shape.*
- double `isCircularMeasure ()` override  
*Get the measure that the cluster has a circular shape.*
- void `updateCentrePoint ()` override  
*Update the centre of the region.*
- void `updateCentrePointWhenRegionDefinedBySinglePoint ()`  
*Update the centre of the region in case the region is defined by a single point.*
- void `updateCentrePointWhenRegionDefinedByMultiplePoints ()`  
*Update the centre of the region in case the region is defined by multiple points.*
- `SpatialEntityPseudo3DType type ()` override  
*Return the type of the pseudo 3D spatial entity.*

## Private Attributes

- `std::vector< cv::Point > outerBorderPolygon`
- `std::vector< std::vector < cv::Point > > innerBorderPolygons`

## Static Private Attributes

- static const bool `CONTOUR_ORIENTED` = false
- static const bool `CONTOUR_CLOSED` = true

### 6.159.1 Detailed Description

Class for representing a region.

Definition at line 16 of file analysis/spatial/include/multiscale/analysis/spatial/model/-Region.hpp.

### 6.159.2 Constructor & Destructor Documentation

6.159.2.1 `Region::Region ( double density, double distanceFromOrigin, double angleWrtOrigin, const std::vector < cv::Point > & outerBorderPolygon, const std::vector < std::vector < cv::Point > > & innerBorderPolygons )`

Definition at line 11 of file Region.cpp.

References multiscale::analysis::SpatialEntityPseudo3D::angle, multiscale::analysis::SpatialEntityPseudo3D::density, multiscale::analysis::SpatialEntityPseudo3D::distanceFromOrigin, innerBorderPolygons, outerBorderPolygon, and validateInputValues().

Referenced by type().

#### 6.159.2.2 Region::~Region( )

Definition at line 25 of file Region.cpp.

### 6.159.3 Member Function Documentation

**6.159.3.1 bool Region::isValidInputPolygons ( const std::vector< cv::Point > & outerBorderPolygon, const std::vector< std::vector< cv::Point > > & innerBorderPolygons ) [private]**

Check if the given input outer/inner border polygons are valid.

For each polygon p and each point a:  $0 \leq p.a.x \leq p.a.y$

#### Parameters

|                             |                                                      |
|-----------------------------|------------------------------------------------------|
| <i>outerBorder-Polygon</i>  | The polygon defining the outer border of the region  |
| <i>innerBorder-Polygons</i> | The polygon defining the inner borders of the region |

Definition at line 61 of file Region.cpp.

References innerBorderPolygons, and isValidInputPolygon().

Referenced by areValidInputValues().

**6.159.3.2 bool Region::isValidInputPolygons ( const std::vector< std::vector< cv::Point > > & polygons ) [private]**

Check if the given input polygons are valid.

For each polygon p and each point a:  $0 \leq p.a.x \leq p.a.y$

#### Parameters

|                 |                                  |
|-----------------|----------------------------------|
| <i>polygons</i> | The given collection of polygons |
|-----------------|----------------------------------|

Definition at line 69 of file Region.cpp.

References isValidInputPolygon().

6.159.3.3 `bool Region::isValidInputValues ( double density, double distanceFromOrigin, double angleWrtOrigin, const std::vector< cv::Point > & outerBorderPolygon, const std::vector< std::vector< cv::Point > > & innerBorderPolygons ) [private]`

Check if the input values are valid or not.

Validation rules:  $0 < \text{density}$   $0 < \text{distanceFromOrigin}$   $0 \leq \text{angleWrtOrigin} \leq 360$

For each polygon point p:  $0 \leq p.x \leq p.y$

#### Parameters

|                             |                                                      |
|-----------------------------|------------------------------------------------------|
| <i>density</i>              | The density of the region                            |
| <i>distance-FromOrigin</i>  | The distance from the origin                         |
| <i>angleWrt-Origin</i>      | The angle computed wrt to the origin                 |
| <i>outerBorder-Polygon</i>  | The polygon defining the outer border of the region  |
| <i>innerBorder-Polygons</i> | The polygon defining the inner borders of the region |

Definition at line 44 of file Region.cpp.

References multiscale::analysis::SpatialEntityPseudo3D::area, isValidInputPolygons(), multiscale::analysis::SpatialEntityPseudo3D::clusterednessDegree, multiscale::-Numeric::greaterOrEqual(), innerBorderPolygons, and multiscale::Numeric::lessOrEqual().

Referenced by validateInputValues().

6.159.3.4 `double Region::computeAreaIfOuterBoderDefined ( ) [private]`

Compute the value of the area if the outer border of the region is defined.

Definition at line 137 of file Region.cpp.

References multiscale::analysis::SpatialMeasureCalculator::computePolygonArea(), multiscale::analysis::SpatialMeasureCalculator::computePolygonHoleArea(), innerBorderPolygons, and outerBorderPolygon.

Referenced by updateArea().

6.159.3.5 `double Region::computeClusterednessDegreeIfOuterBorderDefined ( ) [private]`

Compute the value of the clusteredness degree if the outer border of the region is defined.

Definition at line 95 of file Region.cpp.

References multiscale::analysis::SpatialMeasureCalculator::computePolygonArea(),

`multiscale::analysis::SpatialMeasureCalculator::computePolygonHoleArea()`, `multiscale::Numeric::division()`, `innerBorderPolygons`, and `outerBorderPolygon`.

Referenced by `updateClusterednessDegree()`.

#### 6.159.3.6 `double Region::computeSpatialEntityShapeAreaIfOuterBorderDefined( ) [private]`

Compute the value of the spatial entity shape area if the outer border of the region is defined.

Definition at line 158 of file `Region.cpp`.

References `CONTOUR_ORIENTED`, `innerBorderPolygons`, and `outerBorderPolygon`.

Referenced by `updateSpatialEntityShapeArea()`.

#### 6.159.3.7 `const std::vector< std::vector< cv::Point > > & Region::getInnerBorderPolygons( ) const`

Get the polygons defining the inner borders of the region.

Definition at line 31 of file `Region.cpp`.

References `innerBorderPolygons`.

Referenced by `multiscale::analysis::RegionDetector::outputRegionToImage()`.

#### 6.159.3.8 `const std::vector< cv::Point > & Region::getOuterBorderPolygon( ) const`

Get the polygon defining the outer border of the region.

Definition at line 27 of file `Region.cpp`.

References `outerBorderPolygon`.

Referenced by `multiscale::analysis::RegionDetector::outputRegionToImage()`.

#### 6.159.3.9 `double Region::isCircularMeasure( ) [override, private, virtual]`

Get the measure that the cluster has a circular shape.

Implements [multiscale::analysis::SpatialEntityPseudo3D](#).

Definition at line 204 of file `Region.cpp`.

References `multiscale::analysis::SpatialEntityPseudo3D::minAreaEnclosingCircleCentre`, `multiscale::analysis::SpatialEntityPseudo3D::minAreaEnclosingCircleRadius`, `multiscale::analysis::SpatialEntityPseudo3D::normalisedShapeMeasure()`, `outerBorderPolygon`, and `multiscale::Geometry2D::PI`.

6.159.3.10 `double Region::isRectangularMeasure( ) [override, private, virtual]`

Get the measure that the cluster has a rectangular shape.

Implements [multiscale::analysis::SpatialEntityPseudo3D](#).

Definition at line 195 of file Region.cpp.

References [multiscale::analysis::SpatialEntityPseudo3D::minAreaEnclosingRect](#), [multiscale::analysis::SpatialEntityPseudo3D::normalisedShapeMeasure\(\)](#), and [outerBorderPolygon](#).

6.159.3.11 `double Region::isTriangularMeasure( ) [override, private, virtual]`

Get the measure that the cluster has a triangular shape.

Implements [multiscale::analysis::SpatialEntityPseudo3D](#).

Definition at line 179 of file Region.cpp.

References [multiscale::Geometry2D::computeConvexHull\(\)](#), [multiscale::analysis::SpatialEntityPseudo3D::CONVEX\\_HULL\\_CLOCKWISE](#), [multiscale::MinEnclosingTriangleFinder::find\(\)](#), [multiscale::analysis::SpatialEntityPseudo3D::minAreaEnclosingTriangle](#), [multiscale::analysis::SpatialEntityPseudo3D::normalisedShapeMeasure\(\)](#), and [outerBorderPolygon](#).

6.159.3.12 `bool Region::isValidInputPolygon( const std::vector< cv::Point > & polygon ) [private]`

Check if the given input polygons are valid.

For each polygon point p:  $0 \leq p.x \leq p.y$

**Parameters**

|                      |                   |
|----------------------|-------------------|
| <code>polygon</code> | The given polygon |
|----------------------|-------------------|

Definition at line 79 of file Region.cpp.

Referenced by [areValidInputPolygons\(\)](#).

6.159.3.13 `SpatialEntityPseudo3DType Region::type( ) [override, private, virtual]`

Return the type of the pseudo 3D spatial entity.

Implements [multiscale::analysis::SpatialEntityPseudo3D](#).

Definition at line 229 of file Region.cpp.

References [Region\(\)](#).

6.159.3.14 void Region::updateArea( ) [override, private, virtual]

Update the area.

Implements [multiscale::analysis::SpatialEntityPseudo3D](#).

Definition at line 125 of file Region.cpp.

References multiscale::analysis::SpatialEntityPseudo3D::area, computeAreaIfOuterBorderDefined(), and outerBorderPolygon.

6.159.3.15 void Region::updateCentrePoint( ) [override, private, virtual]

Update the centre of the region.

Implements [multiscale::analysis::SpatialEntityPseudo3D](#).

Definition at line 213 of file Region.cpp.

References outerBorderPolygon, updateCentrePointWhenRegionDefinedByMultiplePoints(), and updateCentrePointWhenRegionDefinedBySinglePoint().

6.159.3.16 void Region::updateCentrePointWhenRegionDefinedByMultiplePoints( ) [private]

Update the centre of the region in case the region is defined by multiple points.

Definition at line 225 of file Region.cpp.

References multiscale::analysis::SpatialEntityPseudo3D::centre, and outerBorderPolygon.

Referenced by updateCentrePoint().

6.159.3.17 void Region::updateCentrePointWhenRegionDefinedBySinglePoint( ) [private]

Update the centre of the region in case the region is defined by a single point.

Definition at line 221 of file Region.cpp.

References multiscale::analysis::SpatialEntityPseudo3D::centre, and outerBorderPolygon.

Referenced by updateCentrePoint().

6.159.3.18 void Region::updateClusterednessDegree( ) [override, private, virtual]

Update the value of the clusteredness degree.

Implements [multiscale::analysis::SpatialEntityPseudo3D](#).

Definition at line 89 of file Region.cpp.

References multiscale::analysis::SpatialEntityPseudo3D::clusterednessDegree, computeClusterednessDegreeIfOuterBorderDefined(), and outerBorderPolygon.

```
6.159.3.19 void Region::updateDensity( ) [override, private,
   virtual]
```

Update the value of the density.

Implements [multiscale::analysis::SpatialEntityPseudo3D](#).

Definition at line 123 of file Region.cpp.

```
6.159.3.20 void Region::updatePerimeter( ) [override, private,
   virtual]
```

Update the perimeter.

Implements [multiscale::analysis::SpatialEntityPseudo3D](#).

Definition at line 169 of file Region.cpp.

References multiscale::analysis::SpatialMeasureCalculator::computePolygonPerimeter(), outerBorderPolygon, and multiscale::analysis::SpatialEntityPseudo3D::perimeter.

```
6.159.3.21 void Region::updateSpatialEntityShapeArea( ) [override,
   private, virtual]
```

Update the spatial entity shape area.

Implements [multiscale::analysis::SpatialEntityPseudo3D](#).

Definition at line 131 of file Region.cpp.

References computeSpatialEntityShapeAreaIfOuterBoderDefined(), outerBorderPolygon, and multiscale::analysis::SpatialEntityPseudo3D::spatialEntityShapeArea.

```
6.159.3.22 void Region::validateInputValues( double density, double distanceFromOrigin,
  double angleWrtOrigin, const std::vector< cv::Point > & outerBorderPolygon,
  const std::vector< std::vector< cv::Point > > & innerBorderPolygons )
[private]
```

Validate the input values.

Validation rules:  $0 < \text{density}$   $0 < \text{distanceFromOrigin}$   $0 \leq \text{angleWrtOrigin} \leq 360$

For each polygon point p:  $0 \leq p.x \leq p.y$

**Parameters**

|                             |                                                      |
|-----------------------------|------------------------------------------------------|
| <i>density</i>              | The density of the region                            |
| <i>distance-FromOrigin</i>  | The distance from the origin                         |
| <i>angleWrt-Origin</i>      | The angle computed wrt to the origin                 |
| <i>outerBorder-Polygon</i>  | The polygon defining the outer border of the region  |
| <i>innerBorder-Polygons</i> | The polygon defining the inner borders of the region |

Definition at line 35 of file Region.cpp.

References areValidInputValues(), multiscale::analysis::SpatialEntityPseudo3D::ERR\_INPUT, and innerBorderPolygons.

Referenced by Region().

#### 6.159.4 Member Data Documentation

##### 6.159.4.1 const bool Region::CONTOUR\_CLOSED = true [static, private]

Definition at line 163 of file analysis/spatial/include/multiscale/analysis/spatial/model/-Region.hpp.

##### 6.159.4.2 const bool Region::CONTOUR\_ORIENTED = false [static, private]

Definition at line 162 of file analysis/spatial/include/multiscale/analysis/spatial/model/-Region.hpp.

Referenced by computeSpatialEntityShapeArealfOuterBoderDefined().

##### 6.159.4.3 std::vector<std::vector<cv::Point>> multiscale::analysis::Region::innerBorderPolygons [private]

Polygon defining the inner borders of the region

Definition at line 23 of file analysis/spatial/include/multiscale/analysis/spatial/model/-Region.hpp.

Referenced by areValidInputPolygons(), areValidInputValues(), computeArealfOuterBoderDefined(), computeClusterednessDegreelfOuterBorderDefined(), computeSpatialEntityShapeArealfOuterBoderDefined(), getInnerBorderPolygons(), Region(), and validateInputValues().

6.159.4.4 `std::vector<cv::Point> multiscale::analysis::Region::outerBorderPolygon`  
[private]

Polygon defining the outer border of the region

Definition at line 21 of file analysis/spatial/include/multiscale/analysis/spatial/model/-Region.hpp.

Referenced by `computeArealfOuterBoderDefined()`, `computeClusterednessDegreeIfOuterBorderDefined()`, `computeSpatialEntityShapeArealfOuterBoderDefined()`, `getOuterBorderPolygon()`, `isCircularMeasure()`, `isRectangularMeasure()`, `isTriangularMeasure()`, `Region()`, `updateArea()`, `updateCentrePoint()`, `updateCentrePointWhenRegionDefinedByMultiplePoints()`, `updateCentrePointWhenRegionDefinedBySinglePoint()`, `updateClusterednessDegree()`, `updatePerimeter()`, and `updateSpatialEntityShapeArea()`.

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

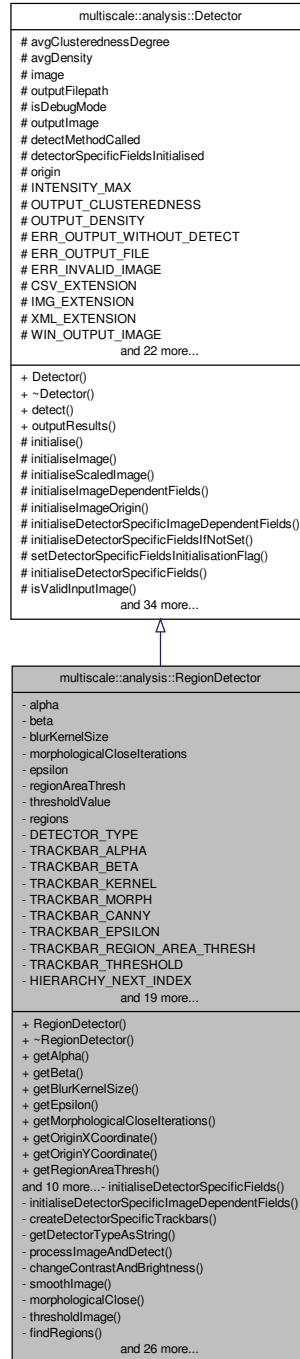
- analysis/spatial/include/multiscale/analysis/spatial/model/Region.hpp
  
- Region.cpp

## 6.160 multiscale::analysis::RegionDetector Class Reference

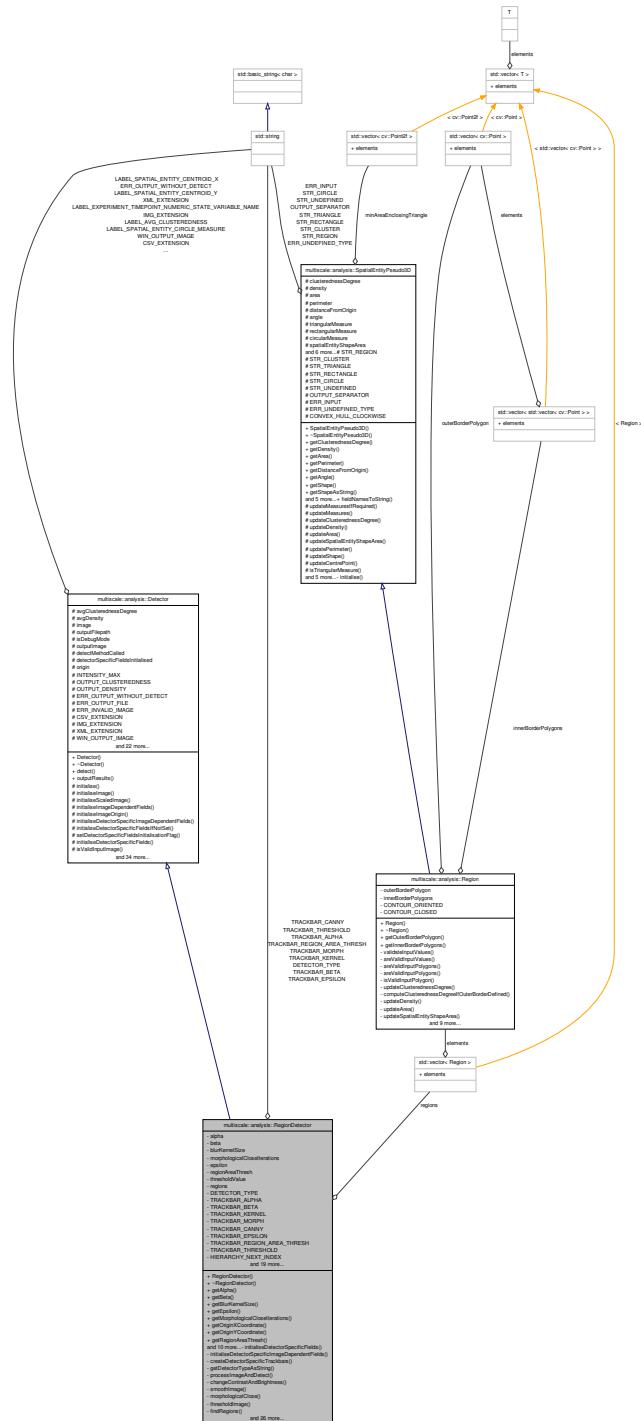
Class for detecting regions of high intensity in grayscale images.

```
#include <RegionDetector.hpp>
```

Inheritance diagram for multiscale::analysis::RegionDetector:



Collaboration diagram for multiscale::analysis::RegionDetector:



## Public Member Functions

- `RegionDetector (bool isDebugMode=false)`
- `~RegionDetector ()`
- `int getAlpha ()`

*Get the value of field alpha.*
- `int getBeta ()`

*Get the value of field beta.*
- `int getBlurKernelSize ()`

*Get the value of field blurKernelSize.*
- `int getEpsilon ()`

*Get the value of field epsilon.*
- `int getMorphologicalCloselterations ()`

*Get the value of field morphologicalCloselterations.*
- `int getOriginXCoordinate ()`

*Get the value of field originXCoordinate.*
- `int getOriginYCoordinate ()`

*Get the value of field originYCoordinate.*
- `int getRegionAreaThresh ()`

*Get the value of field regionAreaThresh.*
- `int getThresholdValue ()`

*Get the value of field thresholdValue.*
- `std::vector< Region > const & getRegions ()`

*Get a const reference to the vector of detected regions.*
- `void setAlpha (int alpha)`

*Set the value of field alpha.*
- `void setBeta (int beta)`

*Set the value of field beta.*
- `void setBlurKernelSize (int blurKernelSize)`

*Set the value of field blurKernelSize.*
- `void setEpsilon (int epsilon)`

*Set the value of field epsilon.*
- `void setMorphologicalCloselterations (int morphologicalCloselterations)`

*Set the value of field morphologicalCloselterations.*
- `void setOriginXCoordinate (int originXCoordinate)`

*Set the value of field originXCoordinate.*
- `void setOriginYCoordinate (int originYCoordinate)`

*Set the value of field originYCoordinate.*
- `void setRegionAreaThresh (int regionAreaThresh)`

*Set the value of field regionAreaThresh.*
- `void setThresholdValue (int thresholdValue)`

*Set the value of field thresholdValue.*

### Private Member Functions

- void `initialiseDetectorSpecificFields () override`  
*Initialise the vision members.*
- void `initialiseDetectorSpecificImageDependentFields () override`  
*Initialisation of the detector specific image dependent values.*
- void `createDetectorSpecificTrackbars () override`  
*Create the trackbars.*
- std::string `getDetectorTypeAsString () override`  
*Get the type of the detector as a string.*
- void `processImageAndDetect () override`  
*Process the given image.*
- void `changeContrastAndBrightness (cv::Mat &processedImage)`  
*Change the contrast and brightness of the image.*
- void `smoothImage (cv::Mat &image)`  
*Smooth out differences in the image.*
- void `morphologicalClose (cv::Mat &image)`  
*Apply the morphological close operator on the image.*
- void `thresholdImage (const cv::Mat &image, cv::Mat &thresholdedImage)`  
*Apply the threshold filter on the image.*
- void `findRegions (const cv::Mat &image, std::vector< Region > &regions)`  
*Find the regions in the image.*
- void `computeAverageMeasures (std::vector< Region > &regions)`  
*Compute the average clusteredness degree and average density.*
- void `computeAverageClusterednessDegree (std::vector< Region > &regions)`  
*Compute the average clusteredness degree.*
- double `computeSumOfAverageCentroidDistances (std::vector< Region > &regions)`  
*Compute the average distances sum between regions' centroids.*
- double `computeAverageCentroidDistance (const cv::Point2f &centroid, std::vector< Region > &regions)`  
*Compute the average distance from the given centroid to all other regions centroids.*
- void `computeAverageDensity (std::vector< Region > &regions)`  
*Compute the average density.*
- std::vector< Polygon > `findPolygonsInImage (const cv::Mat &image)`  
*Find polygons in image.*
- std::vector< Polygon > `createPolygons (const std::vector< std::vector< cv::Point > > &contours, const std::vector< cv::Vec4i > &hierarchy)`  
*Create polygons from the given contours and hierarchy information.*
- bool `existContours (const std::vector< std::vector< cv::Point > > &contours)`  
*Check if the number of contours is greater than 0.*
- void `createPolygonsFromContours (const std::vector< std::vector< cv::Point > > &contours, const std::vector< cv::Vec4i > &hierarchy, std::vector< Polygon > &polygons)`

- **Create polygons from the given contours and hierarchy information.**
- **Polygon createPolygon** (int contourIndex, const std::vector< std::vector< cv::Point > > &contours, const std::vector< cv::Vec4i > &hierarchy)
 

*Create a new polygon considering the given contour index, contours and hierarchy information.*
- void **setPolygonOuterContour** (int contourIndex, const std::vector< std::vector< cv::Point > > &contours, const std::vector< cv::Vec4i > &hierarchy, **Polygon** &polygon)
 

*Set the outer contour of the polygon.*
- void **setPolygonInnerContours** (int contourIndex, const std::vector< std::vector< cv::Point > > &contours, const std::vector< cv::Vec4i > &hierarchy, **Polygon** &polygon)
 

*Set the inner contours of the polygon.*
- void **approximatePolygonOuterBorder** (**Polygon** &polygon)
 

*Approximate the outer contour of the given polygon.*
- **Region createRegionFromPolygon** (const **Polygon** &polygon)
 

*Create a new region from the given polygon.*
- bool **isValidContour** (const std::vector< cv::Point > &contour)
 

*Check if the contour is valid.*
- bool **isValidHole** (const std::vector< cv::Point > &hole, const std::vector< cv::Point > &outerPolygon)
 

*Check if the hole contained by the outer polygon is valid.*
- double **computeRegionDensity** (const **Polygon** &polygon)
 

*Compute the density of the area delimited by the given polygon.*
- cv::Mat **createMaskForPolygon** (const std::vector< cv::Point > &outerBorderPolygon, const std::vector< std::vector< cv::Point > > &innerBorderPolygons)
 

*Create an image mask considering the given outer and inner border polygons.*
- double **computeAverageIntensity** (const cv::Mat &image, const cv::Mat &mask)
 

*Compute the average intensity of the image considering only the positions specified by the mask.*
- void **clearPreviousDetectionResults** () override
 

*Clear the element present in the regions vector.*
- std::vector< std::shared\_ptr < **SpatialEntityPseudo3D** > > **getCollectionOfSpatialEntityPseudo3D** () override
 

*Get the collection of clusters detected in the image.*
- void **outputResultsToImage** () override
 

*Output the results to the outputImage instance.*
- void **outputRegionToImage** (const **Region** &region, cv::Mat &outputImage)
 

*Output the region to the outputImage instance.*
- void **outputRegionOuterBorderToImage** (const std::vector< cv::Point > &outerBorder, cv::Mat &outputImage)
 

*Output the outer border polygon of a region to the outputImage instance.*
- void **outputRegionInnerBordersToImage** (const std::vector< std::vector< cv::Point > > &innerBorders, cv::Mat &outputImage)
 

*Output the inner border polygons of a region to the outputImage instance.*

- double `convertAlpha` (int `alpha`)  
*Convert alpha from the range [0, ALPHA\_MAX] to [ALPHA\_REAL\_MIN, ALPHA\_REAL\_MAX].*
- int `convertBeta` (int `beta`)  
*Convert beta from the range [0, BETA\_MAX] to [BETA\_REAL\_MIN, BETA\_REAL\_MAX].*

### Private Attributes

- int `alpha`
- int `beta`
- int `blurKernelSize`
- int `morphologicalCloselterations`
- int `epsilon`
- int `regionAreaThresh`
- int `thresholdValue`
- std::vector< `Region` > `regions`

### Static Private Attributes

- static const std::string `DETECTOR_TYPE` = "Regions"
- static const std::string `TRACKBAR_ALPHA` = "Alpha"
- static const std::string `TRACKBAR_BETA` = "Beta"
- static const std::string `TRACKBAR_KERNEL` = "Gaussian blur kernel size"
- static const std::string `TRACKBAR_MORPH` = "Morphological open, number of iterations"
- static const std::string `TRACKBAR_CANNY` = "Canny lower threshold"
- static const std::string `TRACKBAR_EPSILON` = "Epsilon"
- static const std::string `TRACKBAR_REGION_AREA_THRESH` = "Region area threshold"
- static const std::string `TRACKBAR_THRESHOLD` = "Threshold value"
- static const int `HIERARCHY_NEXT_INDEX` = 0
- static const int `HIERARCHY_PREV_INDEX` = 1
- static const int `HIERARCHY_FIRST_CHILD_INDEX` = 2
- static const int `HIERARCHY_PARENT_INDEX` = 3
- static const bool `CONTOUR_AREA_ORIENTED` = false
- static const double `ALPHA_REAL_MIN` = 1.0
- static const double `ALPHA_REAL_MAX` = 3.0
- static const int `BETA_REAL_MIN` = -100
- static const int `BETA_REAL_MAX` = 100
- static const int `ALPHA_MAX` = 1000
- static const int `BETA_MAX` = 200
- static const int `KERNEL_MAX` = 2000
- static const int `MORPH_ITER_MAX` = 100
- static const int `CANNY_THRESH_MAX` = 100

- static const int **EPSILON\_MAX** = 100
- static const int **REGION\_AREA\_THRESH\_MAX** = 200000
- static const int **THRESHOLD\_MAX** = 255
- static const int **THRESHOLD\_CLUSTEREDNESS** = 0
- static const int **THRESHOLD\_HOLE\_AREA** = 0
- static const bool **POLYGON\_CLOSED** = true
- static const int **DISPLAY\_LINE\_THICKNESS** = 1

### 6.160.1 Detailed Description

Class for detecting regions of high intensity in grayscale images.

Definition at line 24 of file RegionDetector.hpp.

### 6.160.2 Constructor & Destructor Documentation

#### 6.160.2.1 RegionDetector::RegionDetector ( *bool isDebugMode = false* )

Definition at line 14 of file RegionDetector.cpp.

References alpha, multiscale::analysis::Detector::avgClusterednessDegree, multiscale::analysis::Detector::avgDensity, beta, blurKernelSize, epsilon, morphologicalCloseIterations, regionAreaThresh, and thresholdValue.

#### 6.160.2.2 RegionDetector::~RegionDetector ( )

Definition at line 27 of file RegionDetector.cpp.

### 6.160.3 Member Function Documentation

#### 6.160.3.1 void RegionDetector::approximatePolygonOuterBorder ( *Polygon & polygon* ) [private]

Approximate the outer contour of the given polygon.

##### Parameters

|                |                   |
|----------------|-------------------|
| <i>polygon</i> | The given polygon |
|----------------|-------------------|

Definition at line 338 of file RegionDetector.cpp.

References epsilon.

Referenced by findRegions().

6.160.3.2 void RegionDetector::changeContrastAndBrightness ( cv::Mat & *processedImage* ) [private]

Change the contrast and brightness of the image.

Change the contrast and brightness of the image by the factors alpha and gamma

Parameters

|                        |                     |
|------------------------|---------------------|
| <i>processed-Image</i> | The processed image |
|------------------------|---------------------|

Definition at line 163 of file RegionDetector.cpp.

References alpha, beta, convertAlpha(), convertBeta(), and multiscale::analysis::-Detector::image.

Referenced by processImageAndDetect().

6.160.3.3 void RegionDetector::clearPreviousDetectionResults ( ) [override, private, virtual]

Clear the element present in the regions vector.

Implements [multiscale::analysis::Detector](#).

Definition at line 437 of file RegionDetector.cpp.

References regions.

6.160.3.4 double RegionDetector::computeAverageCentroidDistance ( const cv::Point2f & *centroid*, std::vector< Region > & *regions* ) [private]

Compute the average distance from the given centroid to all other regions centroids.

Parameters

|                 |                                                                                   |
|-----------------|-----------------------------------------------------------------------------------|
| <i>centroid</i> | The centroid from which the distance to all other regions' centroids are computed |
| <i>regions</i>  | The collection of all regions                                                     |

Definition at line 233 of file RegionDetector.cpp.

References multiscale::Geometry2D::distanceBtwPoints().

Referenced by computeSumOfAverageCentroidDistances().

6.160.3.5 void RegionDetector::computeAverageClusterednessDegree ( std::vector< Region > & *regions* ) [private]

Compute the average clusteredness degree.

**Parameters**

|                |                          |
|----------------|--------------------------|
| <i>regions</i> | The regions in the image |
|----------------|--------------------------|

Definition at line 208 of file RegionDetector.cpp.

References multiscale::analysis::Detector::avgClusterednessDegree, computeSumOfAverageCentroidDistances(), and multiscale::Numeric::division().

Referenced by computeAverageMeasures().

**6.160.3.6 void RegionDetector::computeAverageDensity ( std::vector< Region > & *regions* ) [private]**

Compute the average density.

**Parameters**

|                |                          |
|----------------|--------------------------|
| <i>regions</i> | The regions in the image |
|----------------|--------------------------|

Definition at line 253 of file RegionDetector.cpp.

References multiscale::analysis::Detector::avgDensity, and multiscale::Numeric::division().

Referenced by computeAverageMeasures().

**6.160.3.7 double RegionDetector::computeAverageIntensity ( const cv::Mat & *image*, const cv::Mat & *mask* ) [private]**

Compute the average intensity of the image considering only the positions specified by the mask.

The type of the image is assumed to be CV\_32FC1, respectively of the mask CV\_8UC1.

**Parameters**

|              |                     |
|--------------|---------------------|
| <i>image</i> | The provided image  |
| <i>mask</i>  | The considered mask |

Definition at line 416 of file RegionDetector.cpp.

References multiscale::Numeric::average(), and multiscale::analysis::Detector::INTENSITY\_MAX.

Referenced by computeRegionDensity().

**6.160.3.8 void RegionDetector::computeAverageMeasures ( std::vector< Region > & *regions* ) [private]**

Compute the average clusteredness degree and average density.

**Parameters**

|                |                          |
|----------------|--------------------------|
| <i>regions</i> | The regions in the image |
|----------------|--------------------------|

Definition at line 203 of file RegionDetector.cpp.

References computeAverageClusterednessDegree(), and computeAverageDensity().

Referenced by processImageAndDetect().

**6.160.3.9 double RegionDetector::computeRegionDensity ( const Polygon & *polygon* ) [private]**

Compute the density of the area delimited by the given polygon.

The density is equal to the average intensity of the pixels in the area delimited by the given polygon divided by INTENSITY\_MAX.

**Parameters**

|                |                   |
|----------------|-------------------|
| <i>polygon</i> | The given polygon |
|----------------|-------------------|

Definition at line 373 of file RegionDetector.cpp.

References computeAverageIntensity(), createMaskForPolygon(), multiscale::Numeric::division(), multiscale::analysis::Detector::image, and multiscale::analysis::Detector::INTENSITY\_MAX.

Referenced by createRegionFromPolygon().

**6.160.3.10 double RegionDetector::computeSumOfAverageCentroidDistances ( std::vector< Region > & *regions* ) [private]**

Compute the average distances sum between regions' centroids.

**Parameters**

|                |                          |
|----------------|--------------------------|
| <i>regions</i> | The regions in the image |
|----------------|--------------------------|

Definition at line 220 of file RegionDetector.cpp.

References computeAverageCentroidDistance(), and regions.

Referenced by computeAverageClusterednessDegree().

**6.160.3.11 double RegionDetector::convertAlpha ( int *alpha* ) [private]**

Convert alpha from the range [0, ALPHA\_MAX] to [ALPHA\_REAL\_MIN, ALPHA\_REAL\_MAX].

**Parameters**

|              |       |
|--------------|-------|
| <i>alpha</i> | Alpha |
|--------------|-------|

Definition at line 483 of file RegionDetector.cpp.

References ALPHA\_MAX, ALPHA\_REAL\_MAX, and ALPHA\_REAL\_MIN.

Referenced by changeContrastAndBrightness().

#### 6.160.3.12 int RegionDetector::convertBeta ( int *beta* ) [private]

Convert beta from the range [0, BETA\_MAX] to [BETA\_REAL\_MIN, BETA\_REAL\_MAX].

**Parameters**

|             |      |
|-------------|------|
| <i>beta</i> | Beta |
|-------------|------|

Definition at line 491 of file RegionDetector.cpp.

References BETA\_MAX, BETA\_REAL\_MAX, and BETA\_REAL\_MIN.

Referenced by changeContrastAndBrightness().

#### 6.160.3.13 void RegionDetector::createDetectorSpecificTrackbars ( ) [override, private, virtual]

Create the trackbars.

Implements [multiscale::analysis::Detector](#).

Definition at line 135 of file RegionDetector.cpp.

References alpha, ALPHA\_MAX, beta, BETA\_MAX, blurKernelSize, epsilon, EPSILON\_MAX, KERNEL\_MAX, MORPH\_ITER\_MAX, morphologicalCloselterations, REGION\_AREA\_THRESH\_MAX, regionAreaThresh, THRESHOLD\_MAX, thresholdValue, TRACKBAR\_ALPHA, TRACKBAR\_BETA, TRACKBAR\_EPSILON, TRACKBAR\_KERNEL, TRACKBAR\_MORPH, TRACKBAR\_REGION\_AREA\_THRESH, TRACKBAR\_THRESHOLD, and multiscale::analysis::Detector::WIN\_OUTPUT\_IMAGE.

#### 6.160.3.14 cv::Mat RegionDetector::createMaskForPolygon ( const std::vector<cv::Point > & *outerBorderPolygon*, const std::vector<std::vector<cv::Point>> & *innerBorderPolygons* ) [private]

Create an image mask considering the given outer and inner border polygons.

The value of a mask pixel is maximum if the pixel is contained by the outer and not the inner border polygons. Otherwise the value of the mask pixel is minimum.

**Parameters**

|                             |                                         |
|-----------------------------|-----------------------------------------|
| <i>outerBorder-Polygon</i>  | The outer border polygon                |
| <i>innerBorder-Polygons</i> | The collection of inner border polygons |

Definition at line 388 of file RegionDetector.cpp.

References multiscale::analysis::Detector::image, and multiscale::analysis::Detector::INTENSITY\_MAX.

Referenced by computeRegionDensity().

```
6.160.3.15 Polygon RegionDetector::createPolygon ( int contourIndex, const
std::vector< std::vector< cv::Point > > & contours, const std::vector< cv::Vec4i
> & hierarchy ) [private]
```

Create a new polygon considering the given contour index, contours and hierarchy information.

**Parameters**

|                      |                                                          |
|----------------------|----------------------------------------------------------|
| <i>contour-Index</i> | The index of the outer contour                           |
| <i>contours</i>      | The collection of all contours                           |
| <i>hierarchy</i>     | The information regarding the hierarchy between contours |

Definition at line 311 of file RegionDetector.cpp.

References setPolygonInnerContours(), and setPolygonOuterContour().

Referenced by createPolygonsFromContours().

```
6.160.3.16 std::vector< Polygon > RegionDetector::createPolygons ( const
std::vector< std::vector< cv::Point > > & contours, const std::vector< cv::Vec4i
> & hierarchy ) [private]
```

Create polygons from the given contours and hierarchy information.

**Parameters**

|                  |                                                          |
|------------------|----------------------------------------------------------|
| <i>contours</i>  | The given contours                                       |
| <i>hierarchy</i> | The information regarding the hierarchy between contours |

Definition at line 284 of file RegionDetector.cpp.

References createPolygonsFromContours(), and existContours().

Referenced by findPolygonsInImage().

---

**6.160.3.17 void RegionDetector::createPolygonsFromContours ( const std::vector< std::vector< cv::Point > > & contours, const std::vector< cv::Vec4i > & hierarchy, std::vector< Polygon > & polygons ) [private]**

Create polygons from the given contours and hierarchy information.

**Parameters**

|                  |                                                            |
|------------------|------------------------------------------------------------|
| <i>contours</i>  | The given contours                                         |
| <i>hierarchy</i> | The information regarding the hierarchy between contours   |
| <i>polygons</i>  | The collection of polygons created from the given contours |

Definition at line 299 of file RegionDetector.cpp.

References createPolygon(), HIERARCHY\_NEXT\_INDEX, and isValidContour().

Referenced by createPolygons().

**6.160.3.18 Region RegionDetector::createRegionFromPolygon ( const Polygon & polygon ) [private]**

Create a new region from the given polygon.

Process the polygon in order to get the required information (e.g. clusteredness, area etc.) and create a region using this information

**Parameters**

|                |                                |
|----------------|--------------------------------|
| <i>polygon</i> | Polygon determining the region |
|----------------|--------------------------------|

Definition at line 344 of file RegionDetector.cpp.

References multiscale::analysis::Detector::computeDistanceFromOrigin(), multiscale::analysis::Detector::computePolygonAngle(), computeRegionDensity(), and multiscale::analysis::Region.

Referenced by findRegions().

**6.160.3.19 bool RegionDetector::existContours ( const std::vector< std::vector< cv::Point > > & contours ) [private]**

Check if the number of contours is greater than 0.

**Parameters**

|                 |                    |
|-----------------|--------------------|
| <i>contours</i> | The given contours |
|-----------------|--------------------|

Definition at line 295 of file RegionDetector.cpp.

Referenced by createPolygons().

6.160.3.20 `std::vector< Polygon > RegionDetector::findPolygonsInImage ( const cv::Mat & image ) [private]`

Find polygons in image.

Parameters

|                           |           |
|---------------------------|-----------|
| <code><i>image</i></code> | The image |
|---------------------------|-----------|

Definition at line 263 of file RegionDetector.cpp.

References `createPolygons()`, and `multiscale::analysis::Detector::offsetPolygons()`.

Referenced by `findRegions()`.

6.160.3.21 `void RegionDetector::findRegions ( const cv::Mat & image, std::vector< Region > & regions ) [private]`

Find the regions in the image.

Find the contours, approximate the polygons and extract the required information from them.

Parameters

|                             |                          |
|-----------------------------|--------------------------|
| <code><i>image</i></code>   | The image                |
| <code><i>regions</i></code> | The regions in the image |

Definition at line 191 of file RegionDetector.cpp.

References `approximatePolygonOuterBorder()`, `createRegionFromPolygon()`, and `findPolygonsInImage()`.

Referenced by `processImageAndDetect()`.

6.160.3.22 `int RegionDetector::getAlpha ( )`

Get the value of field alpha.

Definition at line 29 of file RegionDetector.cpp.

References alpha.

6.160.3.23 `int RegionDetector::getBeta ( )`

Get the value of field beta.

Definition at line 33 of file RegionDetector.cpp.

References beta.

**6.160.3.24 int RegionDetector::getBlurKernelSize ( )**

Get the value of field blurKernelSize.

Definition at line 37 of file RegionDetector.cpp.

References blurKernelSize.

**6.160.3.25 std::vector< std::shared\_ptr< SpatialEntityPseudo3D > >  
RegionDetector::getCollectionOfSpatialEntityPseudo3D ( )  
[override, private, virtual]**

Get the collection of clusters detected in the image.

Implements [multiscale::analysis::Detector](#).

Definition at line 441 of file RegionDetector.cpp.

References multiscale::analysis::Region, and regions.

**6.160.3.26 std::string RegionDetector::getDetectorTypeAsString ( )  
[override, private, virtual]**

Get the type of the detector as a string.

Implements [multiscale::analysis::Detector](#).

Definition at line 147 of file RegionDetector.cpp.

References DETECTOR\_TYPE.

**6.160.3.27 int RegionDetector::getEpsilon ( )**

Get the value of field epsilon.

Definition at line 45 of file RegionDetector.cpp.

References epsilon.

**6.160.3.28 int RegionDetector::getMorphologicalCloselterations ( )**

Get the value of field morphologicalCloselterations.

Definition at line 41 of file RegionDetector.cpp.

References morphologicalCloselterations.

**6.160.3.29 int RegionDetector::getOriginXCoordinate ( )**

Get the value of field originXCoordinate.

Definition at line 53 of file RegionDetector.cpp.

References multiscale::analysis::Detector::origin.

#### 6.160.3.30 int RegionDetector::getOriginYCoordinate ( )

Get the value of field originYCoordinate.

Definition at line 57 of file RegionDetector.cpp.

References multiscale::analysis::Detector::origin.

#### 6.160.3.31 int RegionDetector::getRegionAreaThresh ( )

Get the value of field regionAreaThresh.

Definition at line 49 of file RegionDetector.cpp.

References regionAreaThresh.

#### 6.160.3.32 std::vector< Region > const & RegionDetector::getRegions ( )

Get a const reference to the vector of detected regions.

Definition at line 65 of file RegionDetector.cpp.

References regions.

#### 6.160.3.33 int RegionDetector::getThresholdValue ( )

Get the value of field thresholdValue.

Definition at line 61 of file RegionDetector.cpp.

References thresholdValue.

#### 6.160.3.34 void RegionDetector::initialiseDetectorSpecificFields ( ) [override, private, virtual]

Initialise the vision members.

Implements [multiscale::analysis::Detector](#).

Definition at line 123 of file RegionDetector.cpp.

References alpha, beta, blurKernelSize, epsilon, morphologicalCloselterations, regionAreaThresh, and thresholdValue.

#### 6.160.3.35 void RegionDetector::initialiseDetectorSpecificImageDependentFields ( ) [override, private, virtual]

Initialisation of the detector specific image dependent values.

Implements [multiscale::analysis::Detector](#).

Definition at line 133 of file RegionDetector.cpp.

```
6.160.3.36 bool RegionDetector::isValidContour ( const std::vector< cv::Point > &
contour ) [private]
```

Check if the contour is valid.

The contour is valid if its area > regionAreaThreshold

#### Parameters

|                |                   |
|----------------|-------------------|
| <i>contour</i> | The given contour |
|----------------|-------------------|

Definition at line 360 of file RegionDetector.cpp.

References [multiscale::analysis::SpatialMeasureCalculator::computePolygonArea\(\)](#), and [regionAreaThresh](#).

Referenced by [createPolygonsFromContours\(\)](#).

```
6.160.3.37 bool RegionDetector::isValidHole ( const std::vector< cv::Point > & hole,
const std::vector< cv::Point > & outerPolygon ) [private]
```

Check if the hole contained by the outer polygon is valid.

The hole is valid if its area > THRESHOLD\_HOLE\_AREA

#### Parameters

|                      |                                                          |
|----------------------|----------------------------------------------------------|
| <i>hole</i>          | The contour of the hole                                  |
| <i>outer-Polygon</i> | The contour of the outer polygon which contains the hole |

Definition at line 366 of file RegionDetector.cpp.

References [multiscale::analysis::SpatialMeasureCalculator::computePolygonHoleArea\(\)](#), and [THRESHOLD\\_HOLE\\_AREA](#).

Referenced by [setPolygonInnerContours\(\)](#).

```
6.160.3.38 void RegionDetector::morphologicalClose ( cv::Mat & image ) [private]
```

Apply the morphological close operator on the image.

#### Parameters

|              |           |
|--------------|-----------|
| <i>image</i> | The image |
|--------------|-----------|

Definition at line 175 of file RegionDetector.cpp.

References morphologicalCloselterations.

Referenced by processImageAndDetect().

```
6.160.3.39 void RegionDetector::outputRegionInnerBordersToImage ( const
std::vector< std::vector< cv::Point > > & innerBorders, cv::Mat & outputImage )
[private]
```

Output the inner border polygons of a region to the outputImage instance.

#### Parameters

|                     |                                                         |
|---------------------|---------------------------------------------------------|
| <i>innerBorders</i> | The polygons defining the inner border(s) of the region |
| <i>outputImage</i>  | The given output image                                  |

Definition at line 475 of file RegionDetector.cpp.

References DISPLAY\_LINE\_THICKNESS, multiscale::analysis::Detector::INTENSITY\_MAX, and POLYGON\_CLOSED.

Referenced by outputRegionToImage().

```
6.160.3.40 void RegionDetector::outputRegionOuterBorderToImage ( const
std::vector< cv::Point > & outerBorder, cv::Mat & outputImage ) [private]
```

Output the outer border polygon of a region to the outputImage instance.

#### Parameters

|                    |                                                     |
|--------------------|-----------------------------------------------------|
| <i>outerBorder</i> | The polygon defining the outer border of the region |
| <i>outputImage</i> | The given output image                              |

Definition at line 469 of file RegionDetector.cpp.

References DISPLAY\_LINE\_THICKNESS, multiscale::analysis::Detector::INTENSITY\_MAX, and POLYGON\_CLOSED.

Referenced by outputRegionToImage().

```
6.160.3.41 void RegionDetector::outputRegionToImage ( const Region & region,
cv::Mat & outputImage ) [private]
```

Output the region to the outputImage instance.

#### Parameters

|                    |                        |
|--------------------|------------------------|
| <i>region</i>      | The given region       |
| <i>outputImage</i> | The given output image |

Definition at line 464 of file RegionDetector.cpp.

References multiscale::analysis::Region::getInnerBorderPolygons(), multiscale::analysis::Region::getOuterBorderPolygon(), multiscale::analysis::Detector::outputImage, outputRegionInnerBordersToImage(), and outputRegionOuterBorderToImage().

Referenced by outputResultsToImage().

**6.160.3.42 void RegionDetector::outputResultsToImage( ) [override, private, virtual]**

Output the results to the outputImage instance.

Implements multiscale::analysis::Detector.

Definition at line 451 of file RegionDetector.cpp.

References multiscale::analysis::Detector::image, multiscale::analysis::Detector::outputImage, outputRegionToImage(), and regions.

**6.160.3.43 void RegionDetector::processImageAndDetect( ) [override, private, virtual]**

Process the given image.

Apply filters to the image, threshold it, find its contours, approximate the polygons from these contours. Afterwards, process the polygons to find their distance from the origin, their area and the angle determined by the points from the contour which are on the edge and the closest point to the origin. Return all the polygons together with the processed information as a vector of regions.

Implements multiscale::analysis::Detector.

Definition at line 151 of file RegionDetector.cpp.

References changeContrastAndBrightness(), computeAverageMeasures(), findRegions(), morphologicalClose(), regions, smoothImage(), and thresholdImage().

**6.160.3.44 void RegionDetector::setAlpha( int alpha )**

Set the value of field alpha.

**Parameters**

|              |                |
|--------------|----------------|
| <i>alpha</i> | Value of alpha |
|--------------|----------------|

Definition at line 69 of file RegionDetector.cpp.

References alpha, and multiscale::analysis::Detector::setDetectorSpecificFieldsInitialisationFlag().

**6.160.3.45 void RegionDetector::setBeta( int beta )**

Set the value of field beta.

**Parameters**

|             |               |
|-------------|---------------|
| <i>beta</i> | Value of beta |
|-------------|---------------|

Definition at line 75 of file RegionDetector.cpp.

References *beta*, and *multiscale::analysis::Detector::setDetectorSpecificFieldsInitialisationFlag()*.

**6.160.3.46 void RegionDetector::setBlurKernelSize ( int *blurKernelSize* )**

Set the value of field *blurKernelSize*.

**Parameters**

|                             |                                |
|-----------------------------|--------------------------------|
| <i>blurKernel-<br/>Size</i> | Value of <i>blurKernelSize</i> |
|-----------------------------|--------------------------------|

Definition at line 81 of file RegionDetector.cpp.

References *blurKernelSize*, and *multiscale::analysis::Detector::setDetectorSpecificFieldsInitialisationFlag()*.

**6.160.3.47 void RegionDetector::setEpsilon ( int *epsilon* )**

Set the value of field *epsilon*.

**Parameters**

|                |                         |
|----------------|-------------------------|
| <i>epsilon</i> | Value of <i>epsilon</i> |
|----------------|-------------------------|

Definition at line 87 of file RegionDetector.cpp.

References *epsilon*, and *multiscale::analysis::Detector::setDetectorSpecificFieldsInitialisationFlag()*.

**6.160.3.48 void RegionDetector::setMorphologicalCloselterations ( int *morphologicalCloselterations* )**

Set the value of field *morphologicalCloselterations*.

**Parameters**

|                                                 |                                              |
|-------------------------------------------------|----------------------------------------------|
| <i>morphological-<br/>Close-<br/>Iterations</i> | Value of <i>morphologicalCloselterations</i> |
|-------------------------------------------------|----------------------------------------------|

Definition at line 93 of file RegionDetector.cpp.

References morphologicalCloselterations, and multiscale::analysis::Detector::setDetectorSpecificFieldsInitialisationFlag().

#### 6.160.3.49 void RegionDetector::setOriginXCoordinate ( int *originXCoordinate* )

Set the value of field originXCoordinate.

##### Parameters

|                           |                            |
|---------------------------|----------------------------|
| <i>originX-Coordinate</i> | Value of originXCoordinate |
|---------------------------|----------------------------|

Definition at line 99 of file RegionDetector.cpp.

References multiscale::analysis::Detector::origin, and multiscale::analysis::Detector::setDetectorSpecificFieldsInitialisationFlag().

#### 6.160.3.50 void RegionDetector::setOriginYCoordinate ( int *originYCoordinate* )

Set the value of field originYCoordinate.

##### Parameters

|                           |                            |
|---------------------------|----------------------------|
| <i>originY-Coordinate</i> | Value of originYCoordinate |
|---------------------------|----------------------------|

Definition at line 105 of file RegionDetector.cpp.

References multiscale::analysis::Detector::origin, and multiscale::analysis::Detector::setDetectorSpecificFieldsInitialisationFlag().

#### 6.160.3.51 void RegionDetector::setPolygonInnerContours ( int *contourIndex*, const std::vector< std::vector< cv::Point > > & *contours*, const std::vector< cv::Vec4i > & *hierarchy*, Polygon & *polygon* ) [private]

Set the inner contours of the polygon.

##### Parameters

|                      |                                                          |
|----------------------|----------------------------------------------------------|
| <i>contour-Index</i> | The index of the outer contour                           |
| <i>contours</i>      | The collection of all contours                           |
| <i>hierarchy</i>     | The information regarding the hierarchy between contours |
| <i>polygon</i>       | The polygon for which the outer contour is set           |

Definition at line 326 of file RegionDetector.cpp.

References HIERARCHY\_PARENT\_INDEX, and isValidHole().

Referenced by createPolygon().

---

**6.160.3.52 void RegionDetector::setPolygonOuterContour ( int *contourIndex*, const std::vector< std::vector< cv::Point > > & *contours*, const std::vector< cv::Vec4i > & *hierarchy*, Polygon & *polygon* ) [private]**

Set the outer contour of the polygon.

**Parameters**

|                     |                                                          |
|---------------------|----------------------------------------------------------|
| <i>contourIndex</i> | The index of the outer contour                           |
| <i>contours</i>     | The collection of all contours                           |
| <i>hierarchy</i>    | The information regarding the hierarchy between contours |
| <i>polygon</i>      | The polygon for which the outer contour is set           |

Definition at line 321 of file RegionDetector.cpp.

Referenced by createPolygon().

**6.160.3.53 void RegionDetector::setRegionAreaThresh ( int *regionAreaThresh* )**

Set the value of field regionAreaThresh.

**Parameters**

|                         |                           |
|-------------------------|---------------------------|
| <i>regionAreaThresh</i> | Value of regionAreaThresh |
|-------------------------|---------------------------|

Definition at line 111 of file RegionDetector.cpp.

References regionAreaThresh, and multiscale::analysis::Detector::setDetectorSpecificFieldsInitialisationFlag().

**6.160.3.54 void RegionDetector::setThresholdValue ( int *thresholdValue* )**

Set the value of field thresholdValue.

**Parameters**

|                       |                         |
|-----------------------|-------------------------|
| <i>thresholdValue</i> | Value of thresholdValue |
|-----------------------|-------------------------|

Definition at line 117 of file RegionDetector.cpp.

References multiscale::analysis::Detector::setDetectorSpecificFieldsInitialisationFlag(), and thresholdValue.

**6.160.3.55 void RegionDetector::smoothImage ( cv::Mat & *image* ) [private]**

Smooth out differences in the image.

Apply a Gaussian blur filter

**Parameters**

|              |           |
|--------------|-----------|
| <i>image</i> | The image |
|--------------|-----------|

Definition at line 167 of file RegionDetector.cpp.

References blurKernelSize.

Referenced by processImageAndDetect().

**6.160.3.56 void RegionDetector::thresholdImage ( const cv::Mat & *image*, cv::Mat & *thresholdedImage* ) [private]**

Apply the threshold filter on the image.

**Parameters**

|                          |                       |
|--------------------------|-----------------------|
| <i>image</i>             | The image             |
| <i>thresholded-Image</i> | The thresholded image |

Definition at line 181 of file RegionDetector.cpp.

References THRESHOLD\_MAX, and thresholdValue.

Referenced by processImageAndDetect().

## 6.160.4 Member Data Documentation

**6.160.4.1 int multiscale::analysis::RegionDetector::alpha [private]**

Alpha for brightness and contrast adjustments

Definition at line 28 of file RegionDetector.hpp.

Referenced by changeContrastAndBrightness(), createDetectorSpecificTrackbars(), getAlpha(), initialiseDetectorSpecificFields(), RegionDetector(), and setAlpha().

**6.160.4.2 const int RegionDetector::ALPHA\_MAX = 1000 [static, private]**

Definition at line 409 of file RegionDetector.hpp.

Referenced by convertAlpha(), and createDetectorSpecificTrackbars().

**6.160.4.3 const double RegionDetector::ALPHA\_REAL\_MAX = 3.0 [static, private]**

Definition at line 404 of file RegionDetector.hpp.

Referenced by convertAlpha().

**6.160.4.4 const double RegionDetector::ALPHA\_REAL\_MIN = 1.0 [static, private]**

Definition at line 403 of file RegionDetector.hpp.

Referenced by convertAlpha().

**6.160.4.5 int multiscale::analysis::RegionDetector::beta [private]**

Beta for brightness and contrast adjustments

Definition at line 29 of file RegionDetector.hpp.

Referenced by changeContrastAndBrightness(), createDetectorSpecificTrackbars(), getBeta(), initialiseDetectorSpecificFields(), RegionDetector(), and setBeta().

**6.160.4.6 const int RegionDetector::BETA\_MAX = 200 [static, private]**

Definition at line 410 of file RegionDetector.hpp.

Referenced by convertBeta(), and createDetectorSpecificTrackbars().

**6.160.4.7 const int RegionDetector::BETA\_REAL\_MAX = 100 [static, private]**

Definition at line 407 of file RegionDetector.hpp.

Referenced by convertBeta().

**6.160.4.8 const int RegionDetector::BETA\_REAL\_MIN = -100 [static, private]**

Definition at line 406 of file RegionDetector.hpp.

Referenced by convertBeta().

**6.160.4.9 int multiscale::analysis::RegionDetector::blurKernelSize [private]**

Kernel size for Gaussian blur

Definition at line 30 of file RegionDetector.hpp.

Referenced by createDetectorSpecificTrackbars(), getBlurKernelSize(), initialiseDetectorSpecificFields(), RegionDetector(), setBlurKernelSize(), and smoothImage().

6.160.4.10 `const int RegionDetector::CANNY_THRESH_MAX = 100 [static, private]`

Definition at line 413 of file RegionDetector.hpp.

6.160.4.11 `const bool RegionDetector::CONTOUR_AREA_ORIENTED = false [static, private]`

Definition at line 401 of file RegionDetector.hpp.

6.160.4.12 `const std::string RegionDetector::DETECTOR_TYPE = "Regions" [static, private]`

Definition at line 385 of file RegionDetector.hpp.

Referenced by `getDetectorTypeAsString()`.

6.160.4.13 `const int RegionDetector::DISPLAY_LINE_THICKNESS = 1 [static, private]`

Definition at line 423 of file RegionDetector.hpp.

Referenced by `outputRegionInnerBordersToImage()`, and `outputRegionOuterBorderToImage()`.

6.160.4.14 `int multiscale::analysis::RegionDetector::epsilon [private]`

Epsilon for polygon approximation

Definition at line 32 of file RegionDetector.hpp.

Referenced by `approximatePolygonOuterBorder()`, `createDetectorSpecificTrackbars()`, `getEpsilon()`, `initialiseDetectorSpecificFields()`, `RegionDetector()`, and `setEpsilon()`.

6.160.4.15 `const int RegionDetector::EPSILON_MAX = 100 [static, private]`

Definition at line 414 of file RegionDetector.hpp.

Referenced by `createDetectorSpecificTrackbars()`.

6.160.4.16 `const int RegionDetector::HIERARCHY_FIRST_CHILD_INDEX = 2 [static, private]`

Definition at line 398 of file RegionDetector.hpp.

6.160.4.17 **const int RegionDetector::HIERARCHY\_NEXT\_INDEX = 0** [static, private]

Definition at line 396 of file RegionDetector.hpp.

Referenced by createPolygonsFromContours().

6.160.4.18 **const int RegionDetector::HIERARCHY\_PARENT\_INDEX = 3** [static, private]

Definition at line 399 of file RegionDetector.hpp.

Referenced by setPolygonInnerContours().

6.160.4.19 **const int RegionDetector::HIERARCHY\_PREV\_INDEX = 1** [static, private]

Definition at line 397 of file RegionDetector.hpp.

6.160.4.20 **const int RegionDetector::KERNEL\_MAX = 2000** [static, private]

Definition at line 411 of file RegionDetector.hpp.

Referenced by createDetectorSpecificTrackbars().

6.160.4.21 **const int RegionDetector::MORPH\_ITER\_MAX = 100** [static, private]

Definition at line 412 of file RegionDetector.hpp.

Referenced by createDetectorSpecificTrackbars().

6.160.4.22 **int multiscale::analysis::RegionDetector::morphologicalCloseIterations** [private]

Number of iterations for morphological close operator

Definition at line 31 of file RegionDetector.hpp.

Referenced by createDetectorSpecificTrackbars(), getMorphologicalCloselterations(), initialiseDetectorSpecificFields(), morphologicalClose(), RegionDetector(), and setMorphologicalCloselterations().

6.160.4.23 **const bool RegionDetector::POLYGON\_CLOSED = true** [static, private]

Definition at line 421 of file RegionDetector.hpp.

---

Referenced by `outputRegionInnerBordersToImage()`, and `outputRegionOuterBorderToImage()`.

**6.160.4.24 const int RegionDetector::REGION\_AREA\_THRESH\_MAX = 200000**  
[static, private]

Definition at line 415 of file `RegionDetector.hpp`.

Referenced by `createDetectorSpecificTrackbars()`.

**6.160.4.25 int multiscale::analysis::RegionDetector::regionAreaThresh**  
[private]

Threshold for considering a region

Definition at line 33 of file `RegionDetector.hpp`.

Referenced by `createDetectorSpecificTrackbars()`, `getRegionAreaThresh()`, `initialiseDetectorSpecificFields()`, `isValidContour()`, `RegionDetector()`, and `setRegionAreaThresh()`.

**6.160.4.26 std::vector<Region> multiscale::analysis::RegionDetector::regions**  
[private]

Regions detected in the image

Definition at line 36 of file `RegionDetector.hpp`.

Referenced by `clearPreviousDetectionResults()`, `computeSumOfAverageCentroidDistances()`, `getCollectionOfSpatialEntityPseudo3D()`, `getRegions()`, `outputResultsToImage()`, and `processImageAndDetect()`.

**6.160.4.27 const int RegionDetector::THRESHOLD\_CLUSTEREDNESS = 0**  
[static, private]

Definition at line 417 of file `RegionDetector.hpp`.

**6.160.4.28 const int RegionDetector::THRESHOLD\_HOLE\_AREA = 0** [static,  
private]

Definition at line 419 of file `RegionDetector.hpp`.

Referenced by `isValidHole()`.

**6.160.4.29 const int RegionDetector::THRESHOLD\_MAX = 255** [static,  
private]

Definition at line 416 of file `RegionDetector.hpp`.

Referenced by createDetectorSpecificTrackbars(), and thresholdImage().

**6.160.4.30 int multiscale::analysis::RegionDetector::thresholdValue  
[private]**

Value of the threshold for the threshold filter

Definition at line 34 of file RegionDetector.hpp.

Referenced by createDetectorSpecificTrackbars(), getThresholdValue(), initialiseDetectorSpecificFields(), RegionDetector(), setThresholdValue(), and thresholdImage().

**6.160.4.31 const std::string RegionDetector::TRACKBAR\_ALPHA = "Alpha"  
[static, private]**

Definition at line 387 of file RegionDetector.hpp.

Referenced by createDetectorSpecificTrackbars().

**6.160.4.32 const std::string RegionDetector::TRACKBAR\_BETA = "Beta"  
[static, private]**

Definition at line 388 of file RegionDetector.hpp.

Referenced by createDetectorSpecificTrackbars().

**6.160.4.33 const std::string RegionDetector::TRACKBAR\_CANNY = "Canny lower  
threshold" [static, private]**

Definition at line 391 of file RegionDetector.hpp.

**6.160.4.34 const std::string RegionDetector::TRACKBAR\_EPSILON = "Epsilon"  
[static, private]**

Definition at line 392 of file RegionDetector.hpp.

Referenced by createDetectorSpecificTrackbars().

**6.160.4.35 const std::string RegionDetector::TRACKBAR\_KERNEL = "Gaussian blur  
kernel size" [static, private]**

Definition at line 389 of file RegionDetector.hpp.

Referenced by createDetectorSpecificTrackbars().

---

6.160.4.36 `const std::string RegionDetector::TRACKBAR_MORPH = "Morphological open, number of iterations"` [static, private]

Definition at line 390 of file RegionDetector.hpp.

Referenced by `createDetectorSpecificTrackbars()`.

6.160.4.37 `const std::string RegionDetector::TRACKBAR_REGION_AREA_THRESH = "Region area threshold"` [static, private]

Definition at line 393 of file RegionDetector.hpp.

Referenced by `createDetectorSpecificTrackbars()`.

6.160.4.38 `const std::string RegionDetector::TRACKBAR_THRESHOLD = "Threshold value"` [static, private]

Definition at line 394 of file RegionDetector.hpp.

Referenced by `createDetectorSpecificTrackbars()`.

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

- RegionDetector.hpp
- RegionDetector.cpp

## 6.161 multiscale::verification::ProbabilityErrorHandler::result<typename, typename, typename> Struct Template Reference

Structure for specifying the type of the result.

```
#include <ProbabilityErrorHandler.hpp>
```

### Public Types

- `typedef void type`

#### 6.161.1 Detailed Description

```
template<typename, typename, typename>struct multiscale::verification::ProbabilityErrorHandler::result<typename, typename, typename>
```

Structure for specifying the type of the result.

Definition at line 23 of file ProbabilityErrorHandler.hpp.

### 6.161.2 Member Typedef Documentation

6.161.2.1 template<typename , typename , typename > typedef void  
multiscale::verification::ProbabilityErrorHandler::result< typename,  
typename, typename >::type

Definition at line 23 of file ProbabilityErrorHandler.hpp.

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

- ProbabilityErrorHandler.hpp

## 6.162 multiscale::verification::UnexpectedErrorHandler:- :result< typename, typename, typename > Struct Template Reference

Structure for specifying the type of the result.

```
#include <UnexpectedErrorHandler.hpp>
```

### Public Types

- typedef void [type](#)

#### 6.162.1 Detailed Description

```
template<typename, typename, typename>struct multiscale::verification::UnexpectedToken-  
ErrorHandler::result< typename, typename, typename >
```

Structure for specifying the type of the result.

Definition at line 23 of file UnexpectedErrorHandler.hpp.

### 6.162.2 Member Typedef Documentation

6.162.2.1 template<typename , typename , typename > typedef void  
multiscale::verification::UnexpectedErrorHandler::result<  
typename, typename, typename >::type

Definition at line 23 of file UnexpectedErrorHandler.hpp.

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

- UnexpectedErrorHandler.hpp

## 6.163 multiscale::RGBColourGenerator Class Reference

Generate a RGB colour.

```
#include <RGBColourGenerator.hpp>
```

### Public Member Functions

- std::string [generate](#) (double concentrationMin, double concentrationMax, double concentration)  
*Generate a RGB colour for the given concentration.*
- cv::Scalar [generate](#) (cv::RNG &randomNumberGenerator)  
*Generate a random RGB colour.*

### Static Public Attributes

- static const int [HUE\\_MIN](#) = 0
- static const int [HUE\\_MAX](#) = 120
- static const int [SATURATION](#) = 1
- static const int [VALUE](#) = 1

### Private Member Functions

- std::string [convertHSVToRGB](#) (double hue, double saturation, double value)  
*Convert a colour from HSV to RGB colour space.*
- void [computeRGBValues](#) (int huePrime, double X, double chroma, double m)  
*Compute RGB values from HSV specific values.*
- std::string [convertRGBToString](#) ()  
*Convert the RGB colour to a string.*

### Private Attributes

- double [red](#)
- double [green](#)
- double [blue](#)

### 6.163.1 Detailed Description

Generate a RGB colour.

Generate a RGB colour given the possible range for concentrations and the value of one of the concentrations

The conversion HSV->RGB is based on the wikipedia page on this topic

Definition at line 18 of file RGBColourGenerator.hpp.

### 6.163.2 Member Function Documentation

6.163.2.1 void RGBColourGenerator::computeRGBValues ( int *huePrime*, double *X*, double *chroma*, double *m* ) [private]

Compute RGB values from HSV specific values.

#### Parameters

|                 |        |
|-----------------|--------|
| <i>huePrime</i> | Hue'   |
| <i>X</i>        | X      |
| <i>chroma</i>   | Chroma |
| <i>m</i>        | m      |

Definition at line 41 of file RGBColourGenerator.cpp.

References blue, green, and red.

Referenced by convertHSVToRGB().

6.163.2.2 std::string RGBColourGenerator::convertHSVToRGB ( double *hue*, double *saturation*, double *value* ) [private]

Convert a colour from HSV to RGB colour space.

#### Parameters

|                   |            |
|-------------------|------------|
| <i>hue</i>        | Hue        |
| <i>saturation</i> | Saturation |
| <i>value</i>      | Value      |

Definition at line 27 of file RGBColourGenerator.cpp.

References computeRGBValues(), and convertRGBToString().

Referenced by generate().

6.163.2.3 std::string RGBColourGenerator::convertRGBToString ( ) [private]

Convert the RGB colour to a string.

Definition at line 85 of file RGBColourGenerator.cpp.

References blue, green, and red.

Referenced by convertHSVToRGB().

6.163.2.4 std::string RGBColourGenerator::generate ( double *concentrationMin*, double *concentrationMax*, double *concentration* )

Generate a RGB colour for the given concentration.

Generate a RGB colour considering the range of values a concentration can have and the value of the concentration

#### Parameters

|                          |                                                             |
|--------------------------|-------------------------------------------------------------|
| <i>concentration_Min</i> | The minimum of the range of values a concentration can take |
| <i>concentration_Max</i> | The maximum of the range of values a concentration can take |
| <i>concentration</i>     | The concentration                                           |

Definition at line 11 of file RGBColourGenerator.cpp.

References convertHSVToRGB(), HUE\_MAX, HUE\_MIN, SATURATION, and VALUE.

Referenced by multiscale::analysis::SimulationClusterDetector::outputResultsToImage().

#### 6.163.2.5 cv::Scalar RGBColourGenerator::generate ( cv::RNG & randomNumberGenerator )

Generate a random RGB colour.

Generate a random RGB colour using the given random number generator

#### Parameters

|                                |                         |
|--------------------------------|-------------------------|
| <i>random-Number-Generator</i> | Random number generator |
|--------------------------------|-------------------------|

Definition at line 21 of file RGBColourGenerator.cpp.

### 6.163.3 Member Data Documentation

#### 6.163.3.1 double multiscale::RGBColourGenerator::blue [private]

The amount of blue

Definition at line 24 of file RGBColourGenerator.hpp.

Referenced by computeRGBValues(), and convertRGBToString().

#### 6.163.3.2 double multiscale::RGBColourGenerator::green [private]

The amount of green

Definition at line 23 of file RGBColourGenerator.hpp.

Referenced by computeRGBValues(), and convertRGBToString().

**6.163.3.3 const int RGBColourGenerator::HUE\_MAX = 120 [static]**

Definition at line 73 of file RGBColourGenerator.hpp.

Referenced by generate().

**6.163.3.4 const int RGBColourGenerator::HUE\_MIN = 0 [static]**

Definition at line 72 of file RGBColourGenerator.hpp.

Referenced by generate().

**6.163.3.5 double multiscale::RGBColourGenerator::red [private]**

The amount of red

Definition at line 22 of file RGBColourGenerator.hpp.

Referenced by computeRGBValues(), and convertRGBToString().

**6.163.3.6 const int RGBColourGenerator::SATURATION = 1 [static]**

Definition at line 74 of file RGBColourGenerator.hpp.

Referenced by generate().

**6.163.3.7 const int RGBColourGenerator::VALUE = 1 [static]**

Definition at line 75 of file RGBColourGenerator.hpp.

Referenced by generate().

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

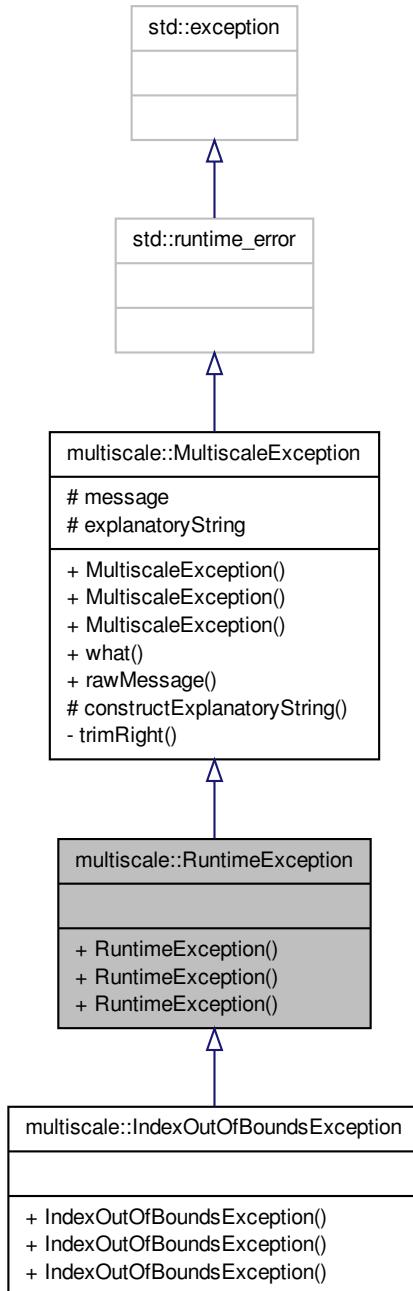
- RGBColourGenerator.hpp
- RGBColourGenerator.cpp

## 6.164 multiscale::RuntimeException Class Reference

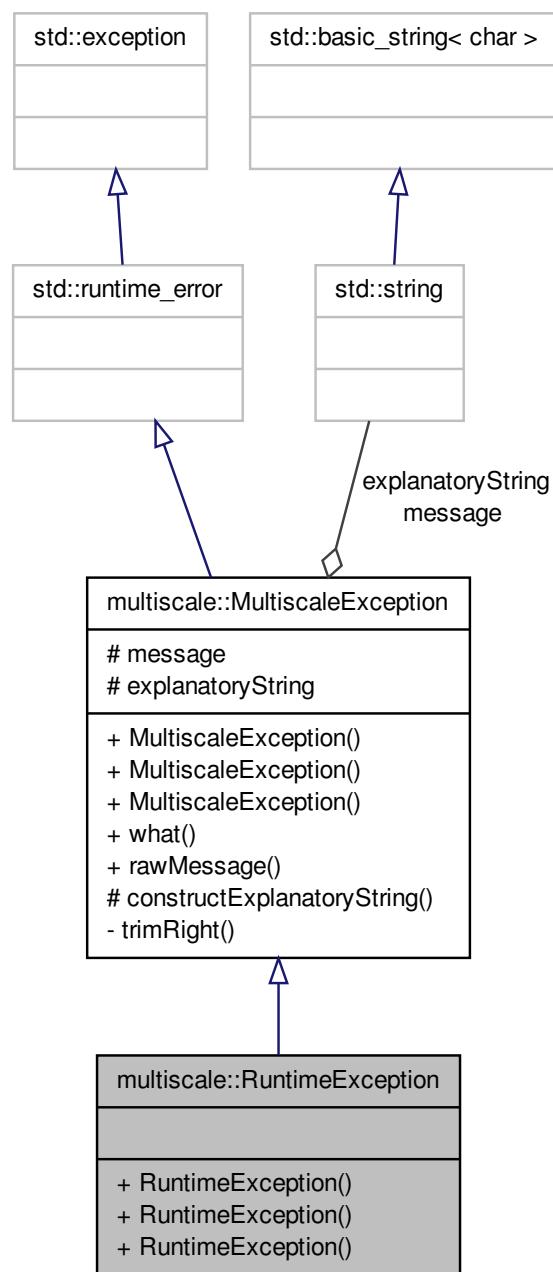
Class for representing runtime exceptions.

```
#include <RuntimeException.hpp>
```

Inheritance diagram for multiscale::RuntimeException:



Collaboration diagram for multiscale::RuntimeException:



## Public Member Functions

- [RuntimeException \(\)](#)
- [RuntimeException \(const std::string &file, int line, const std::string &msg\)](#)
- [RuntimeException \(const std::string &file, int line, const char \\*msg\)](#)

### 6.164.1 Detailed Description

Class for representing runtime exceptions.

Definition at line 12 of file `RuntimeException.hpp`.

### 6.164.2 Constructor & Destructor Documentation

#### 6.164.2.1 `multiscale::RuntimeException::RuntimeException( ) [inline]`

Definition at line 16 of file `RuntimeException.hpp`.

#### 6.164.2.2 `multiscale::RuntimeException::RuntimeException( const std::string & file, int line, const std::string & msg ) [inline, explicit]`

Definition at line 18 of file `RuntimeException.hpp`.

#### 6.164.2.3 `multiscale::RuntimeException::RuntimeException( const std::string & file, int line, const char * msg ) [inline, explicit]`

Definition at line 23 of file `RuntimeException.hpp`.

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

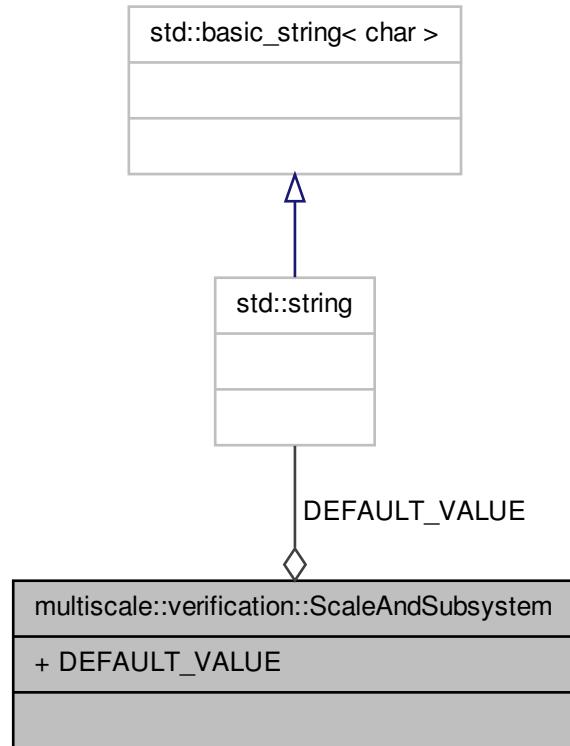
- `RuntimeException.hpp`

## 6.165 `multiscale::verification::ScaleAndSubsystem` Class Reference

Class for representing a scale and subsystem.

```
#include <ScaleAndSubsystem.hpp>
```

Collaboration diagram for multiscale::verification::ScaleAndSubsystem:



### Static Public Attributes

- static const std::string `DEFAULT_VALUE` = ""

#### 6.165.1 Detailed Description

Class for representing a scale and subsystem.

Definition at line 12 of file ScaleAndSubsystem.hpp.

#### 6.165.2 Member Data Documentation

6.165.2.1 const std::string ScaleAndSubsystem::DEFAULT\_VALUE = "" [static]

The default scale and subsystem value used when no explicit scale and subsystem is associated to a spatial entity and/or numeric state variable

Definition at line 17 of file ScaleAndSubsystem.hpp.

Referenced by multiscale::verification::SpatialTemporalDataReader::addNumericStateVariableToTimePoint(), multiscale::verification::SpatialTemporalDataReader::setSpatialEntityScaleAndSubsystem(), and multiscale::verification::ScaleAndSubsystemEvaluator::validateScaleAndSubsystem().

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

- ScaleAndSubsystem.hpp

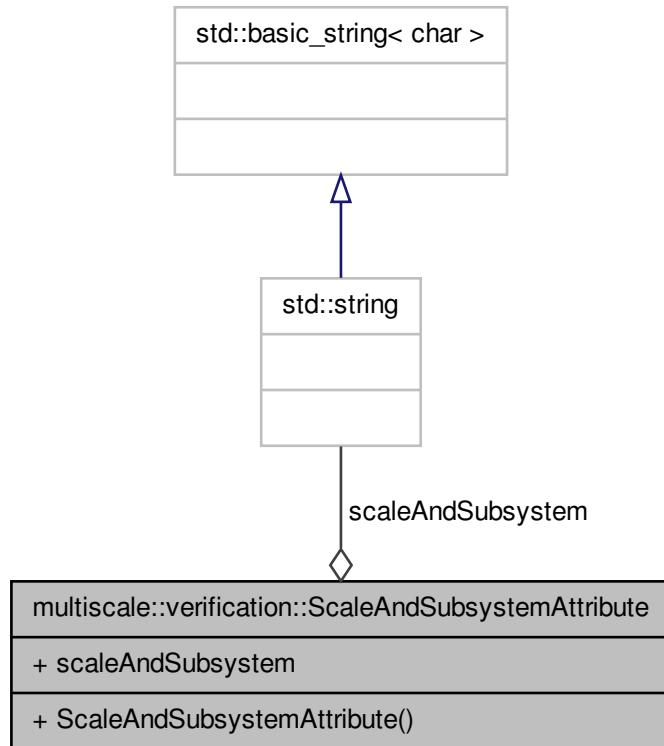
- ScaleAndSubsystem.cpp

## 6.166 multiscale::verification::ScaleAndSubsystemAttribute Class Reference

Class for representing a scale and subsystem attribute.

```
#include <ScaleAndSubsystemAttribute.hpp>
```

Collaboration diagram for multiscale::verification::ScaleAndSubsystemAttribute:



### Public Member Functions

- `ScaleAndSubsystemAttribute (const std::string &scaleAndSubsystem=ScaleAndSubsystem::DEFAULT_VALUE)`

### Public Attributes

- `std::string scaleAndSubsystem`

#### 6.166.1 Detailed Description

Class for representing a scale and subsystem attribute.

Definition at line 16 of file `ScaleAndSubsystemAttribute.hpp`.

### 6.166.2 Constructor & Destructor Documentation

6.166.2.1 **multiscale::verification::ScaleAndSubsystemAttribute::ScaleAndSubsystemAttribute ( const std::string & *scaleAndSubsystem* = ScaleAndSubsystem::DEFAULT\_VALUE ) [inline]**

Definition at line 24 of file ScaleAndSubsystemAttribute.hpp.

### 6.166.3 Member Data Documentation

6.166.3.1 **std::string multiscale::verification::ScaleAndSubsystemAttribute::scaleAndSubsystem**

The considered scale and subsystem

Definition at line 20 of file ScaleAndSubsystemAttribute.hpp.

Referenced by multiscale::verification::ConstraintVisitor::filterSpatialEntitiesWrtScaleAndSubsystem().

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

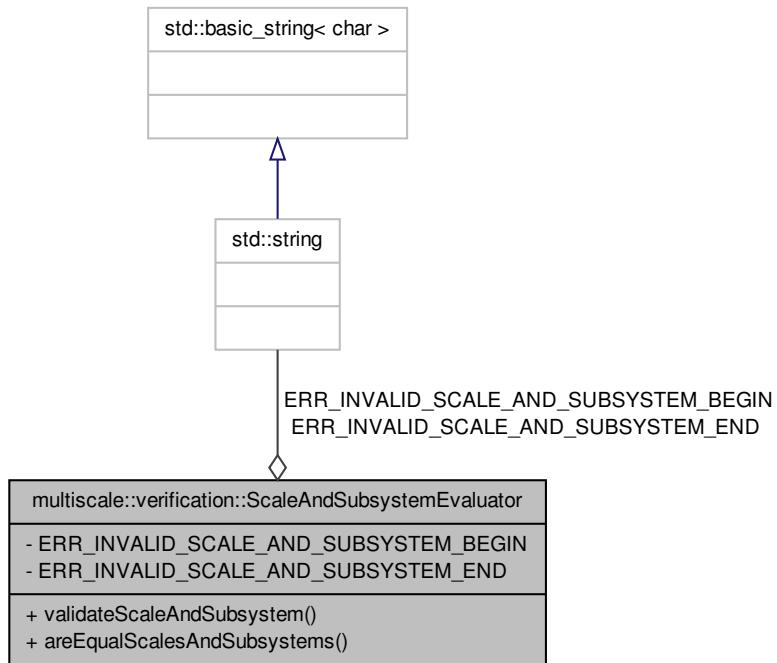
- ScaleAndSubsystemAttribute.hpp

## 6.167 multiscale::verification::ScaleAndSubsystemEvaluator - Class Reference

---

```
#include <ScaleAndSubsystemEvaluator.hpp>
```

Collaboration diagram for multiscale::verification::ScaleAndSubsystemEvaluator:



### Static Public Member Functions

- static void [validateScaleAndSubsystem](#) (const std::string &scaleAndSubsystem, const [MultiscaleArchitectureGraph](#) &multiscaleArchitectureGraph)  
*Check if the provided scale and subsystem exists and otherwise throw an exception.*
- static bool [areEqualScalesAndSubsystems](#) (const std::string &lhsScaleAndSubsystem, const std::string &rhsScaleAndSubsystem)  
*Check if the provided scales and subsystems are equal.*

### Static Private Attributes

- static const std::string [ERR\\_INVALID\\_SCALE\\_AND\\_SUBSYSTEM\\_BEGIN](#) = "- The scale and subsystem ("
- static const std::string [ERR\\_INVALID\\_SCALE\\_AND\\_SUBSYSTEM\\_END](#) = ") specified in the logic statement does not exist in the multiscale architecture graph."

### 6.167.1 Detailed Description

Definition at line 12 of file ScaleAndSubsystemEvaluator.hpp.

### 6.167.2 Member Function Documentation

**6.167.2.1 bool ScaleAndSubsystemEvaluator::areEqualScalesAndSubsystems ( const std::string & *lhsScaleAndSubsystem*, const std::string & *rhsScaleAndSubsystem* ) [static]**

Check if the provided scales and subsystems are equal.

Precondition: The provided scales and subsystem are valid.

#### Parameters

|                                         |                                         |
|-----------------------------------------|-----------------------------------------|
| <i>lhsScale-<br/>And-<br/>Subsystem</i> | The left hand side scale and subsystem  |
| <i>rhsScale-<br/>And-<br/>Subsystem</i> | The right hand side scale and subsystem |

Definition at line 27 of file ScaleAndSubsystemEvaluator.cpp.

Referenced by multiscale::verification::ComparatorEvaluator::evaluate().

**6.167.2.2 void ScaleAndSubsystemEvaluator::validateScaleAndSubsystem ( const std::string & *scaleAndSubsystem*, const MultiscaleArchitectureGraph & *multiscaleArchitectureGraph* ) [static]**

Check if the provided scale and subsystem exists and otherwise throw an exception.

#### Parameters

|                                                |                                                                                |
|------------------------------------------------|--------------------------------------------------------------------------------|
| <i>scaleAnd-<br/>Subsystem</i>                 | The provided scale and subsystem                                               |
| <i>multiscale-<br/>Architecture-<br/>Graph</i> | The multiscale architecture graph which stores all valid scales and subsystems |

Definition at line 9 of file ScaleAndSubsystemEvaluator.cpp.

References multiscale::verification::ScaleAndSubsystem::DEFAULT\_VALUE, ERR\_INVALID\_SCALE\_AND\_SUBSYSTEM\_BEGIN, ERR\_INVALID\_SCALE\_AND\_SUBSYSTEM\_END, and multiscale::verification::MultiscaleArchitectureGraph::existsScaleAndSubsystem().

Referenced by multiscale::verification::NumericStateVariableEvaluator::evaluate(), and multiscale::verification::ConstraintVisitor::filterSpatialEntitiesWrtScaleAndSubsystem().

### 6.167.3 Member Data Documentation

**6.167.3.1 const std::string ScaleAndSubsystemEvaluator::ERR\_INVALID\_SCALE\_-  
AND\_SUBSYSTEM\_BEGIN = "The scale and subsystem (" [static,  
private]**

Definition at line 37 of file ScaleAndSubsystemEvaluator.hpp.

Referenced by validateScaleAndSubsystem().

**6.167.3.2 const std::string ScaleAndSubsystemEvaluator::ERR\_INVALID\_SCALE\_-  
AND\_SUBSYSTEM\_END = ") specified in the logic statement does not exist in the  
multiscale architecture graph." [static, private]**

Definition at line 38 of file ScaleAndSubsystemEvaluator.hpp.

Referenced by validateScaleAndSubsystem().

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

- ScaleAndSubsystemEvaluator.hpp
  
- ScaleAndSubsystemEvaluator.cpp

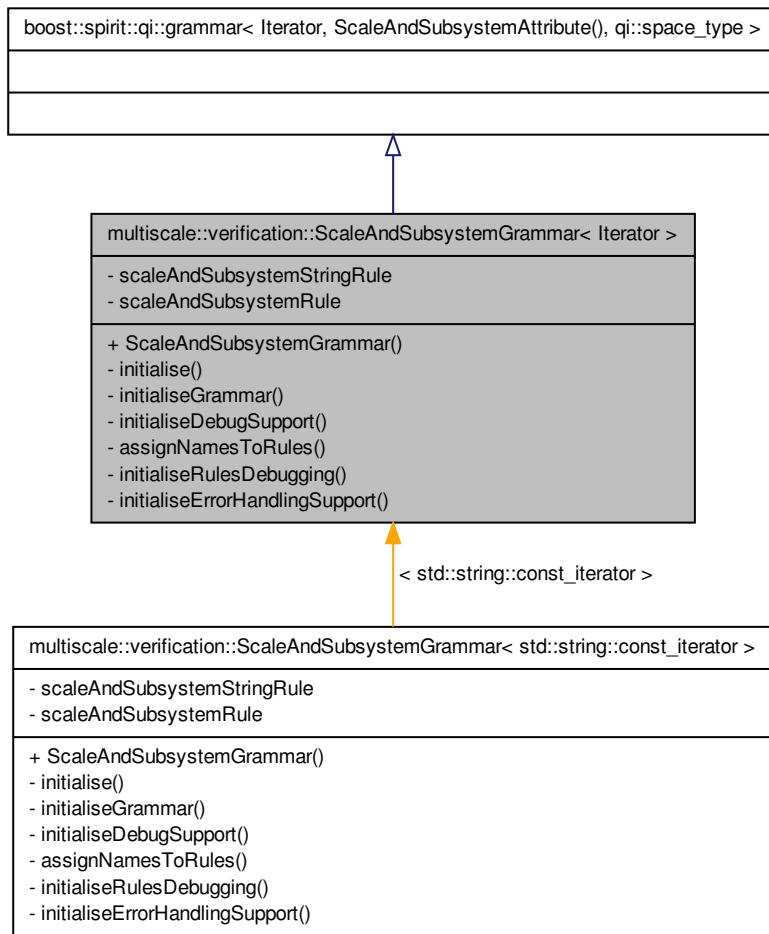
## **6.168 multiscale::verification::ScaleAndSubsystemGrammar< - Iterator > Class Template Reference**

The grammar for parsing scale and subsystem statements.

```
#include <ScaleAndSubsystemGrammar.hpp>
```

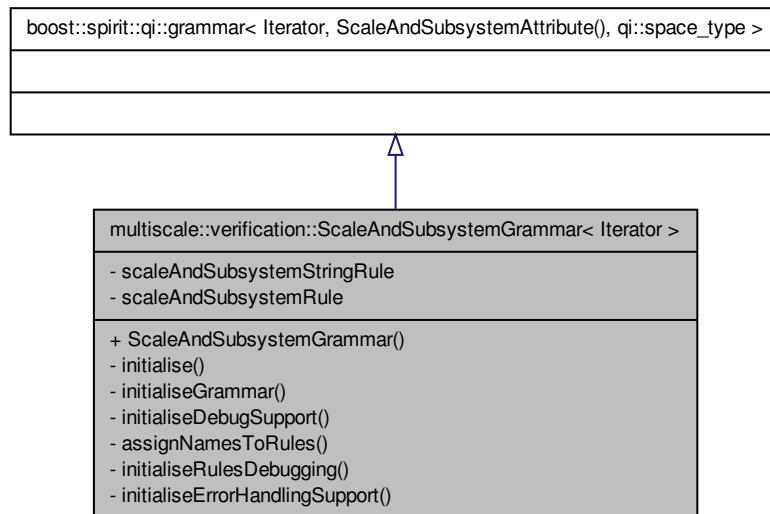
Inheritance diagram for multiscale::verification::ScaleAndSubsystemGrammar< Iterator

&gt;:



Collaboration diagram for `multiscale::verification::ScaleAndSubsystemGrammar< - >`

Iterator >:



## Public Member Functions

- [ScaleAndSubsystemGrammar \(\)](#)

## Private Member Functions

- void [initialise \(\)](#)  
*Initialisation function.*
- void [initialiseGrammar \(\)](#)  
*Initialise the grammar.*
- void [initialiseDebugSupport \(\)](#)  
*Initialise debug support.*
- void [assignNamesToRules \(\)](#)  
*Assign names to the rules.*
- void [initialiseRulesDebugging \(\)](#)  
*Initialise the debugging of rules.*
- void [initialiseErrorHandlingSupport \(\)](#)  
*Initialise the error handling routines.*

## Private Attributes

- `ScaleAndSubsystemStringGrammar < Iterator >` `scaleAndSubsystemStringRule`
- `qi::rule < Iterator, ScaleAndSubsystemAttribute(), qi::space_type >` `scaleAnd-SubsystemRule`

### 6.168.1 Detailed Description

```
template<typename Iterator>class multiscale::verification::ScaleAndSubsystemGrammar< Iterator >
```

The grammar for parsing scale and subsystem statements.

Definition at line 23 of file ScaleAndSubsystemGrammar.hpp.

### 6.168.2 Constructor & Destructor Documentation

```
6.168.2.1 template<typename Iterator > multiscale::verification::ScaleAnd-  
SubsystemGrammar< Iterator >::ScaleAndSubsystemGrammar ( )
```

Definition at line 23 of file ScaleAndSubsystemGrammarDefinition.hpp.

References `multiscale::verification::ScaleAndSubsystemGrammar < Iterator >::initialise()`.

### 6.168.3 Member Function Documentation

```
6.168.3.1 template<typename Iterator > void multiscale::verification::Scale-  
AndSubsystemGrammar< Iterator >::assignNamesToRules ( ) [private]
```

Assign names to the rules.

Definition at line 57 of file ScaleAndSubsystemGrammarDefinition.hpp.

```
6.168.3.2 template<typename Iterator > void multiscale::verification::-  
ScaleAndSubsystemGrammar< Iterator >::initialise ( ) [private]
```

Initialisation function.

Definition at line 33 of file ScaleAndSubsystemGrammarDefinition.hpp.

Referenced by `multiscale::verification::ScaleAndSubsystemGrammar < Iterator >::ScaleAndSubsystemGrammar()`.

**6.168.3.3 template<typename Iterator > void multiscale::verification::Scale-  
AndSubsystemGrammar< Iterator >::initialiseDebugSupport ( )**  
[private]

Initialise debug support.

Definition at line 48 of file ScaleAndSubsystemGrammarDefinition.hpp.

**6.168.3.4 template<typename Iterator > void multiscale::verification::ScaleAnd-  
SubsystemGrammar< Iterator >::initialiseErrorHandlingSupport ( )**  
[private]

Initialise the error handling routines.

Definition at line 69 of file ScaleAndSubsystemGrammarDefinition.hpp.

**6.168.3.5 template<typename Iterator > void multiscale::verification::Scale-  
AndSubsystemGrammar< Iterator >::initialiseGrammar ( )**  
[private]

Initialise the grammar.

Definition at line 41 of file ScaleAndSubsystemGrammarDefinition.hpp.

**6.168.3.6 template<typename Iterator > void multiscale::verification::ScaleAnd-  
SubsystemGrammar< Iterator >::initialiseRulesDebugging ( )**  
[private]

Initialise the debugging of rules.

Definition at line 63 of file ScaleAndSubsystemGrammarDefinition.hpp.

#### **6.168.4 Member Data Documentation**

**6.168.4.1 template<typename Iterator> qi::rule<Iterator, ScaleAndSubsystemAttribute(),  
qi::space\_type> multiscale::verification::ScaleAndSubsystemGrammar<  
Iterator >::scaleAndSubsystemRule [private]**

The rule for parsing a scale and subsystem

Definition at line 37 of file ScaleAndSubsystemGrammar.hpp.

**6.168.4.2 template<typename Iterator> ScaleAndSubsystemStringGrammar<Iterator>  
multiscale::verification::ScaleAndSubsystemGrammar< Iterator  
>::scaleAndSubsystemStringRule [private]**

The rule for parsing a string representing a scale and subsystem

Definition at line 31 of file ScaleAndSubsystemGrammar.hpp.

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

- ScaleAndSubsystemGrammar.hpp
- ScaleAndSubsystemGrammarDefinition.hpp

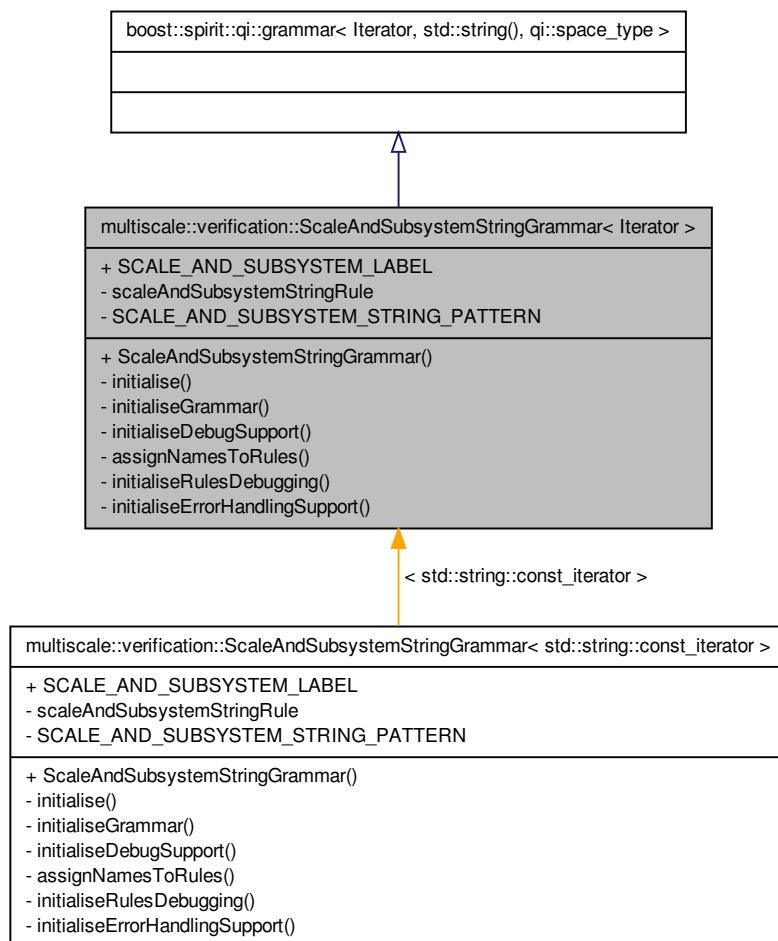
## 6.169 multiscale::verification::ScaleAndSubsystemStringGrammar< Iterator > Class Template Reference

The grammar for parsing scale and subsystem string statements.

```
#include <ScaleAndSubsystemStringGrammar.hpp>
```

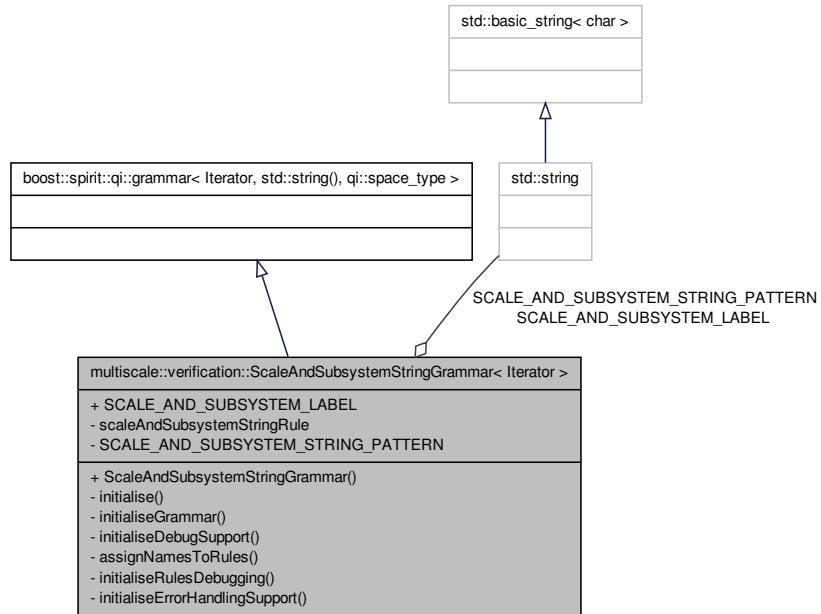
Inheritance diagram for multiscale::verification::ScaleAndSubsystemStringGrammar< -

Iterator >:



Collaboration diagram for multiscale::verification::ScaleAndSubsystemStringGrammar<

Iterator >:



## Public Member Functions

- [ScaleAndSubsystemStringGrammar \(\)](#)

## Static Public Attributes

- static const std::string [SCALE\\_AND\\_SUBSYSTEM\\_LABEL](#)

## Private Member Functions

- void [initialise \(\)](#)  
*Initialisation function.*
- void [initialiseGrammar \(\)](#)  
*Initialise the grammar.*
- void [initialiseDebugSupport \(\)](#)  
*Initialise debug support.*
- void [assignNamesToRules \(\)](#)  
*Assign names to the rules.*
- void [initialiseRulesDebugging \(\)](#)

*Initialise the debugging of rules.*

- void [initialiseErrorHandlingSupport \(\)](#)

*Initialise the error handling routines.*

## Private Attributes

- `qi::rule< Iterator, std::string(), qi::space_type > scaleAndSubsystemStringRule`

## Static Private Attributes

- static const std::string [SCALE\\_AND\\_SUBSYSTEM\\_STRING\\_PATTERN](#)

### 6.169.1 Detailed Description

```
template<typename Iterator>class multiscale::verification::ScaleAndSubsystemStringGrammar<  
Iterator >
```

The grammar for parsing scale and subsystem string statements.

Definition at line 21 of file ScaleAndSubsystemStringGrammar.hpp.

### 6.169.2 Constructor & Destructor Documentation

```
6.169.2.1 template<typename Iterator > multiscale::verification-  
::ScaleAndSubsystemStringGrammar< Iterator  
>::ScaleAndSubsystemStringGrammar( )
```

Definition at line 23 of file ScaleAndSubsystemStringGrammarDefinition.hpp.

References `multiscale::verification::ScaleAndSubsystemStringGrammar< Iterator >->::initialise()`.

### 6.169.3 Member Function Documentation

```
6.169.3.1 template<typename Iterator > void multiscale::verification::ScaleAnd-  
SubsystemStringGrammar< Iterator >::assignNamesToRules( )  
[private]
```

Assign names to the rules.

Definition at line 61 of file ScaleAndSubsystemStringGrammarDefinition.hpp.

6.169.3.2 **template<typename Iterator > void multiscale::verification::ScaleAndSubsystemStringGrammar< Iterator >::initialise( ) [private]**

Initialisation function.

Definition at line 33 of file ScaleAndSubsystemStringGrammarDefinition.hpp.

Referenced by multiscale::verification::ScaleAndSubsystemStringGrammar< Iterator >::ScaleAndSubsystemStringGrammar().

6.169.3.3 **template<typename Iterator > void multiscale::verification::ScaleAndSubsystemStringGrammar< Iterator >::initialiseDebugSupport( ) [private]**

Initialise debug support.

Definition at line 52 of file ScaleAndSubsystemStringGrammarDefinition.hpp.

6.169.3.4 **template<typename Iterator > void multiscale::verification::ScaleAndSubsystemStringGrammar< Iterator >::initialiseErrorHandlingSupport( ) [private]**

Initialise the error handling routines.

Definition at line 73 of file ScaleAndSubsystemStringGrammarDefinition.hpp.

References multiscale::verification::handleUnexpectedTokenError.

6.169.3.5 **template<typename Iterator > void multiscale::verification::ScaleAndSubsystemStringGrammar< Iterator >::initialiseGrammar( ) [private]**

Initialise the grammar.

Definition at line 41 of file ScaleAndSubsystemStringGrammarDefinition.hpp.

6.169.3.6 **template<typename Iterator > void multiscale::verification::ScaleAndSubsystemStringGrammar< Iterator >::initialiseRulesDebugging( ) [private]**

Initialise the debugging of rules.

Definition at line 67 of file ScaleAndSubsystemStringGrammarDefinition.hpp.

## 6.169.4 Member Data Documentation

---

```
6.169.4.1 template<typename Iterator> const std::string multiscale-
    ::verification::ScaleAndSubsystemStringGrammar< Iterator
    >::SCALE_AND_SUBSYSTEM_LABEL [static]
```

Definition at line 64 of file ScaleAndSubsystemStringGrammar.hpp.

```
6.169.4.2 template<typename Iterator> const std::string multiscale-
    ::verification::ScaleAndSubsystemStringGrammar< Iterator
    >::SCALE_AND_SUBSYSTEM_STRING_PATTERN [static,
    private]
```

Definition at line 59 of file ScaleAndSubsystemStringGrammar.hpp.

```
6.169.4.3 template<typename Iterator> qi::rule<Iterator, std::string(), qi::space_type>
    multiscale::verification::ScaleAndSubsystemStringGrammar< Iterator
    >::scaleAndSubsystemStringRule [private]
```

The rule for parsing a string representing a scale and subsystem

Definition at line 29 of file ScaleAndSubsystemStringGrammar.hpp.

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

- ScaleAndSubsystemStringGrammar.hpp
- ScaleAndSubsystemStringGrammarDefinition.hpp

## 6.170 multiscale::analysis::Silhouette Class Reference

Class for computing the "Silhouette" clustering index.

```
#include <Silhouette.hpp>
```

### Static Public Member Functions

- static double `computeOverallAverageMeasure` (const std::vector< `Cluster` > &clusters)  
*Compute the overall average silhouette measure for the given collection of clusters.*
- static double `computeAverageMeasure` (std::size\_t clusterIndex, const std::vector< `Cluster` > &clusters)  
*Compute the average silhouette measure for the given cluster.*
- static double `computeMeasure` (std::size\_t entityIndex, std::size\_t clusterIndex, const std::vector< `Cluster` > &clusters)  
*Compute the silhouette measure for the given entity.*

## Static Private Member Functions

- static double `computeAverageDissimilarityWithinCluster` (std::size\_t entityIndex, std::size\_t clusterIndex, const std::vector<`Cluster`> &clusters)
 

*Compute the average dissimilarity within cluster to which the entity belongs.*
- static double `computeAverageDissimilarityToOtherClusters` (std::size\_t entityIndex, std::size\_t clusterIndex, const std::vector<`Cluster`> &clusters)
 

*Compute the average dissimilarity of the entity to the other clusters.*
- static double `computeAverageDissimilarityBtwEntityAndCluster` (std::size\_t entityIndex, std::size\_t entityClusterIndex, std::size\_t clusterIndex, const std::vector<`Cluster`> &clusters)
 

*Compute the average dissimilarity between entity and cluster.*
- static void `validateClusterIndex` (std::size\_t clusterIndex, std::size\_t totalNrOfClusters)
 

*Check if the provided cluster index is valid.*
- static void `validateEntityIndex` (std::size\_t entityIndex, std::size\_t totalNrOfEntities)
 

*Check if the provided entity index is valid.*
- static void `validateElementIndex` (std::size\_t elementIndex, std::size\_t totalNrOfElements)
 

*Check if the provided element index is valid.*

### 6.170.1 Detailed Description

Class for computing the "Silhouette" clustering index.

Definition at line 12 of file Silhouette.hpp.

### 6.170.2 Member Function Documentation

6.170.2.1 double `Silhouette::computeAverageDissimilarityBtwEntityAndCluster` (std::size\_t `entityIndex`, std::size\_t `entityClusterIndex`, std::size\_t `clusterIndex`, const std::vector<`Cluster`> & `clusters`) [static, private]

Compute the average dissimilarity between entity and cluster.

#### Parameters

|                                                   |                                                                           |
|---------------------------------------------------|---------------------------------------------------------------------------|
| <code>entityIndex</code>                          | The index of the entity in the cluster for which the distance is computed |
| <code>entity-</code><br><code>ClusterIndex</code> | The index of the cluster to which the entity belongs                      |
| <code>clusterIndex</code>                         | The index of the cluster to which the average distance is computed        |
| <code>clusters</code>                             | Collection of all clusters                                                |

Definition at line 92 of file Silhouette.cpp.

References `multiscale::Geometry2D::distanceBtwPoints()`, and `multiscale::Numeric::division()`.

**6.170.2.2 double Silhouette::computeAverageDissimilarityToOtherClusters ( std::size\_t entityIndex, std::size\_t clusterIndex, const std::vector< Cluster > & clusters ) [static, private]**

Compute the average dissimilarity of the entity to the other clusters.

Compute the average dissimilarity of the entity to the other clusters (i.e. clusters which are different from the cluster to which the entity belongs)

#### Parameters

|                           |                                                                                     |
|---------------------------|-------------------------------------------------------------------------------------|
| <code>entityIndex</code>  | The index of the entity in the cluster for which the silhouette measure is computed |
| <code>clusterIndex</code> | The index of the cluster to which the entity belongs                                |
| <code>clusters</code>     | Collection of all clusters                                                          |

Definition at line 72 of file `Silhouette.cpp`.

**6.170.2.3 double Silhouette::computeAverageDissimilarityWithinCluster ( std::size\_t entityIndex, std::size\_t clusterIndex, const std::vector< Cluster > & clusters ) [static, private]**

Compute the average dissimilarity within cluster to which the entity belongs.

#### Parameters

|                           |                                                                                     |
|---------------------------|-------------------------------------------------------------------------------------|
| <code>entityIndex</code>  | The index of the entity in the cluster for which the silhouette measure is computed |
| <code>clusterIndex</code> | The index of the cluster to which the entity belongs                                |
| <code>clusters</code>     | Collection of all clusters                                                          |

Definition at line 55 of file `Silhouette.cpp`.

References `multiscale::Geometry2D::distanceBtwPoints()`, and `multiscale::Numeric::division()`.

**6.170.2.4 double Silhouette::computeAverageMeasure ( std::size\_t clusterIndex, const std::vector< Cluster > & clusters ) [static]**

Compute the average silhouette measure for the given cluster.

#### Parameters

|                           |                                                                               |
|---------------------------|-------------------------------------------------------------------------------|
| <code>clusterIndex</code> | The index of the cluster for which the average silhouette measure is computed |
| <code>clusters</code>     | Collection of all clusters                                                    |

Definition at line 24 of file Silhouette.cpp.

References multiscale::Numeric::division().

**6.170.2.5 double Silhouette::computeMeasure ( std::size\_t entityIndex, std::size\_t clusterIndex, const std::vector< Cluster > & clusters ) [static]**

Compute the silhouette measure for the given entity.

#### Parameters

|                     |                                                                                     |
|---------------------|-------------------------------------------------------------------------------------|
| <i>entityIndex</i>  | The index of the entity in the cluster for which the silhouette measure is computed |
| <i>clusterIndex</i> | The index of the cluster to which the entity belongs                                |
| <i>clusters</i>     | Collection of all clusters                                                          |

Definition at line 39 of file Silhouette.cpp.

References multiscale::Numeric::division().

**6.170.2.6 double Silhouette::computeOverallAverageMeasure ( const std::vector< Cluster > & clusters ) [static]**

Compute the overall average silhouette measure for the given collection of clusters.

#### Parameters

|                 |                            |
|-----------------|----------------------------|
| <i>clusters</i> | Collection of all clusters |
|-----------------|----------------------------|

Definition at line 13 of file Silhouette.cpp.

References multiscale::Numeric::division().

Referenced by multiscale::analysis::ClusterDetector::computeClusterednessIndex().

**6.170.2.7 void Silhouette::validateClusterIndex ( std::size\_t clusterIndex, std::size\_t totalNrOfClusters ) [static, private]**

Check if the provided cluster index is valid.

The cluster index clusterIndex (0-based indexing) is valid if and only if:  $0 \leq \text{clusterIndex} < \text{total number of clusters}$

#### Parameters

|                          |                              |
|--------------------------|------------------------------|
| <i>clusterIndex</i>      | The index of the cluster     |
| <i>totalNrOfClusters</i> | The total number of clusters |

Definition at line 109 of file Silhouette.cpp.

**6.170.2.8 void Silhouette::validateElementIndex ( std::size\_t elementIndex, std::size\_t totalNrOfElements ) [static, private]**

Check if the provided element index is valid.

The element index elementIndex (0-based indexing) is valid if and only if:  $0 \leq \text{elementIndex} < \text{total number of elements}$

**Parameters**

|                           |                              |
|---------------------------|------------------------------|
| <i>element-Index</i>      | The index of the element     |
| <i>totalNrOf-Elements</i> | The total number of elements |

Definition at line 117 of file Silhouette.cpp.

**6.170.2.9 void Silhouette::validateEntityIndex ( std::size\_t entityIndex, std::size\_t totalNrOfEntities ) [static, private]**

Check if the provided entity index is valid.

The entity index entityIndex (0-based indexing) is valid if and only if:  $0 \leq \text{entityIndex} < \text{total number of entities}$

**Parameters**

|                           |                              |
|---------------------------|------------------------------|
| <i>entityIndex</i>        | The index of the entity      |
| <i>totalNrOf-Entities</i> | The total number of entities |

Definition at line 113 of file Silhouette.cpp.

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

- Silhouette.hpp
- Silhouette.cpp

## **6.171 multiscale::verification::SimilarityMeasureAttribute Class - Reference**

Class for representing a similarity measure attribute.

```
#include <SimilarityMeasureAttribute.hpp>
```

### **Public Attributes**

- [SimilarityMeasureType similarityMeasure](#)

### 6.171.1 Detailed Description

Class for representing a similarity measure attribute.

Definition at line 28 of file SimilarityMeasureAttribute.hpp.

### 6.171.2 Member Data Documentation

#### 6.171.2.1 **SimilarityMeasureType multiscale::verification::SimilarityMeasureAttribute::similarityMeasure**

The similarity measure

Definition at line 32 of file SimilarityMeasureAttribute.hpp.

Referenced by multiscale::verification::LogicPropertyVisitor::operator()().

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

- [SimilarityMeasureAttribute.hpp](#)

## 6.172 multiscale::verification::SimilarityMeasureTypeParser Struct Reference

Symbol table and parser for the similarity measure type.

```
#include <SymbolTables.hpp>
```

### Public Member Functions

- [SimilarityMeasureTypeParser \(\)](#)

### 6.172.1 Detailed Description

Symbol table and parser for the similarity measure type.

Definition at line 156 of file SymbolTables.hpp.

### 6.172.2 Constructor & Destructor Documentation

#### 6.172.2.1 **multiscale::verification::SimilarityMeasureTypeParser::SimilarityMeasureTypeParser( ) [inline]**

Definition at line 159 of file SymbolTables.hpp.

References multiscale::verification::Similar.

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

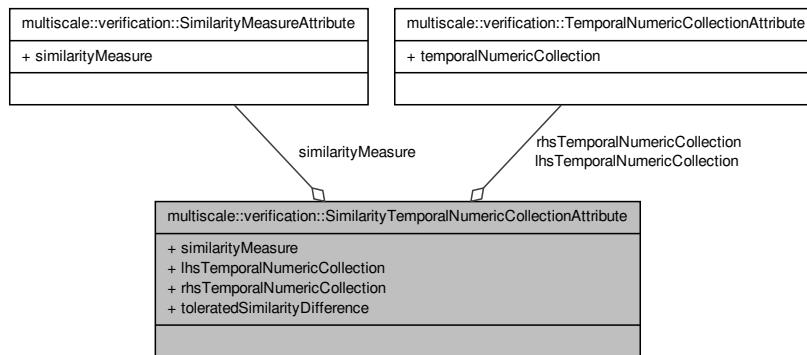
- SymbolTables.hpp

## 6.173 multiscale::verification::SimilarityTemporalNumericCollection-Attribute Class Reference

Class for representing a similarity temporal numeric collection attribute.

```
#include <SimilarityTemporalNumericCollectionAttribute.-  
hpp>
```

Collaboration diagram for multiscale::verification::SimilarityTemporalNumericCollection-Attribute:



### Public Attributes

- `SimilarityMeasureAttribute similarityMeasure`
- `TemporalNumericCollectionAttribute lhsTemporalNumericCollection`
- `TemporalNumericCollectionAttribute rhsTemporalNumericCollection`
- `double toleratedSimilarityDifference`

### 6.173.1 Detailed Description

Class for representing a similarity temporal numeric collection attribute.

Definition at line 16 of file `SimilarityTemporalNumericCollectionAttribute.hpp`.

### 6.173.2 Member Data Documentation

---

**6.173.2.1 TemporalNumericCollectionAttribute multiscale::verification::-  
SimilarityTemporalNumericCollectionAttribute::lhsTemporalNumeric-  
Collection**

The left hand side temporal numeric collection

Definition at line 23 of file SimilarityTemporalNumericCollectionAttribute.hpp.

Referenced by multiscale::verification::LogicPropertyVisitor::operator()().

**6.173.2.2 TemporalNumericCollectionAttribute multiscale::verification::-  
SimilarityTemporalNumericCollectionAttribute::rhsTemporalNumeric-  
Collection**

The right hand side temporal numeric collection

Definition at line 25 of file SimilarityTemporalNumericCollectionAttribute.hpp.

Referenced by multiscale::verification::LogicPropertyVisitor::operator()().

**6.173.2.3 SimilarityMeasureAttribute multiscale::verification::SimilarityTemporal-  
NumericCollectionAttribute::similarityMeasure**

The similarity measure

Definition at line 21 of file SimilarityTemporalNumericCollectionAttribute.hpp.

Referenced by multiscale::verification::LogicPropertyVisitor::operator()().

**6.173.2.4 double multiscale::verification::SimilarityTemporalNumericCollection-  
Attribute::toleratedSimilarityDifference**

The tolerated similarity difference

Definition at line 27 of file SimilarityTemporalNumericCollectionAttribute.hpp.

Referenced by multiscale::verification::LogicPropertyVisitor::operator()().

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

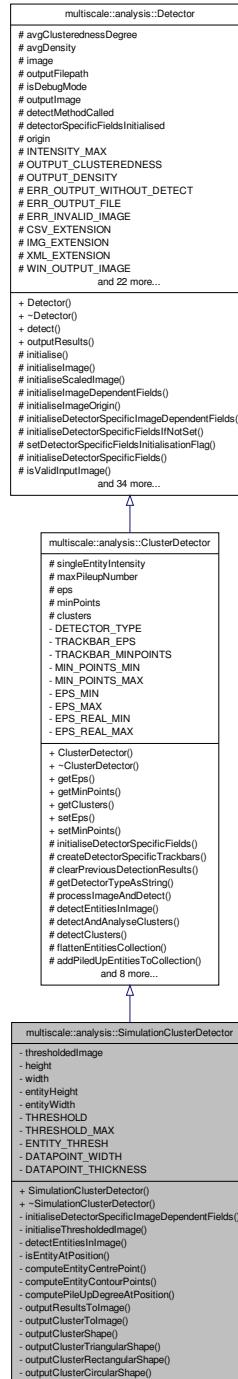
- SimilarityTemporalNumericCollectionAttribute.hpp

**6.174 multiscale::analysis::SimulationClusterDetector Class -  
Reference**

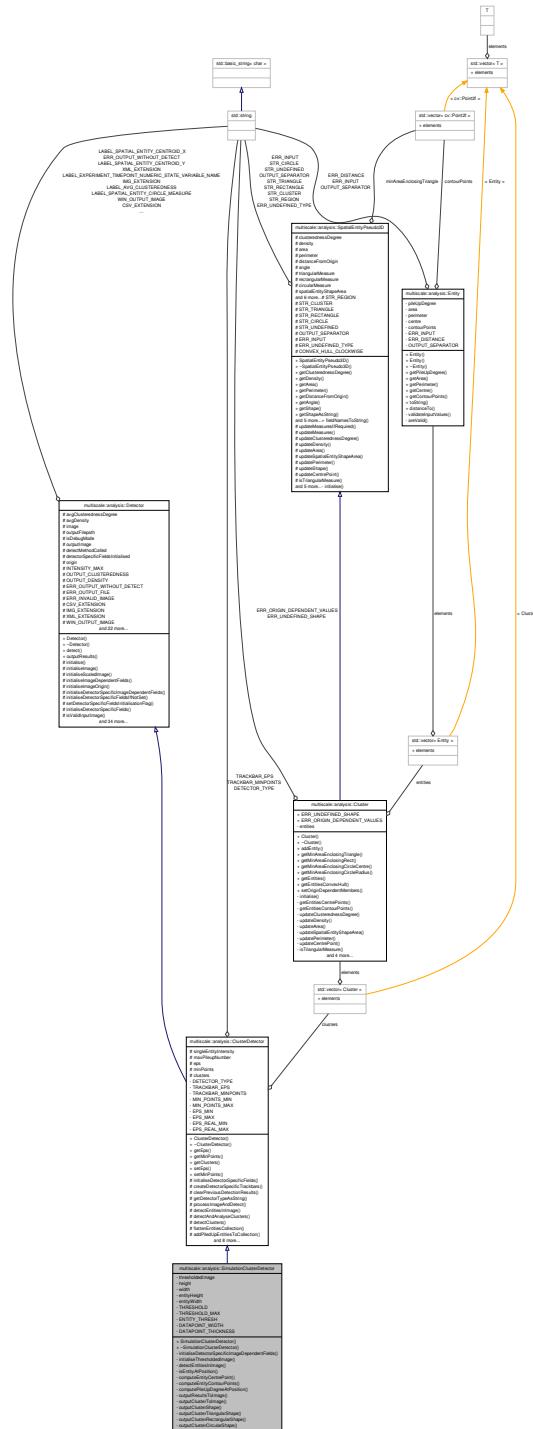
Class for detecting clusters in 2D images obtained from simulations.

```
#include <SimulationClusterDetector.hpp>
```

Inheritance diagram for multiscale::analysis::SimulationClusterDetector:



## Collaboration diagram for multiscale::analysis::SimulationClusterDetector:



## Public Member Functions

- `SimulationClusterDetector` (`unsigned int height`, `unsigned int width`, `unsigned int maxPileupNumber`, `bool isDebugMode=false`)
- `~SimulationClusterDetector ()`

## Private Member Functions

- `void initialiseDetectorSpecificImageDependentFields () override`  
*Initialise the image dependent values.*
- `void initialiseThresholdedImage ()`  
*Initialise the thresholdedImage field.*
- `void detectEntitiesInImage (std::vector< Entity > &entities) override`  
*Detect the entities in the image.*
- `bool isEntityAtPosition (int x, int y)`  
*Check if there is an entity in the image at the given position.*
- `cv::Point2f computeEntityCentrePoint (int x, int y)`  
*Compute the point representing the centre of the entity.*
- `std::vector< cv::Point2f > computeEntityContourPoints (int x, int y)`  
*Compute the points representing the contour of the entity.*
- `double computePileUpDegreeAtPosition (int x, int y)`  
*Compute the pile up degree at the given position.*
- `void outputResultsToImage () override`  
*Display clusters on image.*
- `void outputClusterToImage (Cluster &cluster, cv::Scalar colour, cv::Mat &image)`  
*Display cluster on the image.*
- `void outputClusterShape (Cluster &cluster, cv::Scalar colour, cv::Mat &image)`  
*Draw the best matching shape (triangular, rectangular, circular) of the cluster on the image.*
- `void outputClusterTriangularShape (Cluster &cluster, cv::Scalar colour, cv::Mat &image)`  
*Draw the best matching triangular shape of the cluster on the image.*
- `void outputClusterRectangularShape (Cluster &cluster, cv::Scalar colour, cv::Mat &image)`  
*Draw the best matching rectangular shape of the cluster on the image.*
- `void outputClusterCircularShape (Cluster &cluster, cv::Scalar colour, cv::Mat &image)`  
*Draw the best matching circular shape of the cluster on the image.*

### Private Attributes

- cv::Mat `thresholdedImage`
- unsigned int `height`
- unsigned int `width`
- double `entityHeight`
- double `entityWidth`

### Static Private Attributes

- static const int `THRESHOLD` = 1
- static const int `THRESHOLD_MAX` = 255
- static const int `ENTITY_THRESH` = 200
- static const int `DATAPOINT_WIDTH` = 1
- static const int `DATAPOINT_THICKNESS` = -1

#### 6.174.1 Detailed Description

Class for detecting clusters in 2D images obtained from simulations.

Definition at line 15 of file `SimulationClusterDetector.hpp`.

#### 6.174.2 Constructor & Destructor Documentation

**6.174.2.1 `SimulationClusterDetector::SimulationClusterDetector( unsigned int height, unsigned int width, unsigned int maxPileupNumber, bool isDebugEnabled = false )`**

Definition at line 11 of file `SimulationClusterDetector.cpp`.

References `entityHeight`, `entityWidth`, `height`, and `width`.

**6.174.2.2 `SimulationClusterDetector::~SimulationClusterDetector( )`**

Definition at line 22 of file `SimulationClusterDetector.cpp`.

#### 6.174.3 Member Function Documentation

**6.174.3.1 `cv::Point2f SimulationClusterDetector::computeEntityCentrePoint( int x, int y ) [private]`**

Compute the point representing the centre of the entity.

##### Parameters

|                |               |
|----------------|---------------|
| <code>x</code> | Ox coordinate |
| <code>y</code> | Oy coordinate |

Definition at line 74 of file SimulationClusterDetector.cpp.

References entityHeight, and entityWidth.

Referenced by detectEntitiesInImage().

```
6.174.3.2 std::vector< cv::Point2f > SimulationClusterDetector-
    ::computeEntityContourPoints ( int x, int y )
    [private]
```

Compute the points representing the contour of the entity.

#### Parameters

|   |               |
|---|---------------|
| x | Ox coordinate |
| y | Oy coordinate |

Definition at line 81 of file SimulationClusterDetector.cpp.

References entityHeight, and entityWidth.

Referenced by detectEntitiesInImage().

```
6.174.3.3 double SimulationClusterDetector::computePileUpDegreeAtPosition ( int
    x, int y ) [private]
```

Compute the pile up degree at the given position.

#### Parameters

|   |                        |
|---|------------------------|
| x | Coordinate for Ox axis |
| y | Coordinate for Oy axis |

Definition at line 98 of file SimulationClusterDetector.cpp.

References multiscale::Numeric::division(), entityHeight, entityWidth, multiscale::analysis::Detector::image, multiscale::analysis::ClusterDetector::maxPileupNumber, and multiscale::analysis::ClusterDetector::singleEntityIntensity.

Referenced by detectEntitiesInImage().

```
6.174.3.4 void SimulationClusterDetector::detectEntitiesInImage ( std::vector<
    Entity > & entities ) [override, private, virtual]
```

Detect the entities in the image.

Detect the entities in the image, compute their centre point and degree of pile up

#### Parameters

|          |                                |
|----------|--------------------------------|
| entities | Entities detected in the image |
|----------|--------------------------------|

Implements [multiscale::analysis::ClusterDetector](#).

Definition at line 41 of file [SimulationClusterDetector.cpp](#).

References [computeEntityCentrePoint\(\)](#), [computeEntityContourPoints\(\)](#), [computePileUpDegreeAtPosition\(\)](#), [entityHeight](#), [entityWidth](#), [height](#), [isEntityAtPosition\(\)](#), and [width](#).

```
6.174.3.5 void SimulationClusterDetector::initialiseDetectorSpecific-
    ImageDependentFields( ) [override, private,
    virtual]
```

Initialise the image dependent values.

Implements [multiscale::analysis::Detector](#).

Definition at line 24 of file [SimulationClusterDetector.cpp](#).

References [multiscale::Numeric::division\(\)](#), [entityHeight](#), [entityWidth](#), [height](#), [multiscale-
 ::analysis::Detector::image](#), [initialiseThresholdedImage\(\)](#), and [width](#).

```
6.174.3.6 void SimulationClusterDetector::initialiseThresholdedImage( )
    [private]
```

Initialise the thresholdedImage field.

Definition at line 37 of file [SimulationClusterDetector.cpp](#).

References [multiscale::analysis::Detector::image](#), [THRESHOLD](#), [THRESHOLD\\_MAX](#), and [thresholdedImage](#).

Referenced by [initialiseDetectorSpecificImageDependentFields\(\)](#).

```
6.174.3.7 bool SimulationClusterDetector::isEntityAtPosition( int x, int y )
    [private]
```

Check if there is an entity in the image at the given position.

#### Parameters

|   |                        |
|---|------------------------|
| x | Coordinate for Ox axis |
| y | Coordinate for Oy axis |

Definition at line 66 of file [SimulationClusterDetector.cpp](#).

References [ENTITY\\_THRESH](#), [entityHeight](#), [entityWidth](#), and [thresholdedImage](#).

Referenced by [detectEntitiesInImage\(\)](#).

```
6.174.3.8 void SimulationClusterDetector::outputClusterCircularShape( Cluster &
    cluster, cv::Scalar colour, cv::Mat & image ) [private]
```

Draw the best matching circular shape of the cluster on the image.

**Parameters**

|                |                                                               |
|----------------|---------------------------------------------------------------|
| <i>cluster</i> | <a href="#">Cluster</a>                                       |
| <i>colour</i>  | Colour associated to all entities in the cluster              |
| <i>image</i>   | The image on which to display the cluster related information |

Definition at line 192 of file SimulationClusterDetector.cpp.

References `DATAPoint_WIDTH`, `multiscale::analysis::Cluster::getMinAreaEnclosingCircleCentre()`, and `multiscale::analysis::Cluster::getMinAreaEnclosingCircleRadius()`.

Referenced by `outputClusterShape()`.

#### 6.174.3.9 void [SimulationClusterDetector::outputClusterRectangularShape](#) (`Cluster & cluster, cv::Scalar colour, cv::Mat & image`) [private]

Draw the best matching rectangular shape of the cluster on the image.

**Parameters**

|                |                                                               |
|----------------|---------------------------------------------------------------|
| <i>cluster</i> | <a href="#">Cluster</a>                                       |
| <i>colour</i>  | Colour associated to all entities in the cluster              |
| <i>image</i>   | The image on which to display the cluster related information |

Definition at line 182 of file SimulationClusterDetector.cpp.

References `DATAPoint_WIDTH`, and `multiscale::analysis::Cluster::getMinAreaEnclosingRect()`.

Referenced by `outputClusterShape()`.

#### 6.174.3.10 void [SimulationClusterDetector::outputClusterShape](#) (`Cluster & cluster, cv::Scalar colour, cv::Mat & image`) [private]

Draw the best matching shape (triangular, rectangular, circular) of the cluster on the image.

**Parameters**

|                |                                                               |
|----------------|---------------------------------------------------------------|
| <i>cluster</i> | <a href="#">Cluster</a>                                       |
| <i>colour</i>  | Colour associated to all entities in the cluster              |
| <i>image</i>   | The image on which to display the cluster related information |

Definition at line 150 of file SimulationClusterDetector.cpp.

References `multiscale::analysis::Cluster::ERR_UNDEFINED_SHAPE`, `multiscale::analysis::SpatialEntityPseudo3D::getShape()`, `outputClusterCircularShape()`, `outputClusterRectangularShape()`, and `outputClusterTriangularShape()`.

Referenced by `outputClusterToImage()`.

---

**6.174.3.11 void SimulationClusterDetector::outputClusterToImage ( Cluster & cluster, cv::Scalar colour, cv::Mat & image ) [private]**

Display cluster on the image.

**Parameters**

|                |                                                               |
|----------------|---------------------------------------------------------------|
| <i>cluster</i> | Cluster                                                       |
| <i>colour</i>  | Colour associated to all entities in the cluster              |
| <i>image</i>   | The image on which to display the cluster related information |

Definition at line 140 of file SimulationClusterDetector.cpp.

References DATAPOINT\_THICKNESS, DATAPOINT\_WIDTH, multiscale::analysis::Cluster::getEntities(), and outputClusterShape().

Referenced by outputResultsToImage().

**6.174.3.12 void SimulationClusterDetector::outputClusterTriangularShape ( Cluster & cluster, cv::Scalar colour, cv::Mat & image ) [private]**

Draw the best matching triangular shape of the cluster on the image.

**Parameters**

|                |                                                               |
|----------------|---------------------------------------------------------------|
| <i>cluster</i> | Cluster                                                       |
| <i>colour</i>  | Colour associated to all entities in the cluster              |
| <i>image</i>   | The image on which to display the cluster related information |

Definition at line 172 of file SimulationClusterDetector.cpp.

References DATAPOINT\_WIDTH, and multiscale::analysis::Cluster::getMinAreaEnclosingTriangle().

Referenced by outputClusterShape().

**6.174.3.13 void SimulationClusterDetector::outputResultsToImage ( ) [override, private, virtual]**

Dsiaply clusters on image.

Implements multiscale::analysis::Detector.

Definition at line 121 of file SimulationClusterDetector.cpp.

References multiscale::analysis::ClusterDetector::clusters, multiscale::RGBColourGenerator::generate(), multiscale::analysis::Detector::image, outputClusterToImage(), and multiscale::analysis::Detector::outputImage.

## 6.174.4 Member Data Documentation

6.174.4.1 **const int SimulationClusterDetector::DATAPOINT\_THICKNESS = -1**  
[static, private]

Definition at line 130 of file SimulationClusterDetector.hpp.

Referenced by outputClusterToImage().

6.174.4.2 **const int SimulationClusterDetector::DATAPOINT\_WIDTH = 1**  
[static, private]

Definition at line 129 of file SimulationClusterDetector.hpp.

Referenced by outputClusterCircularShape(), outputClusterRectangularShape(), outputClusterToImage(), and outputClusterTriangularShape().

6.174.4.3 **const int SimulationClusterDetector::ENTITY\_THRESH = 200** [static, private]

Definition at line 127 of file SimulationClusterDetector.hpp.

Referenced by isEntityAtPosition().

6.174.4.4 **double multiscale::analysis::SimulationClusterDetector::entityHeight**  
[private]

Height of an entity

Definition at line 24 of file SimulationClusterDetector.hpp.

Referenced by computeEntityCentrePoint(), computeEntityContourPoints(), computePileUpDegreeAtPosition(), detectEntitiesInImage(), initialiseDetectorSpecificImageDependentFields(), isEntityAtPosition(), and SimulationClusterDetector().

6.174.4.5 **double multiscale::analysis::SimulationClusterDetector::entityWidth**  
[private]

Width of an entity

Definition at line 25 of file SimulationClusterDetector.hpp.

Referenced by computeEntityCentrePoint(), computeEntityContourPoints(), computePileUpDegreeAtPosition(), detectEntitiesInImage(), initialiseDetectorSpecificImageDependentFields(), isEntityAtPosition(), and SimulationClusterDetector().

6.174.4.6 **unsigned int multiscale::analysis::SimulationClusterDetector::height**  
[private]

Height of the grid used in the simulation

Definition at line 21 of file SimulationClusterDetector.hpp.

Referenced by detectEntitiesInImage(), initialiseDetectorSpecificImageDependentFields(), and SimulationClusterDetector().

**6.174.4.7 const int SimulationClusterDetector::THRESHOLD = 1 [static, private]**

Definition at line 124 of file SimulationClusterDetector.hpp.

Referenced by initialiseThresholdedImage().

**6.174.4.8 const int SimulationClusterDetector::THRESHOLD\_MAX = 255 [static, private]**

Definition at line 125 of file SimulationClusterDetector.hpp.

Referenced by initialiseThresholdedImage().

**6.174.4.9 cv::Mat multiscale::analysis::SimulationClusterDetector::thresholdedImage [private]**

Thresholded version of the image

Definition at line 19 of file SimulationClusterDetector.hpp.

Referenced by initialiseThresholdedImage(), and isEntityAtPosition().

**6.174.4.10 unsigned int multiscale::analysis::SimulationClusterDetector::width [private]**

Width of the grid used in the simulation

Definition at line 22 of file SimulationClusterDetector.hpp.

Referenced by detectEntitiesInImage(), initialiseDetectorSpecificImageDependentFields(), and SimulationClusterDetector().

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

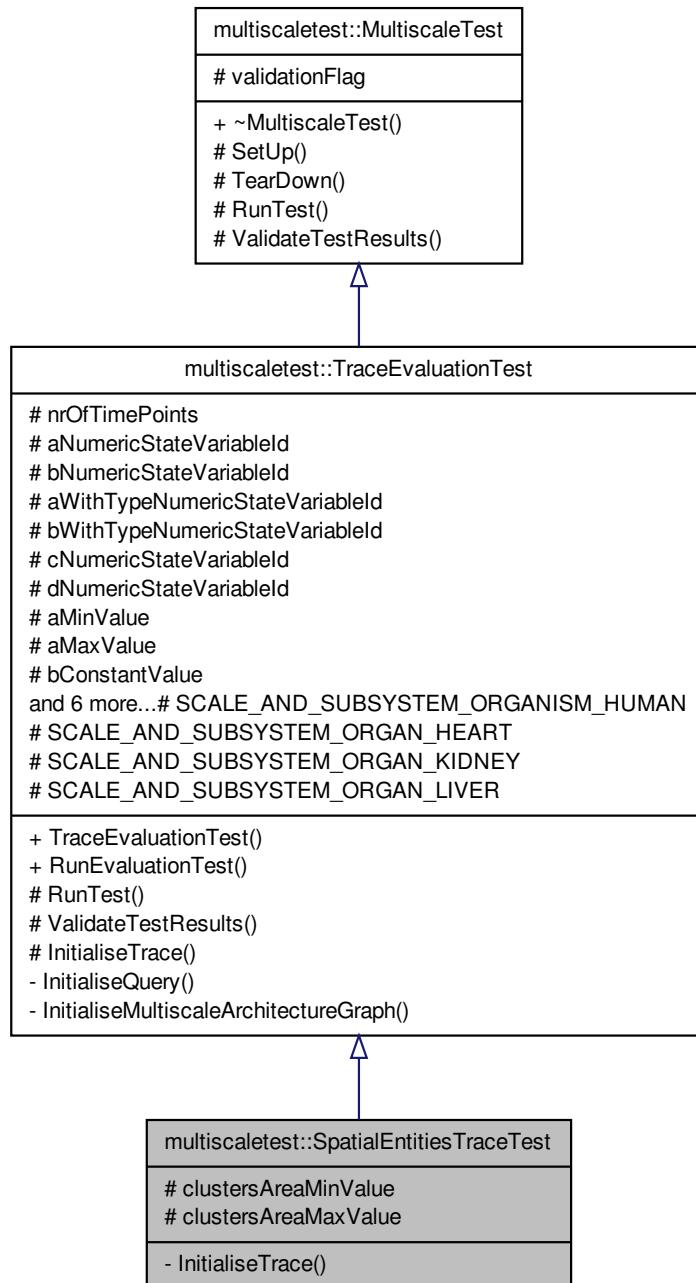
- SimulationClusterDetector.hpp
- SimulationClusterDetector.cpp

## 6.175 multiscaletest::SpatialEntitiesTraceTest Class Reference

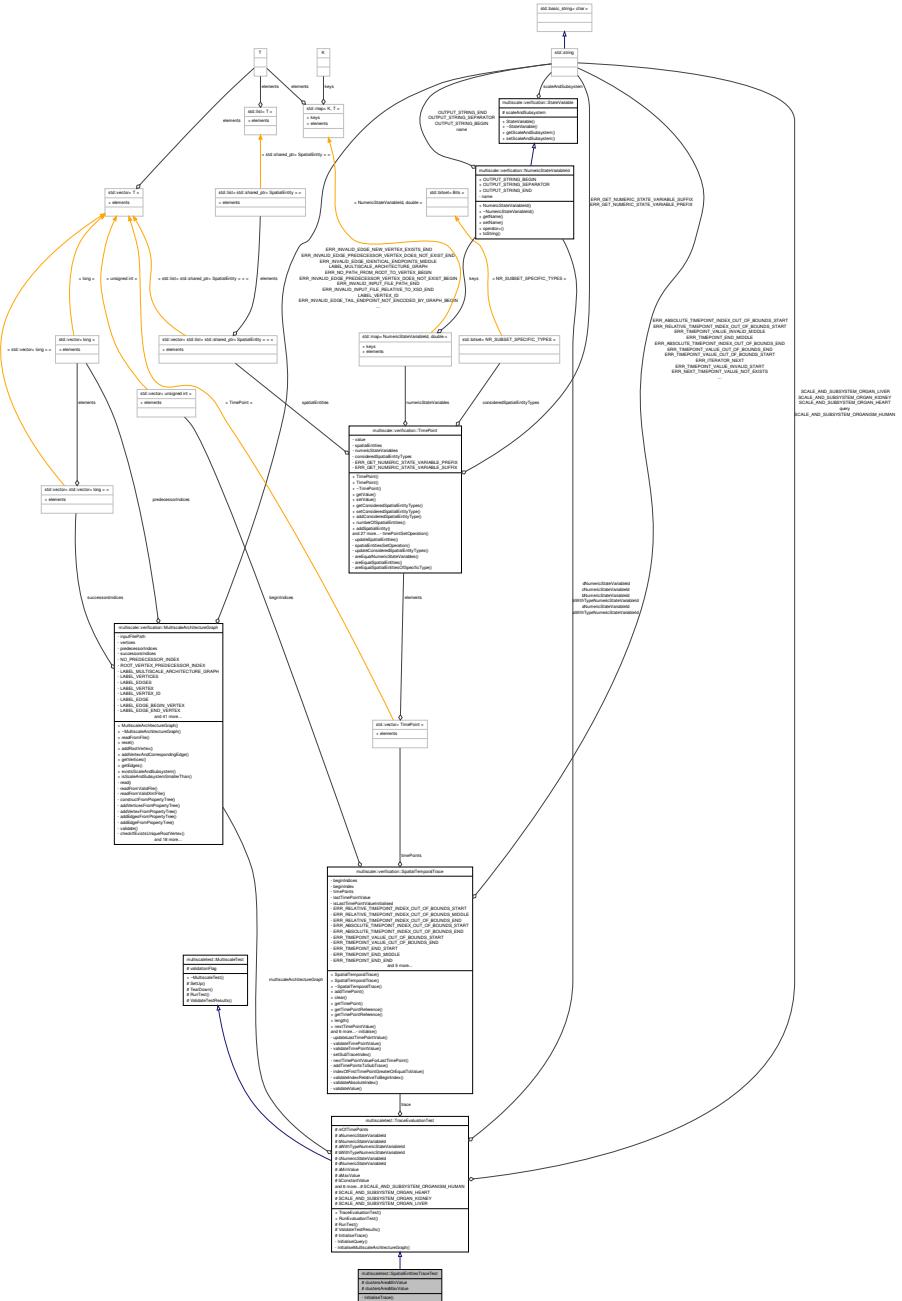
Class for testing evaluation of spatial entities-only traces.

```
#include <SpatialEntitiesTraceTest.hpp>
```

Inheritance diagram for multiscaletest::SpatialEntitiesTraceTest:



## Collaboration diagram for multiscaletest::SpatialEntitiesTraceTest:



## Protected Attributes

- double `clustersAreaMinValue`

- double [clustersArea.MaxValue](#)

### Private Member Functions

- virtual void [InitialiseTrace \(\) override](#)  
*Initialise the trace.*

#### 6.175.1 Detailed Description

Class for testing evaluation of spatial entities-only traces.

Definition at line 25 of file SpatialEntitiesTraceTest.hpp.

#### 6.175.2 Member Function Documentation

##### 6.175.2.1 void multiscaletest::SpatialEntitiesTraceTest::InitialiseTrace ( ) [override, private, virtual]

Initialise the trace.

Implements [multiscaletest::TraceEvaluationTest](#).

Definition at line 39 of file SpatialEntitiesTraceTest.hpp.

#### 6.175.3 Member Data Documentation

##### 6.175.3.1 double multiscaletest::SpatialEntitiesTraceTest::clustersArea.MaxValue [protected]

The maximum area value for the cluster spatial entity type

Definition at line 30 of file SpatialEntitiesTraceTest.hpp.

##### 6.175.3.2 double multiscaletest::SpatialEntitiesTraceTest::clustersArea.MinValue [protected]

The minimum area value for the cluster spatial entity type

Definition at line 29 of file SpatialEntitiesTraceTest.hpp.

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

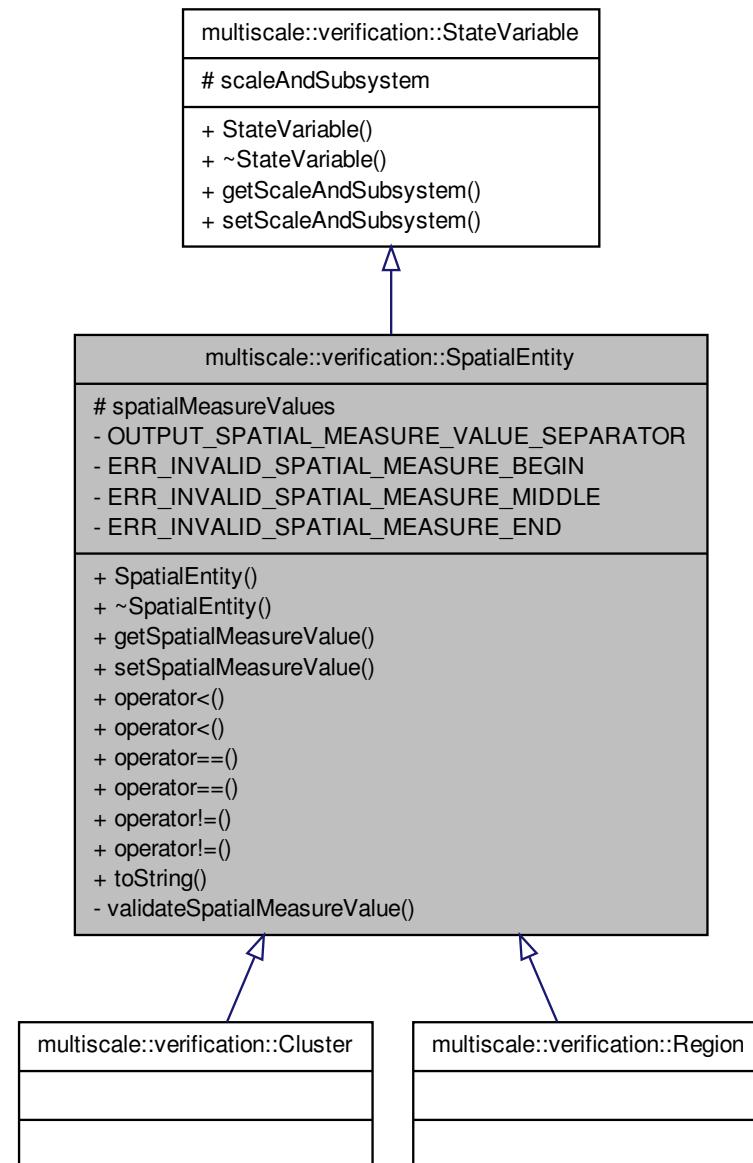
- SpatialEntitiesTraceTest.hpp

## 6.176 multiscale::verification::SpatialEntity Class Reference

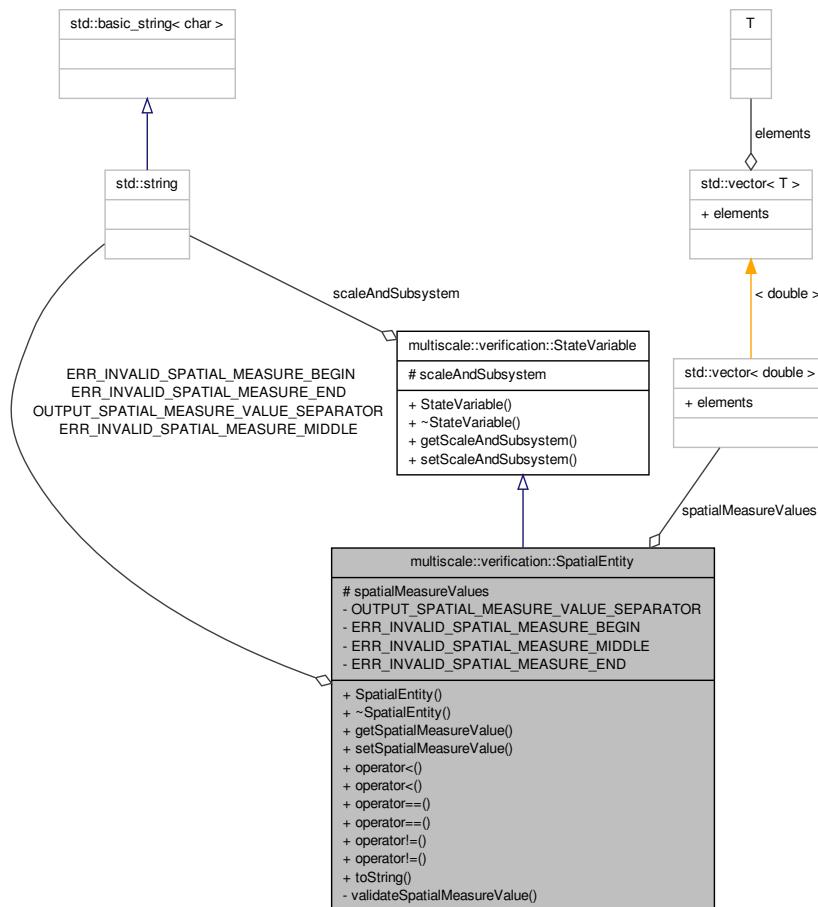
Class for representing a spatial entity.

```
#include <SpatialEntity.hpp>
```

Inheritance diagram for multiscale::verification::SpatialEntity:



Collaboration diagram for multiscale::verification::SpatialEntity:



## Public Member Functions

- `SpatialEntity ()`
- `~SpatialEntity ()`
- `double getSpatialMeasureValue (const SpatialMeasureType &spatialMeasureType) const`

*Get the value of the given spatial measure.*
- `void setSpatialMeasureValue (const SpatialMeasureType &spatialMeasureType, double spatialMeasureValue)`

*Set the value of the given spatial measure.*
- `bool operator< (const SpatialEntity &rhsSpatialEntity)`

*Overload the "<" operator for spatial entities.*

- bool `operator<` (const `SpatialEntity` &rhsSpatialEntity) const  
*Overload the "<" operator for spatial entities.*
- bool `operator==` (const `SpatialEntity` &rhsSpatialEntity)  
*Overload the "==" operator for spatial entities.*
- bool `operator==` (const `SpatialEntity` &rhsSpatialEntity) const  
*Overload the "==" operator for spatial entities.*
- bool `operator!=` (const `SpatialEntity` &rhsSpatialEntity)  
*Overload the "!=" operator for spatial entities.*
- bool `operator!=` (const `SpatialEntity` &rhsSpatialEntity) const  
*Overload the "!=" operator for spatial entities.*
- std::string `toString` () const  
*Return a string representation of the spatial entity contents.*

### Protected Attributes

- std::vector< double > `spatialMeasureValues`

### Private Member Functions

- void `validateSpatialMeasureValue` (double spatialMeasureValue, const `SpatialMeasureType` &spatialMeasureType)  
*Check if the provided value is valid considering the given spatial measure.*

### Static Private Attributes

- static const std::string `OUTPUT_SPATIAL_MEASURE_VALUE_SEPARATOR` = "," "
- static const std::string `ERR_INVALID_SPATIAL_MEASURE_BEGIN` = "The provided spatial measure value ("
- static const std::string `ERR_INVALID_SPATIAL_MEASURE_MIDDLE` = ") is invalid for the given spatial measure type ("
- static const std::string `ERR_INVALID_SPATIAL_MEASURE_END` = "). Please change."

#### 6.176.1 Detailed Description

Class for representing a spatial entity.

Definition at line 18 of file `SpatialEntity.hpp`.

#### 6.176.2 Constructor & Destructor Documentation

##### 6.176.2.1 `SpatialEntity::SpatialEntity( )`

Definition at line 11 of file `SpatialEntity.cpp`.

## 6.176.2.2 SpatialEntity::~SpatialEntity( )

Definition at line 16 of file SpatialEntity.cpp.

## 6.176.3 Member Function Documentation

## 6.176.3.1 double SpatialEntity::getSpatialMeasureValue( const SpatialMeasureType &amp; spatialMeasureType ) const

Get the value of the given spatial measure.

## Parameters

|                                       |                                                     |
|---------------------------------------|-----------------------------------------------------|
| <i>spatial-<br/>Measure-<br/>Type</i> | The spatial measure for which the value is returned |
|---------------------------------------|-----------------------------------------------------|

Definition at line 18 of file SpatialEntity.cpp.

References multiscale::verification::spatialmeasure::computeSpatialMeasureTypeIndex().

Referenced by multiscale::verification::SpatialMeasureEvaluator::evaluate().

## 6.176.3.2 bool SpatialEntity::operator!= ( const SpatialEntity &amp; rhsSpatialEntity )

Overload the "!=" operator for spatial entities.

In this implementation spatial entity se1 is different from spatial entity se2 (se1 != se2) if at least one of the fields in se1 != the corresponding field in se2

## Parameters

|                               |                                                                                 |
|-------------------------------|---------------------------------------------------------------------------------|
| <i>rhsSpatial-<br/>Entity</i> | The spatial entity lying on the right hand side of the comparison opera-<br>tor |
|-------------------------------|---------------------------------------------------------------------------------|

Definition at line 83 of file SpatialEntity.cpp.

## 6.176.3.3 bool SpatialEntity::operator!= ( const SpatialEntity &amp; rhsSpatialEntity ) const

Overload the "!=" operator for spatial entities.

In this implementation spatial entity se1 is different from spatial entity se2 (se1 != se2) if at least one of the fields in se1 != the corresponding field in se2

## Parameters

|                               |                                                                                 |
|-------------------------------|---------------------------------------------------------------------------------|
| <i>rhsSpatial-<br/>Entity</i> | The spatial entity lying on the right hand side of the comparison opera-<br>tor |
|-------------------------------|---------------------------------------------------------------------------------|

Definition at line 90 of file SpatialEntity.cpp.

#### 6.176.3.4 bool SpatialEntity::operator< ( const SpatialEntity & *rhsSpatialEntity* )

Overload the "<" operator for spatial entities.

In this implementation spatial entity se1 is smaller than spatial entity se2 (se1 < se2) if at least one of the fields in se1 < the corresponding field in se2

**Parameters**

|                               |                                                                                 |
|-------------------------------|---------------------------------------------------------------------------------|
| <i>rhsSpatial-<br/>Entity</i> | The spatial entity lying on the right hand side of the comparison opera-<br>tor |
|-------------------------------|---------------------------------------------------------------------------------|

Definition at line 33 of file SpatialEntity.cpp.

#### 6.176.3.5 bool SpatialEntity::operator< ( const SpatialEntity & *rhsSpatialEntity* ) const

Overload the "<" operator for spatial entities.

In this implementation spatial entity se1 is smaller than spatial entity se2 (se1 < se2) if at least one of the fields in se1 < the corresponding field in se2

**Parameters**

|                               |                                                                                 |
|-------------------------------|---------------------------------------------------------------------------------|
| <i>rhsSpatial-<br/>Entity</i> | The spatial entity lying on the right hand side of the comparison opera-<br>tor |
|-------------------------------|---------------------------------------------------------------------------------|

Definition at line 40 of file SpatialEntity.cpp.

References multiscale::verification::StateVariable::scaleAndSubsystem, and spatialMeasureValues.

#### 6.176.3.6 bool SpatialEntity::operator== ( const SpatialEntity & *rhsSpatialEntity* )

Overload the "==" operator for spatial entities.

In this implementation spatial entity se1 is equal to spatial entity se2 (se1 == se2) if all the fields in se1 == the corresponding fields in se2

**Parameters**

|                               |                                                                                 |
|-------------------------------|---------------------------------------------------------------------------------|
| <i>rhsSpatial-<br/>Entity</i> | The spatial entity lying on the right hand side of the comparison opera-<br>tor |
|-------------------------------|---------------------------------------------------------------------------------|

Definition at line 56 of file SpatialEntity.cpp.

**6.176.3.7 bool SpatialEntity::operator== ( const SpatialEntity & *rhsSpatialEntity* ) const**

Overload the "==" operator for spatial entities.

In this implementation spatial entity se1 is equal to spatial entity se2 (*se1 == se2*) if all the fields in se1 == the corresponding fields in se2

**Parameters**

|                         |                                                                            |
|-------------------------|----------------------------------------------------------------------------|
| <i>rhsSpatialEntity</i> | The spatial entity lying on the right hand side of the comparison operator |
|-------------------------|----------------------------------------------------------------------------|

Definition at line 63 of file SpatialEntity.cpp.

References multiscale::Numeric::almostEqual(), multiscale::verification::StateVariable::scaleAndSubsystem, and spatialMeasureValues.

**6.176.3.8 void SpatialEntity::setSpatialMeasureValue ( const SpatialMeasureType & *spatialMeasureType*, double *spatialMeasureValue* )**

Set the value of the given spatial measure.

**Parameters**

|                            |                                                |
|----------------------------|------------------------------------------------|
| <i>spatialMeasureType</i>  | The spatial measure for which the value is set |
| <i>spatialMeasureValue</i> | The new spatial measure value                  |

Definition at line 24 of file SpatialEntity.cpp.

References multiscale::verification::spatialmeasure::computeSpatialMeasureTypeIndex().

**6.176.3.9 std::string SpatialEntity::toString ( ) const**

Return a string representation of the spatial entity contents.

Definition at line 97 of file SpatialEntity.cpp.

**6.176.3.10 void SpatialEntity::validateSpatialMeasureValue ( double *spatialMeasureValue*, const SpatialMeasureType & *spatialMeasureType* ) [private]**

Check if the provided value is valid considering the given spatial measure.

**Parameters**

|                                        |                                                |
|----------------------------------------|------------------------------------------------|
| <i>spatial-<br/>Measure-<br/>Value</i> | The new spatial measure value                  |
| <i>spatial-<br/>Measure-<br/>Type</i>  | The spatial measure for which the value is set |

Definition at line 109 of file SpatialEntity.cpp.

#### 6.176.4 Member Data Documentation

6.176.4.1 `const std::string SpatialEntity::ERR_INVALID_SPATIAL_MEASURE_BEGIN = "The provided spatial measure value (" [static, private]`

Definition at line 111 of file SpatialEntity.hpp.

6.176.4.2 `const std::string SpatialEntity::ERR_INVALID_SPATIAL_MEASURE_END = "). Please change." [static, private]`

Definition at line 113 of file SpatialEntity.hpp.

6.176.4.3 `const std::string SpatialEntity::ERR_INVALID_SPATIAL_MEASURE_MIDDLE = ") is invalid for the given spatial measure type (" [static, private]`

Definition at line 112 of file SpatialEntity.hpp.

6.176.4.4 `const std::string SpatialEntity::OUTPUT_SPATIAL_MEASURE_VALUE_SEPARATOR = "," [static, private]`

Definition at line 109 of file SpatialEntity.hpp.

6.176.4.5 `std::vector<double> multiscale::verification::SpatialEntity::spatialMeasureValues [protected]`

The vector of spatial measures' values. The i-th spatial measure value in the vector corresponds to the i-th SpatialMeasureType enumeration value

Definition at line 22 of file SpatialEntity.hpp.

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

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

- SpatialEntity.hpp

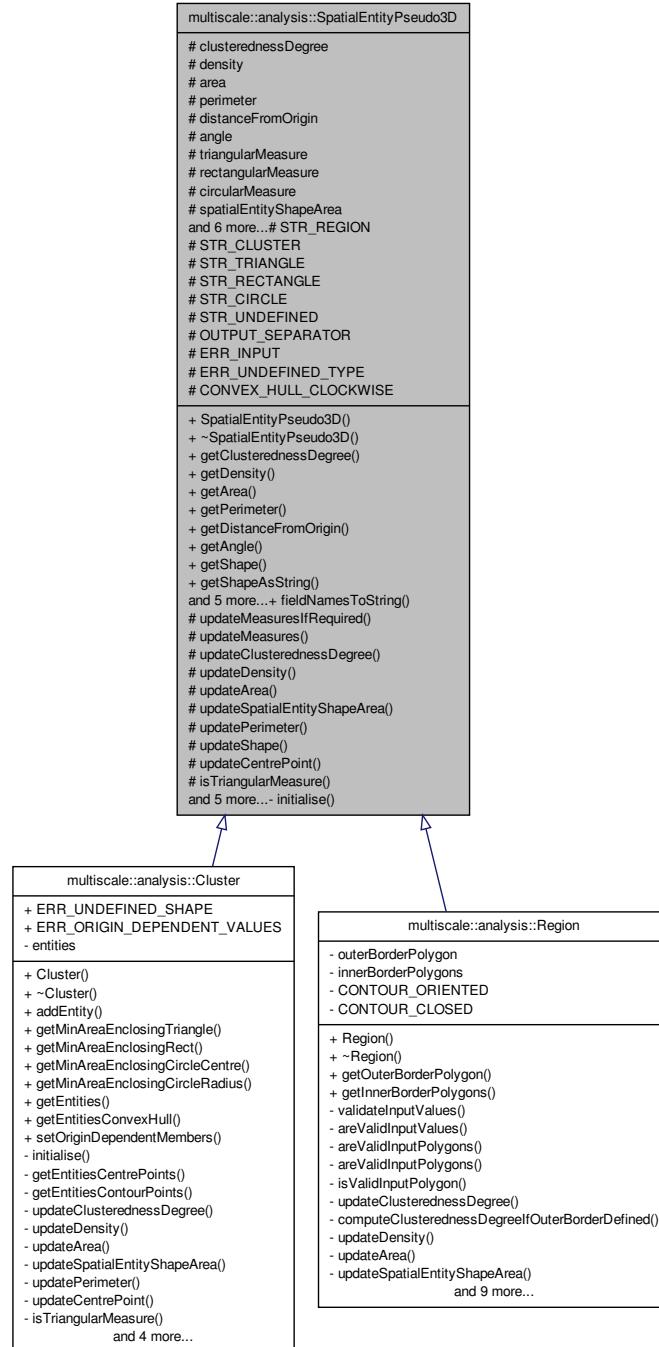
- SpatialEntity.cpp

## 6.177 multiscale::analysis::SpatialEntityPseudo3D Class Reference

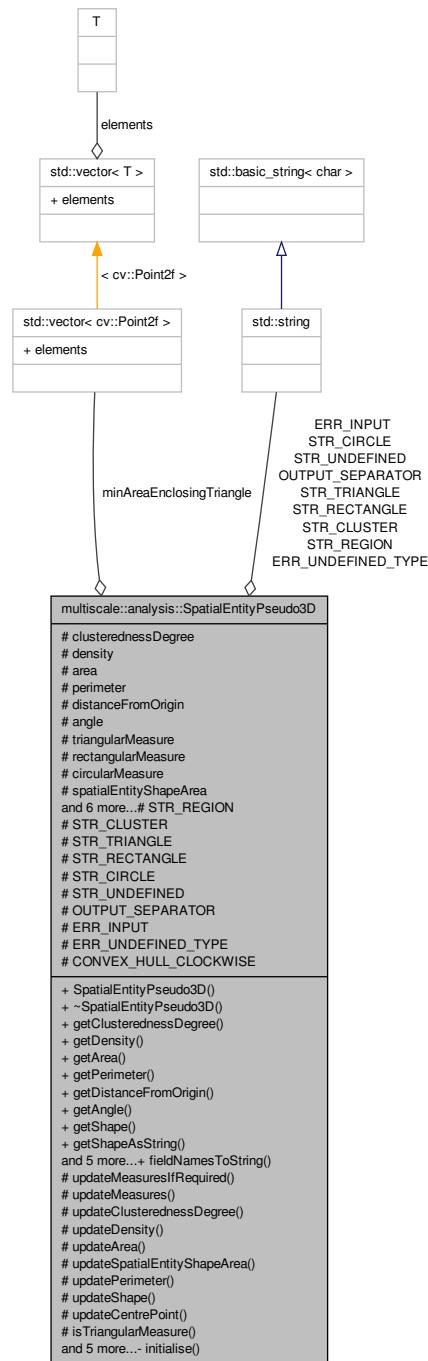
Class for representing a pseudo-3D (explicit 2D + implicit height) object.

```
#include <SpatialEntityPseudo3D.hpp>
```

Inheritance diagram for multiscale::analysis::SpatialEntityPseudo3D:



Collaboration diagram for multiscale::analysis::SpatialEntityPseudo3D:



## Public Member Functions

- `SpatialEntityPseudo3D ()`  
*Get the clusteredness degree.*
- `virtual ~SpatialEntityPseudo3D ()`
- `double getClusterednessDegree ()`  
*Get the clusteredness degree.*
- `double getDensity ()`  
*Get the density.*
- `double getArea ()`  
*Get the area.*
- `double getPerimeter ()`  
*Get the perimeter.*
- `double getDistanceFromOrigin ()`  
*Get the distance from the origin.*
- `double getAngle ()`  
*Get the angle.*
- `Shape2D getShape ()`  
*Get the shape best fitting the spatial collection.*
- `std::string getShapeAsString ()`  
*Get the shape best fitting the spatial collection as a string.*
- `double getTriangularMeasure ()`  
*Get the measure indicating how much the shape of the contour resembles a triangle.*
- `double getRectangularMeasure ()`  
*Get the measure indicating how much the shape of the contour resembles a rectangle.*
- `double getCircularMeasure ()`  
*Get the measure indicating how much the shape of the contour resembles a cv::circle.*
- `cv::Point2f getCentre ()`  
*Get the point defining the centre of the entity.*
- `std::string toString ()`  
*Get the string representation of all field values.*
- `std::string typeAsString ()`  
*Return the type of the pseudo 3D spatial entity as a string.*

## Static Public Member Functions

- `static std::string fieldNamesToString ()`  
*Get a string representation of all the field names printed in the "toString" method.*

### Protected Member Functions

- void [updateMeasuresIfRequired \(\)](#)  
*Update the values of all measures if required.*
- void [updateMeasures \(\)](#)  
*Update the values of all measures.*
- virtual void [updateClusterednessDegree \(\)=0](#)  
*Update the value of the clusteredness degree.*
- virtual void [updateDensity \(\)=0](#)  
*Update the value of the density.*
- virtual void [updateArea \(\)=0](#)  
*Update the value of the area.*
- virtual void [updateSpatialEntityShapeArea \(\)=0](#)  
*Update the value of the spatial entity shape area.*
- virtual void [updatePerimeter \(\)=0](#)  
*Update the value of the perimeter.*
- void [updateShape \(\)](#)  
*Update the shape of the cluster.*
- virtual void [updateCentrePoint \(\)=0](#)  
*Update the point defining the centre of the cluster.*
- virtual double [isTriangularMeasure \(\)=0](#)  
*Get the measure that the cluster has a triangular shape.*
- virtual double [isRectangularMeasure \(\)=0](#)  
*Get the measure that the cluster has a rectangular shape.*
- virtual double [isCircularMeasure \(\)=0](#)  
*Get the measure that the cluster has a circular shape.*
- double [normalisedShapeMeasure \(double shapeArea\)](#)  
*Get the normalised shape measure ([0, 1]) that the spatial entity has a particular shape.*
- std::string [shapeAsString \(\)](#)  
*Return the shape of the cluster as a string.*
- std::string [fieldValuesToString \(\)](#)  
*Return the values of the fields as a string.*
- virtual [SpatialEntityPseudo3DType type \(\)=0](#)  
*Return the type of the pseudo 3D spatial entity.*

### Protected Attributes

- double [clusterednessDegree](#)
- double [density](#)
- double [area](#)
- double [perimeter](#)
- double [distanceFromOrigin](#)
- double [angle](#)

- double `triangularMeasure`
- double `rectangularMeasure`
- double `circularMeasure`
- double `spatialEntityShapeArea`
- `Shape2D shape`
- `cv::Point2f centre`
- bool `updateFlag`
- std::vector< `cv::Point2f > minAreaEnclosingTriangle`
- `cv::RotatedRect minAreaEnclosingRect`
- `cv::Point2f minAreaEnclosingCircleCentre`
- float `minAreaEnclosingCircleRadius`

### Static Protected Attributes

- static const std::string `STR_REGION` = "region"
- static const std::string `STR_CLUSTER` = "cluster"
- static const std::string `STR_TRIANGLE` = "triangular"
- static const std::string `STR_RECTANGLE` = "rectangular"
- static const std::string `STR_CIRCLE` = "circular"
- static const std::string `STR_UNDEFINED` = "undefined"
- static const std::string `OUTPUT_SEPARATOR` = ","
- static const std::string `ERR_INPUT` = "Invalid input parameters were provided to the constructor."
- static const std::string `ERR_UNDEFINED_TYPE` = "Pseudo 3D spatial entity of undefined type encountered."
- static const bool `CONVEX_HULL_CLOCKWISE` = true

### Private Member Functions

- void `initialise ()`  
*Initialisation function for the class.*

#### 6.177.1 Detailed Description

Class for representing a pseudo-3D (explicit 2D + implicit height) object.

Definition at line 15 of file SpatialEntityPseudo3D.hpp.

#### 6.177.2 Constructor & Destructor Documentation

##### 6.177.2.1 SpatialEntityPseudo3D::SpatialEntityPseudo3D ( )

Definition at line 8 of file SpatialEntityPseudo3D.cpp.

References `initialise()`.

**6.177.2.2 SpatialEntityPseudo3D::~SpatialEntityPseudo3D( ) [virtual]**

Definition at line 12 of file SpatialEntityPseudo3D.cpp.

**6.177.3 Member Function Documentation****6.177.3.1 std::string SpatialEntityPseudo3D::fieldNamesToString( ) [static]**

Get a string representation of all the field names printed in the "toString" method.

Definition at line 78 of file SpatialEntityPseudo3D.cpp.

Referenced by multiscale::analysis::Detector::outputResultsToCsvFile().

**6.177.3.2 std::string SpatialEntityPseudo3D::fieldValuesToString( ) [protected]**

Return the values of the fields as a string.

Definition at line 184 of file SpatialEntityPseudo3D.cpp.

References angle, area, centre, circularMeasure, clusterednessDegree, density, distanceFromOrigin, OUTPUT\_SEPARATOR, perimeter, rectangularMeasure, shapeAsString(), and triangularMeasure.

Referenced by toString().

**6.177.3.3 double SpatialEntityPseudo3D::getAngle( )**

Get the angle.

Definition at line 44 of file SpatialEntityPseudo3D.cpp.

References angle, and updateMeasuresIfRequired().

Referenced by multiscale::analysis::Detector::addSpatialEntityPropertiesToTree().

**6.177.3.4 double SpatialEntityPseudo3D::getArea( )**

Get the area.

Definition at line 26 of file SpatialEntityPseudo3D.cpp.

References area, and updateMeasuresIfRequired().

Referenced by multiscale::analysis::Detector::addSpatialEntityPropertiesToTree().

**6.177.3.5 cv::Point2f SpatialEntityPseudo3D::getCentre( )**

Get the point defining the centre of the entity.

Definition at line 72 of file SpatialEntityPseudo3D.cpp.

References centre, and updateMeasuresIfRequired().

Referenced by multiscale::analysis::Detector::addSpatialEntityPropertiesToTree().

#### 6.177.3.6 double SpatialEntityPseudo3D::getCircularMeasure( )

Get the measure indicating how much the shape of the contour resembles a cv::circle.

Definition at line 68 of file SpatialEntityPseudo3D.cpp.

References circularMeasure.

Referenced by multiscale::analysis::Detector::addSpatialEntityPropertiesToTree().

#### 6.177.3.7 double SpatialEntityPseudo3D::getClusterednessDegree( )

Get the clusteredness degree.

Definition at line 14 of file SpatialEntityPseudo3D.cpp.

References clusterednessDegree, and updateMeasuresIfRequired().

Referenced by multiscale::analysis::Detector::addSpatialEntityPropertiesToTree().

#### 6.177.3.8 double SpatialEntityPseudo3D::getDensity( )

Get the density.

Definition at line 20 of file SpatialEntityPseudo3D.cpp.

References density, and updateMeasuresIfRequired().

Referenced by multiscale::analysis::Detector::addSpatialEntityPropertiesToTree().

#### 6.177.3.9 double SpatialEntityPseudo3D::getDistanceFromOrigin( )

Get the distance from the origin.

Definition at line 38 of file SpatialEntityPseudo3D.cpp.

References distanceFromOrigin, and updateMeasuresIfRequired().

Referenced by multiscale::analysis::Detector::addSpatialEntityPropertiesToTree().

#### 6.177.3.10 double SpatialEntityPseudo3D::getPerimeter( )

Get the perimeter.

Definition at line 32 of file SpatialEntityPseudo3D.cpp.

References perimeter, and updateMeasuresIfRequired().

Referenced by multiscale::analysis::Detector::addSpatialEntityPropertiesToTree().

**6.177.3.11 double SpatialEntityPseudo3D::getRectangularMeasure( )**

Get the measure indicating how much the shape of the contour resembles a rectangle.

Definition at line 64 of file SpatialEntityPseudo3D.cpp.

References rectangularMeasure.

Referenced by multiscale::analysis::Detector::addSpatialEntityPropertiesToTree().

**6.177.3.12 Shape2D SpatialEntityPseudo3D::getShape( )**

Get the shape best fitting the spatial collection.

Definition at line 50 of file SpatialEntityPseudo3D.cpp.

References shape, and updateMeasuresIfRequired().

Referenced by multiscale::analysis::SimulationClusterDetector::outputClusterShape().

**6.177.3.13 std::string SpatialEntityPseudo3D::getShapeAsString( )**

Get the shape best fitting the spatial collection as a string.

Definition at line 56 of file SpatialEntityPseudo3D.cpp.

References shapeAsString().

**6.177.3.14 double SpatialEntityPseudo3D::getTriangularMeasure( )**

Get the measure indicating how much the shape of the contour resembles a triangle.

Definition at line 60 of file SpatialEntityPseudo3D.cpp.

References triangularMeasure.

Referenced by multiscale::analysis::Detector::addSpatialEntityPropertiesToTree().

**6.177.3.15 void SpatialEntityPseudo3D::initialise( ) [private]**

Initialisation function for the class.

Reimplemented in [multiscale::analysis::Cluster](#).

Definition at line 205 of file SpatialEntityPseudo3D.cpp.

References area, circularMeasure, perimeter, rectangularMeasure, shape, spatialEntityShapeArea, triangularMeasure, multiscale::analysis::Undefined, and updateFlag.

Referenced by SpatialEntityPseudo3D().

---

**6.177.3.16 virtual double multiscale::analysis::SpatialEntityPseudo3D::isCircularMeasure ( ) [protected, pure virtual]**

Get the measure that the cluster has a circular shape.

Implemented in [multiscale::analysis::Region](#), and [multiscale::analysis::Cluster](#).

Referenced by [updateShape\(\)](#).

**6.177.3.17 virtual double multiscale::analysis::SpatialEntityPseudo3D::isRectangularMeasure ( ) [protected, pure virtual]**

Get the measure that the cluster has a rectangular shape.

Implemented in [multiscale::analysis::Region](#), and [multiscale::analysis::Cluster](#).

Referenced by [updateShape\(\)](#).

**6.177.3.18 virtual double multiscale::analysis::SpatialEntityPseudo3D::isTriangularMeasure ( ) [protected, pure virtual]**

Get the measure that the cluster has a triangular shape.

Implemented in [multiscale::analysis::Region](#), and [multiscale::analysis::Cluster](#).

Referenced by [updateShape\(\)](#).

**6.177.3.19 double SpatialEntityPseudo3D::normalisedShapeMeasure ( double shapeArea ) [protected]**

Get the normalised shape measure ([0, 1]) that the spatial entity has a particular shape.

The normalised shape measure is the result of dividing [spatialEntityShapeArea](#) to [shapeArea](#)

#### Parameters

|                  |                       |
|------------------|-----------------------|
| <i>shapeArea</i> | The area of the shape |
|------------------|-----------------------|

Definition at line 150 of file [SpatialEntityPseudo3D.cpp](#).

References [multiscale::Numeric::division\(\)](#), [multiscale::Numeric::greaterOrEqual\(\)](#), [multiscale::Numeric::lessOrEqual\(\)](#), and [spatialEntityShapeArea](#).

Referenced by [multiscale::analysis::Cluster::isCircularMeasure\(\)](#), [multiscale::analysis::Region::isCircularMeasure\(\)](#), [multiscale::analysis::Cluster::isRectangularMeasure\(\)](#), [multiscale::analysis::Region::isRectangularMeasure\(\)](#), [multiscale::analysis::Cluster::isTriangularMeasure\(\)](#), and [multiscale::analysis::Region::isTriangularMeasure\(\)](#).

**6.177.3.20 std::string SpatialEntityPseudo3D::shapeAsString( ) [protected]**

Return the shape of the cluster as a string.

Definition at line 162 of file SpatialEntityPseudo3D.cpp.

References shape, STR\_CIRCLE, STR\_RECTANGLE, STR\_TRIANGLE, and STR\_UNDEFINED.

Referenced by fieldValuesToString(), and getShapeAsString().

**6.177.3.21 std::string SpatialEntityPseudo3D::toString( )**

Get the string representation of all field values.

Definition at line 85 of file SpatialEntityPseudo3D.cpp.

References fieldValuesToString(), and updateMeasuresIfRequired().

**6.177.3.22 virtual SpatialEntityPseudo3DType multiscale::analysis::-  
SpatialEntityPseudo3D::type( ) [protected, pure  
virtual]**

Return the type of the pseudo 3D spatial entity.

Implemented in [multiscale::analysis::Region](#), and [multiscale::analysis::Cluster](#).

Referenced by typeAsString().

**6.177.3.23 std::string SpatialEntityPseudo3D::typeAsString( )**

Return the type of the pseudo 3D spatial entity as a string.

Definition at line 91 of file SpatialEntityPseudo3D.cpp.

References ERR\_UNDEFINED\_TYPE, STR\_CLUSTER, STR\_REGION, STR\_UNDEFINED, and type().

Referenced by multiscale::analysis::Detector::addSpatialEntityTypeToPropertyTree().

**6.177.3.24 virtual void multiscale::analysis::SpatialEntityPseudo3D::updateArea( ) [protected, pure virtual]**

Update the value of the area.

Implemented in [multiscale::analysis::Region](#), and [multiscale::analysis::Cluster](#).

Referenced by updateMeasures().

6.177.3.25 **virtual void multiscale::analysis::SpatialEntityPseudo3D::updateCentrePoint( )** [protected, pure virtual]

Update the point defining the centre of the cluster.

Implemented in [multiscale::analysis::Region](#), and [multiscale::analysis::Cluster](#).

Referenced by [updateMeasures\(\)](#).

6.177.3.26 **virtual void multiscale::analysis::SpatialEntityPseudo3D::updateClusterednessDegree( )** [protected, pure virtual]

Update the value of the clusteredness degree.

Implemented in [multiscale::analysis::Region](#), and [multiscale::analysis::Cluster](#).

Referenced by [updateMeasures\(\)](#).

6.177.3.27 **virtual void multiscale::analysis::SpatialEntityPseudo3D::updateDensity( )** [protected, pure virtual]

Update the value of the density.

Implemented in [multiscale::analysis::Region](#), and [multiscale::analysis::Cluster](#).

Referenced by [updateMeasures\(\)](#).

6.177.3.28 **void SpatialEntityPseudo3D::updateMeasures( )** [protected]

Update the values of all measures.

Definition at line 120 of file [SpatialEntityPseudo3D.cpp](#).

References [updateArea\(\)](#), [updateCentrePoint\(\)](#), [updateClusterednessDegree\(\)](#), [updateDensity\(\)](#), [updatePerimeter\(\)](#), [updateShape\(\)](#), and [updateSpatialEntityShapeArea\(\)](#).

Referenced by [updateMeasuresIfRequired\(\)](#).

6.177.3.29 **void SpatialEntityPseudo3D::updateMeasuresIfRequired( )** [protected]

Update the values of all measures if required.

Definition at line 112 of file [SpatialEntityPseudo3D.cpp](#).

References [updateFlag](#), and [updateMeasures\(\)](#).

Referenced by [getAngle\(\)](#), [getArea\(\)](#), [getCentre\(\)](#), [getClusterednessDegree\(\)](#), [getDensity\(\)](#), [getDistanceFromOrigin\(\)](#), [multiscale::analysis::Cluster::getMinAreaEnclosingCircleCentre\(\)](#), and [multiscale::analysis::Cluster::getMinAreaEnclosingCircle](#).

Radius(), multiscale::analysis::Cluster::getMinAreaEnclosingRect(), multiscale::analysis::Cluster::getMinAreaEnclosingTriangle(), getPerimeter(), getShape(), and toString().

**6.177.3.30 virtual void multiscale::analysis::SpatialEntityPseudo3D::updatePerimeter( ) [protected, pure virtual]**

Update the value of the perimeter.

Implemented in [multiscale::analysis::Region](#), and [multiscale::analysis::Cluster](#).

Referenced by updateMeasures().

**6.177.3.31 void SpatialEntityPseudo3D::updateShape( ) [protected]**

Update the shape of the cluster.

Definition at line 130 of file SpatialEntityPseudo3D.cpp.

References multiscale::analysis::Circle, circularMeasure, isCircularMeasure(), isRectangularMeasure(), isTriangularMeasure(), multiscale::analysis::Rectangle, rectangularMeasure, shape, multiscale::analysis::Triangle, and triangularMeasure.

Referenced by updateMeasures().

**6.177.3.32 virtual void multiscale::analysis::SpatialEntityPseudo3D::updateSpatialEntityShapeArea( ) [protected, pure virtual]**

Update the value of the spatial entity shape area.

Implemented in [multiscale::analysis::Region](#), and [multiscale::analysis::Cluster](#).

Referenced by updateMeasures().

## 6.177.4 Member Data Documentation

**6.177.4.1 double multiscale::analysis::SpatialEntityPseudo3D::angle [protected]**

Angle of the region wrt the origin

Definition at line 31 of file SpatialEntityPseudo3D.hpp.

Referenced by fieldValuesToString(), getAngle(), multiscale::analysis::Cluster::initialise(), multiscale::analysis::Region::Region(), and multiscale::analysis::Cluster::setOriginDependentMembers().

**6.177.4.2 double multiscale::analysis::SpatialEntityPseudo3D::area**  
[protected]

Area of the spatial entity

Definition at line 27 of file SpatialEntityPseudo3D.hpp.

Referenced by multiscale::analysis::Region::isValidInputValues(), fieldValuesToString(), getArea(), initialise(), multiscale::analysis::Cluster::updateArea(), and multiscale::analysis::Region::updateArea().

**6.177.4.3 cv::Point2f multiscale::analysis::SpatialEntityPseudo3D::centre**  
[protected]

Point defining the centre of the spatial entity

Definition at line 47 of file SpatialEntityPseudo3D.hpp.

Referenced by fieldValuesToString(), getCentre(), multiscale::analysis::Cluster::updateCentrePoint(), multiscale::analysis::Region::updateCentrePointWhenRegionDefinedByMultiplePoints(), and multiscale::analysis::Region::updateCentrePointWhenRegionDefinedBySinglePoint().

**6.177.4.4 double multiscale::analysis::SpatialEntityPseudo3D::circularMeasure**  
[protected]

Measure ([0, 1]) indicating the similarity between the shape of the spatial collection and a circle

Definition at line 37 of file SpatialEntityPseudo3D.hpp.

Referenced by fieldValuesToString(), getCircularMeasure(), initialise(), and updateShape().

**6.177.4.5 double multiscale::analysis::SpatialEntityPseudo3D::clusteredness-Degree** [protected]

Degree of clusteredness

Definition at line 19 of file SpatialEntityPseudo3D.hpp.

Referenced by multiscale::analysis::Region::isValidInputValues(), fieldValuesToString(), getClusterednessDegree(), multiscale::analysis::Cluster::initialise(), multiscale::analysis::Cluster::updateClusterednessDegree(), and multiscale::analysis::Region::updateClusterednessDegree().

**6.177.4.6 const bool SpatialEntityPseudo3D::CONVEX\_HULL\_CLOCKWISE = true**  
[static, protected]

Definition at line 191 of file SpatialEntityPseudo3D.hpp.

---

Referenced by multiscale::analysis::Cluster::getEntitiesConvexHull(), and multiscale::analysis::Region::isTriangularMeasure().

**6.177.4.7 double multiscale::analysis::SpatialEntityPseudo3D::density [protected]**

For regions: The average intensity of the pixels in the region normalised to the interval [0, 1]

For clusters: Degree of pile up

Definition at line 20 of file SpatialEntityPseudo3D.hpp.

Referenced by fieldValuesToString(), getDensity(), multiscale::analysis::Cluster::initialise(), multiscale::analysis::Region::Region(), and multiscale::analysis::Cluster::updateDensity().

**6.177.4.8 double multiscale::analysis::SpatialEntityPseudo3D::distanceFromOrigin [protected]**

Distance from the origin

Definition at line 30 of file SpatialEntityPseudo3D.hpp.

Referenced by fieldValuesToString(), getDistanceFromOrigin(), multiscale::analysis::Cluster::initialise(), multiscale::analysis::Region::Region(), and multiscale::analysis::Cluster::setOriginDependentMembers().

**6.177.4.9 const std::string SpatialEntityPseudo3D::ERR\_INPUT = "Invalid input parameters were provided to the constructor." [static, protected]**

Definition at line 188 of file SpatialEntityPseudo3D.hpp.

Referenced by multiscale::analysis::Region::validateInputValues().

**6.177.4.10 const std::string SpatialEntityPseudo3D::ERR\_UNDEFINED\_TYPE = "Pseudo 3D spatial entity of undefined type encountered." [static, protected]**

Definition at line 189 of file SpatialEntityPseudo3D.hpp.

Referenced by typeAsString().

**6.177.4.11 cv::Point2f multiscale::analysis::SpatialEntityPseudo3D::minAreaEnclosingCircleCentre [protected]**

The minimum area enclosing circle centre point

Definition at line 61 of file SpatialEntityPseudo3D.hpp.

Referenced by multiscale::analysis::Cluster::getMinAreaEnclosingCircleCentre(), multiscale::analysis::Cluster::isCircularMeasure(), and multiscale::analysis::Region::isCircularMeasure().

**6.177.4.12 float multiscale::analysis::SpatialEntityPseudo3D::minAreaEnclosingCircleRadius [protected]**

The minimum area enclosing circle radius

Definition at line 63 of file SpatialEntityPseudo3D.hpp.

Referenced by multiscale::analysis::Cluster::getMinAreaEnclosingCircleRadius(), multiscale::analysis::Cluster::initialise(), multiscale::analysis::Cluster::isCircularMeasure(), and multiscale::analysis::Region::isCircularMeasure().

**6.177.4.13 cv::RotatedRect multiscale::analysis::SpatialEntityPseudo3D::minAreaEnclosingRect [protected]**

The minimum area enclosing rectangle

Definition at line 58 of file SpatialEntityPseudo3D.hpp.

Referenced by multiscale::analysis::Cluster::getMinAreaEnclosingRect(), multiscale::analysis::Cluster::isRectangularMeasure(), and multiscale::analysis::Region::isRectangularMeasure().

**6.177.4.14 std::vector<cv::Point2f> multiscale::analysis::-  
SpatialEntityPseudo3D::minAreaEnclosingTriangle  
[protected]**

The minimum area enclosing triangle

Definition at line 55 of file SpatialEntityPseudo3D.hpp.

Referenced by multiscale::analysis::Cluster::getMinAreaEnclosingTriangle(), multiscale::analysis::Cluster::initialise(), multiscale::analysis::Cluster::isTriangularMeasure(), and multiscale::analysis::Region::isTriangularMeasure().

**6.177.4.15 const std::string SpatialEntityPseudo3D::OUTPUT\_SEPARATOR = ","  
[static, protected]**

Definition at line 186 of file SpatialEntityPseudo3D.hpp.

Referenced by fieldValuesToString().

**6.177.4.16 double multiscale::analysis::SpatialEntityPseudo3D::perimeter  
[protected]**

Perimeter of the spatial entity

Definition at line 28 of file SpatialEntityPseudo3D.hpp.

Referenced by fieldValuesToString(), getPerimeter(), initialise(), multiscale::analysis::Cluster::updatePerimeter(), and multiscale::analysis::Region::updatePerimeter().

#### 6.177.4.17 double multiscale::analysis::SpatialEntityPseudo3D::rectangularMeasure [protected]

Measure ([0, 1]) indicating the similarity between the shape of the spatial collection and a rectangle

Definition at line 35 of file SpatialEntityPseudo3D.hpp.

Referenced by fieldValuesToString(), getRectangularMeasure(), initialise(), and updateShape().

#### 6.177.4.18 Shape2D multiscale::analysis::SpatialEntityPseudo3D::shape [protected]

Shape of the spatial entity

Definition at line 46 of file SpatialEntityPseudo3D.hpp.

Referenced by getShape(), initialise(), shapeAsString(), and updateShape().

#### 6.177.4.19 double multiscale::analysis::SpatialEntityPseudo3D::spatialEntityShapeArea [protected]

The area of the spatial entity considered when computing its shape. The difference between the shape area and the regular area is that the former considers the entities centre points while the latter the entities border points

Definition at line 40 of file SpatialEntityPseudo3D.hpp.

Referenced by initialise(), normalisedShapeMeasure(), multiscale::analysis::Cluster::updateSpatialEntityShapeArea(), and multiscale::analysis::Region::updateSpatialEntityShapeArea().

#### 6.177.4.20 const std::string SpatialEntityPseudo3D::STR\_CIRCLE = "circular" [static, protected]

Definition at line 183 of file SpatialEntityPseudo3D.hpp.

Referenced by shapeAsString().

#### 6.177.4.21 const std::string SpatialEntityPseudo3D::STR\_CLUSTER = "cluster" [static, protected]

Definition at line 179 of file SpatialEntityPseudo3D.hpp.

Referenced by typeAsString().

6.177.4.22 **const std::string SpatialEntityPseudo3D::STR\_RECTANGLE = "rectangular"**  
[static, protected]

Definition at line 182 of file SpatialEntityPseudo3D.hpp.

Referenced by shapeAsString().

6.177.4.23 **const std::string SpatialEntityPseudo3D::STR\_REGION = "region"**  
[static, protected]

Definition at line 178 of file SpatialEntityPseudo3D.hpp.

Referenced by typeAsString().

6.177.4.24 **const std::string SpatialEntityPseudo3D::STR\_TRIANGLE = "triangular"**  
[static, protected]

Definition at line 181 of file SpatialEntityPseudo3D.hpp.

Referenced by shapeAsString().

6.177.4.25 **const std::string SpatialEntityPseudo3D::STR\_UNDEFINED = "undefined"**  
[static, protected]

Definition at line 184 of file SpatialEntityPseudo3D.hpp.

Referenced by shapeAsString(), and typeAsString().

6.177.4.26 **double multiscale::analysis::SpatialEntityPseudo3D::triangularMeasure**  
[protected]

Measure ([0, 1]) indicating the similarity between the shape of the spatial collection and a triangle

Definition at line 33 of file SpatialEntityPseudo3D.hpp.

Referenced by fieldValuesToString(), getTriangularMeasure(), initialise(), and updateShape().

6.177.4.27 **bool multiscale::analysis::SpatialEntityPseudo3D::updateFlag**  
[protected]

Flag indicating if the field values dependent on the collection of entities need to be updated. This flag is used for lazy evaluation purposes, such that new field values are computed only when required

Definition at line 49 of file SpatialEntityPseudo3D.hpp.

Referenced by multiscale::analysis::Cluster::addEntity(), multiscale::analysis::Cluster::initialise(), initialise(), and updateMeasuresIfRequired().

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

- `SpatialEntityPseudo3D.hpp`
- `SpatialEntityPseudo3D.cpp`

## **6.178 multiscale::verification::SpatialMeasureAttribute Class Reference**

Class for representing a spatial measure attribute.

```
#include <SpatialMeasureAttribute.hpp>
```

### **Public Attributes**

- `SpatialMeasureType spatialMeasureType`

#### **6.178.1 Detailed Description**

Class for representing a spatial measure attribute.

Definition at line 79 of file `SpatialMeasureAttribute.hpp`.

#### **6.178.2 Member Data Documentation**

##### **6.178.2.1 SpatialMeasureType multiscale::verification::SpatialMeasureAttribute::spatialMeasureType**

The spatial measure type

Definition at line 83 of file `SpatialMeasureAttribute.hpp`.

Referenced by `multiscale::verification::SpatialMeasureCollectionVisitor::operator()()`, `multiscale::verification::FilterNumericVisitor::operator()()`, and `multiscale::verification::ConstraintVisitor::operator()()`.

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

- `SpatialMeasureAttribute.hpp`

## **6.179 multiscale::analysis::SpatialMeasureCalculator Class Reference**

Class for computing spatial measures.

```
#include <SpatialMeasureCalculator.hpp>
```

## Static Public Member Functions

- static int `computePolygonArea` (const std::vector< cv::Point > &polygon)  
*Compute the area of the provided polygon.*
- static int `computePolygonHoleArea` (const std::vector< cv::Point > &hole, const std::vector< cv::Point > &polygon)  
*Compute the area of the given hole within the provided polygon.*
- static int `computeCircleArea` (const cv::Point2f &circleOrigin, double circleRadius)  
*Compute the area of the provided circle.*
- static int `computePolygonPerimeter` (const std::vector< cv::Point > &polygon)  
*Compute the perimeter of the provided polygon.*

## Static Private Member Functions

- static cv::Mat `drawFilledPolygonOnImage` (const std::vector< cv::Point > &polygon)  
*Return an image in which the polygon was drawn.*
- static void `subtractPolygonBorderFromImage` (const std::vector< cv::Point > &polygon, cv::Mat &image)  
*Set the value of the image pixels defined by the polygon to the minimum value.*
- static cv::Mat `drawFilledCircleOnImage` (const cv::Point2f &circleOrigin, double circleRadius)  
*Return an image in which the circle was drawn.*
- static int `computeNrOfMinValuePointSides` (int rowIndex, int colIndex, const cv::Mat &image)  
*Compute the number of minimum value sides of the point in the given image.*
- static bool `isMinValuePointUp` (int rowIndex, int colIndex, const cv::Mat &image)  
*Check if there is a minimum value point above the given point.*
- static bool `isMinValuePointLeft` (int rowIndex, int colIndex, const cv::Mat &image)  
*Check if there is a minimum value point to the left of the given point.*
- static bool `isMinValuePointDown` (int rowIndex, int colIndex, const cv::Mat &image)  
*Check if there is a minimum value point below the given point.*
- static bool `isMinValuePointRight` (int rowIndex, int colIndex, const cv::Mat &image)  
*Check if there is a minimum value point to the right of the given point.*

## Static Private Attributes

- static const int `POINT_MIN_VALUE` = 0
- static const int `POINT_MAX_VALUE` = 255

### 6.179.1 Detailed Description

Class for computing spatial measures.

Definition at line 15 of file SpatialMeasureCalculator.hpp.

### 6.179.2 Member Function Documentation

**6.179.2.1 int SpatialMeasureCalculator::computeCircleArea ( const cv::Point2f & *circleOrigin*, double *circleRadius* ) [static]**

Compute the area of the provided circle.

The area of the circle is computed as the number of pixels on, respectively contained by its contour.

#### Parameters

|                     |                          |
|---------------------|--------------------------|
| <i>circleOrigin</i> | The origin of the circle |
| <i>circleRadius</i> | The radius of the circle |

Definition at line 32 of file SpatialMeasureCalculator.cpp.

References drawFilledCircleOnImage().

**6.179.2.2 int SpatialMeasureCalculator::computeNrOfMinValuePointSides ( int *rowIndex*, int *collIndex*, const cv::Mat & *image* ) [static, private]**

Compute the number of minimum value sides of the point in the given image.

#### Parameters

|                  |                          |
|------------------|--------------------------|
| <i>rowIndex</i>  | The point's row index    |
| <i>collIndex</i> | The point's column index |
| <i>image</i>     | The reference image      |

Definition at line 114 of file SpatialMeasureCalculator.cpp.

References isMinValuePointDown(), isMinValuePointLeft(), isMinValuePointRight(), and isMinValuePointUp().

Referenced by computePolygonPerimeter().

**6.179.2.3 int SpatialMeasureCalculator::computePolygonArea ( const std::vector<cv::Point > & *polygon* ) [static]**

Compute the area of the provided polygon.

The area of the polygon is computed as the number of pixels on, respectively contained by its contour.

**Parameters**

|                |                                            |
|----------------|--------------------------------------------|
| <i>polygon</i> | The polygon for which the area is computed |
|----------------|--------------------------------------------|

Definition at line 8 of file SpatialMeasureCalculator.cpp.

References drawFilledPolygonOnImage().

Referenced by multiscale::analysis::Region::computeArealfOuterBoderDefined(), multiscale::analysis::Region::computeClusterednessDegreelfOuterBorderDefined(), multiscale::analysis::RegionDetector::isValidContour(), and multiscale::analysis::Cluster::updateArea().

```
6.179.2.4 int SpatialMeasureCalculator::computePolygonHoleArea ( const
    std::vector< cv::Point > & hole, const std::vector< cv::Point > & polygon )
    [static]
```

Compute the area of the given hole within the provided polygon.

The area of the hole is computed as the number of pixels on, respectively contained by the hole's contour and which are not on the polygon's contour.

**Parameters**

|                |                                         |
|----------------|-----------------------------------------|
| <i>hole</i>    | The hole for which the area is computed |
| <i>polygon</i> | The polygon which contains the hole     |

Definition at line 17 of file SpatialMeasureCalculator.cpp.

References drawFilledPolygonOnImage(), and subtractPolygonBorderFromImage().

Referenced by multiscale::analysis::Region::computeArealfOuterBoderDefined(), multiscale::analysis::Region::computeClusterednessDegreelfOuterBorderDefined(), and multiscale::analysis::RegionDetector::isValidHole().

```
6.179.2.5 int SpatialMeasureCalculator::computePolygonPerimeter ( const
    std::vector< cv::Point > & polygon ) [static]
```

Compute the perimeter of the provided polygon.

The perimeter of the polygon is computed as the number of pixel sides which are exterior to the polygon.

**Parameters**

|                |                                                 |
|----------------|-------------------------------------------------|
| <i>polygon</i> | The polygon for which the perimeter is computed |
|----------------|-------------------------------------------------|

Definition at line 42 of file SpatialMeasureCalculator.cpp.

References computeNrOfMinValuePointSides(), drawFilledPolygonOnImage(), and PO-INT\_MAX\_VALUE.

Referenced by multiscale::analysis::Cluster::updatePerimeter(), and multiscale-

::analysis::Region::updatePerimeter().

**6.179.2.6 cv::Mat SpatialMeasureCalculator::drawFilledCircleOnImage ( const cv::Point2f & *circleOrigin*, double *circleRadius* ) [static, private]**

Return an image in which the circle was drawn.

The filled white circle is drawn at a minimum distance of 1 pixel from each border of the image. The background of the image is black.

**Parameters**

|                     |                          |
|---------------------|--------------------------|
| <i>circleOrigin</i> | The origin of the circle |
| <i>circleRadius</i> | The radius of the circle |

Definition at line 95 of file SpatialMeasureCalculator.cpp.

References POINT\_MAX\_VALUE.

Referenced by computeCircleArea().

**6.179.2.7 cv::Mat SpatialMeasureCalculator::drawFilledPolygonOnImage ( const std::vector< cv::Point > & *polygon* ) [static, private]**

Return an image in which the polygon was drawn.

The filled white polygon is drawn at a minimum distance of 1 pixel from each border of the image. The background of the image is black.

**Parameters**

|                |                                              |
|----------------|----------------------------------------------|
| <i>polygon</i> | The polygon which will be drawn on the image |
|----------------|----------------------------------------------|

Definition at line 61 of file SpatialMeasureCalculator.cpp.

References POINT\_MAX\_VALUE.

Referenced by computePolygonArea(), computePolygonHoleArea(), and computePolygonPerimeter().

**6.179.2.8 bool SpatialMeasureCalculator::isMinValuePointDown ( int *rowIndex*, int *collIndex*, const cv::Mat & *image* ) [static, private]**

Check if there is a minimum value point below the given point.

**Parameters**

|                  |                                |
|------------------|--------------------------------|
| <i>rowIndex</i>  | The given point's row index    |
| <i>collIndex</i> | The given point's column index |
| <i>image</i>     | The reference image            |

Definition at line 138 of file SpatialMeasureCalculator.cpp.

References POINT\_MIN\_VALUE.

Referenced by computeNrOfMinValuePointSides().

**6.179.2.9 bool SpatialMeasureCalculator::isMinValuePointLeft ( int *rowIndex*, int *collIndex*, const cv::Mat & *image* ) [static, private]**

Check if there is a minimum value point to the left of the given point.

#### Parameters

|                  |                                |
|------------------|--------------------------------|
| <i>rowIndex</i>  | The given point's row index    |
| <i>collIndex</i> | The given point's column index |
| <i>image</i>     | The reference image            |

Definition at line 131 of file SpatialMeasureCalculator.cpp.

References POINT\_MIN\_VALUE.

Referenced by computeNrOfMinValuePointSides().

**6.179.2.10 bool SpatialMeasureCalculator::isMinValuePointRight ( int *rowIndex*, int *collIndex*, const cv::Mat & *image* ) [static, private]**

Check if there is a minimum value point to the right of the given point.

#### Parameters

|                  |                                |
|------------------|--------------------------------|
| <i>rowIndex</i>  | The given point's row index    |
| <i>collIndex</i> | The given point's column index |
| <i>image</i>     | The reference image            |

Definition at line 145 of file SpatialMeasureCalculator.cpp.

References POINT\_MIN\_VALUE.

Referenced by computeNrOfMinValuePointSides().

**6.179.2.11 bool SpatialMeasureCalculator::isMinValuePointUp ( int *rowIndex*, int *collIndex*, const cv::Mat & *image* ) [static, private]**

Check if there is a minimum value point above the given point.

#### Parameters

|                  |                                |
|------------------|--------------------------------|
| <i>rowIndex</i>  | The given point's row index    |
| <i>collIndex</i> | The given point's column index |
| <i>image</i>     | The reference image            |

Definition at line 124 of file SpatialMeasureCalculator.cpp.

References POINT\_MIN\_VALUE.

Referenced by computeNrOfMinValuePointSides().

```
6.179.2.12 void SpatialMeasureCalculator::subtractPolygonBorderFromImage (
    const std::vector< cv::Point > & polygon, cv::Mat & image ) [static,
    private]
```

Set the value of the image pixels defined by the polygon to the minimum value.

#### Parameters

|                |                                                                           |
|----------------|---------------------------------------------------------------------------|
| <i>polygon</i> | The polygon which will be drawn using minimum value on the provided image |
| <i>image</i>   | The considered image                                                      |

Definition at line 83 of file SpatialMeasureCalculator.cpp.

References POINT\_MIN\_VALUE.

Referenced by computePolygonHoleArea().

### 6.179.3 Member Data Documentation

```
6.179.3.1 const int SpatialMeasureCalculator::POINT_MAX_VALUE = 255
[static, private]
```

Definition at line 128 of file SpatialMeasureCalculator.hpp.

Referenced by computePolygonPerimeter(), drawFilledCircleOnImage(), and drawFilledPolygonOnImage().

```
6.179.3.2 const int SpatialMeasureCalculator::POINT_MIN_VALUE = 0 [static,
private]
```

Definition at line 127 of file SpatialMeasureCalculator.hpp.

Referenced by isMinValuePointDown(), isMinValuePointLeft(), isMinValuePointRight(), isMinValuePointUp(), and subtractPolygonBorderFromImage().

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

- SpatialMeasureCalculator.hpp
- SpatialMeasureCalculator.cpp

## 6.180 multiscale::verification::SpatialMeasureCollectionAttribute - Class Reference

Class used to represent a spatial measure collection.

```
#include <SpatialMeasureCollectionAttribute.hpp>
```

### Public Attributes

- [SpatialMeasureCollectionType spatialMeasureCollection](#)

#### 6.180.1 Detailed Description

Class used to represent a spatial measure collection.

Definition at line 28 of file SpatialMeasureCollectionAttribute.hpp.

#### 6.180.2 Member Data Documentation

##### 6.180.2.1 [SpatialMeasureCollectionType multiscale::verification::SpatialMeasureCollectionAttribute::spatialMeasureCollection](#)

The spatial measure collection

Definition at line 32 of file SpatialMeasureCollectionAttribute.hpp.

Referenced by [multiscale::verification::NumericMeasureCollectionEvaluator::evaluate-SpatialMeasureCollection\(\)](#).

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

- [SpatialMeasureCollectionAttribute.hpp](#)

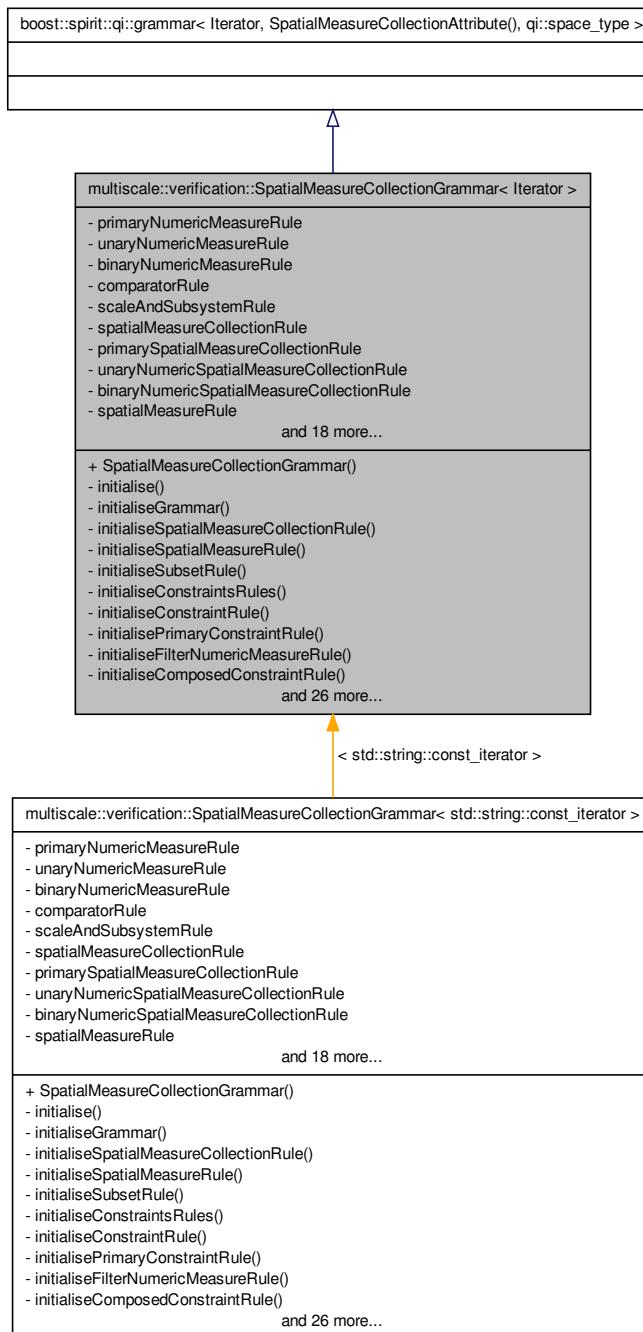
## 6.181 multiscale::verification::SpatialMeasureCollectionGrammar< Iterator > Class Template Reference

The grammar for parsing spatial measure collection statements.

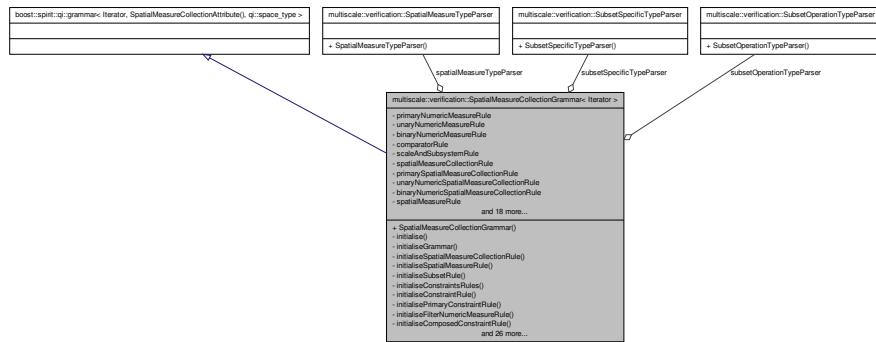
```
#include <SpatialMeasureCollectionGrammar.hpp>
```

Inheritance diagram for multiscale::verification::SpatialMeasureCollectionGrammar< -

Iterator >:



Collaboration diagram for multiscale::verification::SpatialMeasureCollectionGrammar< Iterator >:



## Public Member Functions

- `SpatialMeasureCollectionGrammar ()`

## Private Member Functions

- `void initialise ()`  
*Initialisation function.*
- `void initialiseGrammar ()`  
*Initialise the grammar.*
- `void initialiseSpatialMeasureCollectionRule ()`  
*Initialise the spatial measure collection rule.*
- `void initialiseSpatialMeasureRule ()`  
*Initialise the spatial measure rule.*
- `void initialiseSubsetRule ()`  
*Initialise the subset rule.*
- `void initialiseConstraintsRules ()`  
*Initialise the constraints rules.*
- `void initialiseConstraintRule ()`  
*Initialise the constraint rule.*
- `void initialisePrimaryConstraintRule ()`  
*Initialise the primary constraint rule.*
- `void initialiseFilterNumericMeasureRule ()`  
*Initialise the filter numeric measure rule.*
- `void initialiseComposedConstraintRule ()`  
*Initialise the composed constraint rule.*
- `void initialiseDebugSupport ()`

- Initialise debug support.*
  - void [assignNamesToRules \(\)](#)  
        *Assign names to the rules.*
  - void [assignNamesToSpatialMeasureCollectionRules \(\)](#)  
        *Assign names to the spatial measure collection rules.*
  - void [assignNamesToNumericSpatialMeasureRules \(\)](#)  
        *Assign names to the numeric spatial measure rules.*
  - void [assignNamesToSubsetRules \(\)](#)  
        *Assign names to the subset rules.*
  - void [assignNamesToConstraintsRules \(\)](#)  
        *Assign names to constraints rules.*
  - void [assignNamesToConstraintRules \(\)](#)  
        *Assign names to the constraint rules.*
  - void [assignNamesToPrimaryConstraintRules \(\)](#)  
        *Assign names to the primary constraint rules.*
  - void [assignNamesToFilterNumericMeasureRules \(\)](#)  
        *Assign names to the filter numeric measure rules.*
  - void [assignNamesToComposedConstraintRules \(\)](#)  
        *Assign names to the composed constraint rules.*
  - void [assignNamesToSpatialMeasureRules \(\)](#)  
        *Assign names to the spatial measure rules.*
  - void [initialiseRulesDebugging \(\)](#)  
        *Initialise the debugging of rules.*
  - void [initialiseSpatialMeasureCollectionRuleDebugging \(\)](#)  
        *Initialise debugging for the spatial measure collection rule.*
  - void [initialiseSpatialMeasureRuleDebugging \(\)](#)  
        *Initialise debugging for the numeric spatial measure rule.*
  - void [initialiseSubsetRuleDebugging \(\)](#)  
        *Initialise debugging for the subset rules.*
  - void [initialiseConstraintsRulesDebugging \(\)](#)  
        *Initialise the debugging of the constraints rules.*
  - void [initialiseConstraintRuleDebugging \(\)](#)  
        *Initialise debugging for the constraint rule.*
  - void [initialisePrimaryConstraintRuleDebugging \(\)](#)  
        *Initialise debugging for the primary constraint rules.*
  - void [initialiseFilterNumericMeasureRuleDebugging \(\)](#)  
        *Initialise debugging for the filter numeric measure rules.*
  - void [initialiseComposedConstraintRuleDebugging \(\)](#)  
        *Initialise debugging for the composed constraint rule.*
  - void [initialiseErrorHandlerSupport \(\)](#)  
        *Initialise the error handling routines.*
  - void [initialiseSpatialMeasureCollectionErrorHandlerSupport \(\)](#)  
        *Initialise the numeric measure collection error handling support.*

- void `initialiseSubsetErrorHandlingSupport ()`  
*Initialise the subset error handling support.*
- void `initialiseConstraintsErrorHandlingSupport ()`  
*Initialise the constraints error handling support.*
- void `initialisePrimaryConstraintErrorHandlingSupport ()`  
*Initialise the primary constraint error handling support.*
- void `initialiseFilterNumericMeasureErrorHandlingSupport ()`  
*Initialise the filter numeric measure error handling support.*
- void `initialiseComposedConstraintErrorHandlingSupport ()`  
*Initialise the composed constraint error handling support.*

### Private Attributes

- std::shared\_ptr < PrimaryNumericMeasureGrammar < Iterator > > primary-NumericMeasureRule
- UnaryNumericMeasureGrammar < Iterator > unaryNumericMeasureRule
- BinaryNumericMeasureGrammar < Iterator > binaryNumericMeasureRule
- ComparatorGrammar< Iterator > comparatorRule
- ScaleAndSubsystemGrammar < Iterator > scaleAndSubsystemRule
- qi::rule< Iterator, SpatialMeasureCollectionAttribute(), qi::space\_type > spatial-MeasureCollectionRule
- qi::rule< Iterator, PrimarySpatialMeasureCollectionAttribute(), qi::space\_type > primarySpatialMeasureCollectionRule
- qi::rule< Iterator, UnaryNumericSpatialAttribute(), qi::space\_type > unary-NumericSpatialMeasureCollectionRule
- qi::rule< Iterator, BinaryNumericSpatialAttribute(), qi::space\_type > binary-NumericSpatialMeasureCollectionRule
- qi::rule< Iterator, SpatialMeasureAttribute(), qi::space\_type > spatialMeasure-Rule
- qi::rule< Iterator, SubsetAttribute(), qi::space\_type > subsetRule
- qi::rule< Iterator, SubsetSpecificAttribute(), qi::space\_type > subsetSpecific-Rule
- qi::rule< Iterator, FilterSubsetAttribute(), qi::space\_type > filterSubsetRule
- qi::rule< Iterator, SubsetSubsetOperationAttribute(), qi::space\_type > subset-SubsetOperationRule
- qi::rule< Iterator, ConstraintAttribute(), qi::space\_type > constraintRule
- qi::rule< Iterator, PrimaryConstraintAttribute(), qi::space\_type > primary-ConstraintRule
- qi::rule< Iterator, NotConstraintAttribute(), qi::space\_type > notConstraintRule
- qi::rule< Iterator, UnarySpatialConstraintAttribute(), qi::space\_type > unary-SpatialConstraintRule
- qi::rule< Iterator, UnaryScaleAndSubsystemConstraintAttribute(), qi::space\_-type > unaryScaleAndSubsystemConstraintRule
- qi::rule< Iterator, FilterNumericMeasureAttribute(), qi::space\_type > filter-NumericMeasureRule

- qi::rule< Iterator, UnaryNumericFilterAttribute(), qi::space\_type > unary-NumericFilterRule
- qi::rule< Iterator, BinaryNumericFilterAttribute(), qi::space\_type > binary-NumericFilterRule
- qi::rule< Iterator, AndConstraintAttribute(), qi::space\_type > andConstraintRule
- qi::rule< Iterator, OrConstraintAttribute(), qi::space\_type > orConstraintRule
- qi::rule< Iterator, ImplicationConstraintAttribute(), qi::space\_type > implication-ConstraintRule
- qi::rule< Iterator, EquivalenceConstraintAttribute(), qi::space\_type > equivalenceConstraintRule
- SubsetSpecificTypeParser subsetSpecificTypeParser
- SubsetOperationTypeParser subsetOperationTypeParser
- SpatialMeasureTypeParser spatialMeasureTypeParser

### 6.181.1 Detailed Description

```
template<typename Iterator>class multiscale::verification::SpatialMeasureCollectionGrammar< Iterator >
```

The grammar for parsing spatial measure collection statements.

Definition at line 38 of file SpatialMeasureCollectionGrammar.hpp.

### 6.181.2 Constructor & Destructor Documentation

```
6.181.2.1 template<typename Iterator > multiscale::verification::SpatialMeasure-  
CollectionGrammar< Iterator >::SpatialMeasureCollectionGrammar ( )
```

Definition at line 24 of file SpatialMeasureCollectionGrammarDefinition.hpp.

References multiscale::verification::SpatialMeasureCollectionGrammar< Iterator >->::initialise().

### 6.181.3 Member Function Documentation

```
6.181.3.1 template<typename Iterator > void multiscale::verification-  
::SpatialMeasureCollectionGrammar< Iterator  
>::assignNamesToComposedConstraintRules( ) [private]
```

Assign names to the composed constraint rules.

Definition at line 291 of file SpatialMeasureCollectionGrammarDefinition.hpp.

```
6.181.3.2 template<typename Iterator> void multiscale::verification::SpatialMeasure-
CollectionGrammar< Iterator >::assignNamesToConstraintRules( ) [private]
```

Assign names to the constraint rules.

Definition at line 268 of file SpatialMeasureCollectionGrammarDefinition.hpp.

```
6.181.3.3 template<typename Iterator> void multiscale::verification::SpatialMeasure-
CollectionGrammar< Iterator >::assignNamesToConstraintsRules( ) [private]
```

Assign names to constraints rules.

Definition at line 259 of file SpatialMeasureCollectionGrammarDefinition.hpp.

```
6.181.3.4 template<typename Iterator> void multiscale::verification-
::SpatialMeasureCollectionGrammar< Iterator
>::assignNamesToFilterNumericMeasureRules( ) [private]
```

Assign names to the filter numeric measure rules.

Definition at line 283 of file SpatialMeasureCollectionGrammarDefinition.hpp.

```
6.181.3.5 template<typename Iterator> void multiscale::verification-
::SpatialMeasureCollectionGrammar< Iterator
>::assignNamesToNumericSpatialMeasureRules( ) [private]
```

Assign names to the numeric spatial measure rules.

```
6.181.3.6 template<typename Iterator> void multiscale::verification::SpatialMeasure-
CollectionGrammar< Iterator >::assignNamesToPrimaryConstraintRules
( ) [private]
```

Assign names to the primary constraint rules.

Definition at line 274 of file SpatialMeasureCollectionGrammarDefinition.hpp.

```
6.181.3.7 template<typename Iterator> void multiscale::verification::Spatial-
MeasureCollectionGrammar< Iterator >::assignNamesToRules( ) [private]
```

Assign names to the rules.

Definition at line 226 of file SpatialMeasureCollectionGrammarDefinition.hpp.

---

**6.181.3.8 template<typename Iterator > void multiscale::verification-  
::SpatialMeasureCollectionGrammar< Iterator  
>::assignNamesToSpatialMeasureCollectionRules( ) [private]**

Assign names to the spatial measure collection rules.

Definition at line 235 of file SpatialMeasureCollectionGrammarDefinition.hpp.

**6.181.3.9 template<typename Iterator > void multiscale::verification::SpatialMeasure-  
CollectionGrammar< Iterator >::assignNamesToSpatialMeasureRules( ) [private]**

Assign names to the spatial measure rules.

Definition at line 244 of file SpatialMeasureCollectionGrammarDefinition.hpp.

**6.181.3.10 template<typename Iterator > void multiscale::verification::Spatial-  
MeasureCollectionGrammar< Iterator >::assignNamesToSubsetRules( ) [private]**

Assign names to the subset rules.

Definition at line 250 of file SpatialMeasureCollectionGrammarDefinition.hpp.

**6.181.3.11 template<typename Iterator > void multiscale::verification-  
::SpatialMeasureCollectionGrammar< Iterator >::initialise( ) [private]**

Initialisation function.

Definition at line 37 of file SpatialMeasureCollectionGrammarDefinition.hpp.

Referenced by multiscale::verification::SpatialMeasureCollectionGrammar< Iterator >-  
::SpatialMeasureCollectionGrammar().

**6.181.3.12 template<typename Iterator > void multiscale::verification-  
::SpatialMeasureCollectionGrammar< Iterator  
>::initialiseComposedConstraintErrorHandlingSupport( ) [private]**

Initialise the composed constraint error handling support.

Definition at line 458 of file SpatialMeasureCollectionGrammarDefinition.hpp.

References multiscale::verification::handleUnexpectedTokenError.

6.181.3.13 template<typename Iterator > void multiscale::verification::SpatialMeasureCollectionGrammar< Iterator >::initialiseComposedConstraintRule( ) [private]

Initialise the composed constraint rule.

Definition at line 201 of file SpatialMeasureCollectionGrammarDefinition.hpp.

6.181.3.14 template<typename Iterator > void multiscale::verification::SpatialMeasureCollectionGrammar< Iterator >::initialiseComposedConstraintRuleDebugging( ) [private]

Initialise debugging for the composed constraint rule.

Definition at line 365 of file SpatialMeasureCollectionGrammarDefinition.hpp.

6.181.3.15 template<typename Iterator > void multiscale::verification::SpatialMeasureCollectionGrammar< Iterator >::initialiseConstraintRule( ) [private]

Initialise the constraint rule.

Definition at line 137 of file SpatialMeasureCollectionGrammarDefinition.hpp.

6.181.3.16 template<typename Iterator > void multiscale::verification::SpatialMeasureCollectionGrammar< Iterator >::initialiseConstraintRuleDebugging( ) [private]

Initialise debugging for the constraint rule.

Definition at line 342 of file SpatialMeasureCollectionGrammarDefinition.hpp.

6.181.3.17 template<typename Iterator > void multiscale::verification::SpatialMeasureCollectionGrammar< Iterator >::initialiseConstraintsErrorHandlingSupport( ) [private]

Initialise the constraints error handling support.

Definition at line 412 of file SpatialMeasureCollectionGrammarDefinition.hpp.

6.181.3.18 template<typename Iterator > void multiscale::verification::SpatialMeasureCollectionGrammar< Iterator >::initialiseConstraintsRules( ) [private]

Initialise the constraints rules.

Definition at line 128 of file SpatialMeasureCollectionGrammarDefinition.hpp.

---

**6.181.3.19 template<typename Iterator > void multiscale::verification::SpatialMeasureCollectionGrammar< Iterator >::initialiseConstraintsRulesDebugging ( ) [private]**

Initialise the debugging of the constraints rules.

Definition at line 333 of file SpatialMeasureCollectionGrammarDefinition.hpp.

**6.181.3.20 template<typename Iterator > void multiscale::verification::SpatialMeasureCollectionGrammar< Iterator >::initialiseDebugSupport ( ) [private]**

Initialise debug support.

Definition at line 217 of file SpatialMeasureCollectionGrammarDefinition.hpp.

**6.181.3.21 template<typename Iterator > void multiscale::verification::SpatialMeasureCollectionGrammar< Iterator >::initialiseErrorHandlingSupport ( ) [private]**

Initialise the error handling routines.

Definition at line 374 of file SpatialMeasureCollectionGrammarDefinition.hpp.

**6.181.3.22 template<typename Iterator > void multiscale::verification::SpatialMeasureCollectionGrammar< Iterator >::initialiseFilterNumericMeasureErrorHandlingSupport ( ) [private]**

Initialise the filter numeric measure error handling support.

Definition at line 441 of file SpatialMeasureCollectionGrammarDefinition.hpp.

References multiscale::verification::handleUnexpectedTokenError.

**6.181.3.23 template<typename Iterator > void multiscale::verification::SpatialMeasureCollectionGrammar< Iterator >::initialiseFilterNumericMeasureRule ( ) [private]**

Initialise the filter numeric measure rule.

Definition at line 173 of file SpatialMeasureCollectionGrammarDefinition.hpp.

**6.181.3.24 template<typename Iterator > void multiscale::verification::SpatialMeasureCollectionGrammar< Iterator >::initialiseFilterNumericMeasureRuleDebugging ( ) [private]**

Initialise debugging for the filter numeric measure rules.

Definition at line 357 of file SpatialMeasureCollectionGrammarDefinition.hpp.

```
6.181.3.25 template<typename Iterator> void multiscale::verification::Spatial-
MeasureCollectionGrammar<Iterator>::initialiseGrammar( )
[private]
```

Initialise the grammar.

Definition at line 45 of file SpatialMeasureCollectionGrammarDefinition.hpp.

```
6.181.3.26 template<typename Iterator> void multiscale::verification-
::SpatialMeasureCollectionGrammar<Iterator>::initialisePrimaryConstraintErrorHandlingSupport( ) [private]
```

Initialise the primary constraint error handling support.

Definition at line 420 of file SpatialMeasureCollectionGrammarDefinition.hpp.

References multiscale::verification::handleUnexpectedTokenError.

```
6.181.3.27 template<typename Iterator> void multiscale::verification::SpatialMeasure-
CollectionGrammar<Iterator>::initialisePrimaryConstraintRule( )
[private]
```

Initialise the primary constraint rule.

Definition at line 150 of file SpatialMeasureCollectionGrammarDefinition.hpp.

```
6.181.3.28 template<typename Iterator> void multiscale::verification-
::SpatialMeasureCollectionGrammar<Iterator>::initialisePrimaryConstraintRuleDebugging( ) [private]
```

Initialise debugging for the primary constraint rules.

Definition at line 348 of file SpatialMeasureCollectionGrammarDefinition.hpp.

```
6.181.3.29 template<typename Iterator> void multiscale::verification::Spatial-
MeasureCollectionGrammar<Iterator>::initialiseRulesDebugging( )
[private]
```

Initialise the debugging of rules.

Definition at line 300 of file SpatialMeasureCollectionGrammarDefinition.hpp.

```
6.181.3.30 template<typename Iterator > void multiscale::verification-
::SpatialMeasureCollectionGrammar< Iterator
>::initialiseSpatialMeasureCollectionErrorHandlingSupport( )
[private]
```

Initialise the numeric measure collection error handling support.

Initialise the spatial measure collection error handling support.

Definition at line 382 of file SpatialMeasureCollectionGrammarDefinition.hpp.

References multiscale::verification::handleUnexpectedTokenError.

```
6.181.3.31 template<typename Iterator > void multiscale::verification::SpatialMeasure-
CollectionGrammar< Iterator >::initialiseSpatialMeasureCollectionRule
( ) [private]
```

Initialise the spatial measure collection rule.

Definition at line 54 of file SpatialMeasureCollectionGrammarDefinition.hpp.

```
6.181.3.32 template<typename Iterator > void multiscale::verification-
::SpatialMeasureCollectionGrammar< Iterator
>::initialiseSpatialMeasureCollectionRuleDebugging( ) [private]
```

Initialise debugging for the spatial measure collection rule.

Definition at line 309 of file SpatialMeasureCollectionGrammarDefinition.hpp.

```
6.181.3.33 template<typename Iterator > void multiscale::verification::Spatial-
MeasureCollectionGrammar< Iterator >::initialiseSpatialMeasureRule(
) [private]
```

Initialise the spatial measure rule.

Definition at line 89 of file SpatialMeasureCollectionGrammarDefinition.hpp.

```
6.181.3.34 template<typename Iterator > void multiscale::verification::SpatialMeasure-
CollectionGrammar< Iterator >::initialiseSpatialMeasureRuleDebugging
( ) [private]
```

Initialise debugging for the numeric spatial measure rule.

Initialise debugging for the spatial measure rule.

Definition at line 318 of file SpatialMeasureCollectionGrammarDefinition.hpp.

```
6.181.3.35 template<typename Iterator> void multiscale::verification::SpatialMeasure-
CollectionGrammar< Iterator >::initialiseSubsetErrorHandlingSupport (
) [private]
```

Initialise the subset error handling support.

Definition at line 399 of file SpatialMeasureCollectionGrammarDefinition.hpp.

References multiscale::verification::handleUnexpectedTokenError.

```
6.181.3.36 template<typename Iterator> void multiscale::verification::Spatial-
MeasureCollectionGrammar< Iterator >::initialiseSubsetRule ( ) [private]
```

Initialise the subset rule.

Definition at line 96 of file SpatialMeasureCollectionGrammarDefinition.hpp.

```
6.181.3.37 template<typename Iterator> void multiscale::verification::SpatialMeasure-
CollectionGrammar< Iterator >::initialiseSubsetRuleDebugging ( ) [private]
```

Initialise debugging for the subset rules.

Definition at line 324 of file SpatialMeasureCollectionGrammarDefinition.hpp.

#### 6.181.4 Member Data Documentation

```
6.181.4.1 template<typename Iterator> qi::rule<Iterator, AndConstraintAttribute(),
qi::space_type> multiscale::verification::SpatialMeasure-
CollectionGrammar< Iterator >::andConstraintRule [private]
```

The rule for parsing an "and" constraint

Definition at line 117 of file SpatialMeasureCollectionGrammar.hpp.

```
6.181.4.2 template<typename Iterator> qi::rule<Iterator, BinaryNumericFilterAttribute(),
qi::space_type> multiscale::verification::SpatialMeasure-
CollectionGrammar< Iterator >::binaryNumericFilterRule [private]
```

The rule for parsing a binary numeric filter measure

Definition at line 113 of file SpatialMeasureCollectionGrammar.hpp.

**6.181.4.3 template<typename Iterator> BinaryNumericMeasureGrammar<Iterator>  
multiscale::verification::SpatialMeasureCollectionGrammar< Iterator  
>::binaryNumericMeasureRule [private]**

The grammar for parsing binary numeric measures

Definition at line 53 of file SpatialMeasureCollectionGrammar.hpp.

**6.181.4.4 template<typename Iterator> qi::rule<Iterator,  
BinaryNumericSpatialAttribute(), qi::space\_type>  
multiscale::verification::SpatialMeasureCollectionGrammar< Iterator  
>::binaryNumericSpatialMeasureCollectionRule [private]**

The rule for parsing a binary numeric spatial measure collection

Definition at line 76 of file SpatialMeasureCollectionGrammar.hpp.

**6.181.4.5 template<typename Iterator> ComparatorGrammar<Iterator>  
multiscale::verification::SpatialMeasureCollectionGrammar< Iterator  
>::comparatorRule [private]**

The grammar for parsing comparators

Definition at line 57 of file SpatialMeasureCollectionGrammar.hpp.

**6.181.4.6 template<typename Iterator> qi::rule<Iterator, ConstraintAttribute(), qi::space\_-  
type> multiscale::verification::SpatialMeasureCollectionGrammar<  
Iterator >::constraintRule [private]**

The rule for parsing a constraint

Definition at line 93 of file SpatialMeasureCollectionGrammar.hpp.

**6.181.4.7 template<typename Iterator> qi::rule<Iterator,  
EquivalenceConstraintAttribute(), qi::space\_type>  
multiscale::verification::SpatialMeasureCollectionGrammar< Iterator  
>::equivalenceConstraintRule [private]**

The rule for parsing an "equivalence" constraint

Definition at line 124 of file SpatialMeasureCollectionGrammar.hpp.

**6.181.4.8 template<typename Iterator> qi::rule<Iterator,  
FilterNumericMeasureAttribute(), qi::space\_type>  
multiscale::verification::SpatialMeasureCollectionGrammar< Iterator  
>::filterNumericMeasureRule [private]**

The rule for parsing a filter numeric measure

Definition at line 107 of file SpatialMeasureCollectionGrammar.hpp.

```
6.181.4.9 template<typename Iterator> qi::rule<Iterator, FilterSubsetAttribute(), qi::space_type> multiscale::verification::SpatialMeasureCollectionGrammar<Iterator>::filterSubsetRule [private]
```

The rule for parsing a subset filter

Definition at line 87 of file SpatialMeasureCollectionGrammar.hpp.

```
6.181.4.10 template<typename Iterator> qi::rule<Iterator, ImplicationConstraintAttribute(), qi::space_type> multiscale::verification::SpatialMeasureCollectionGrammar<Iterator>::implicationConstraintRule [private]
```

The rule for parsing an "implication" constraint

Definition at line 121 of file SpatialMeasureCollectionGrammar.hpp.

```
6.181.4.11 template<typename Iterator> qi::rule<Iterator, NotConstraintAttribute(), qi::space_type> multiscale::verification::SpatialMeasureCollectionGrammar<Iterator>::notConstraintRule [private]
```

The rule for parsing a "not" constraint

Definition at line 98 of file SpatialMeasureCollectionGrammar.hpp.

```
6.181.4.12 template<typename Iterator> qi::rule<Iterator, OrConstraintAttribute(), qi::space_type> multiscale::verification::SpatialMeasureCollectionGrammar<Iterator>::orConstraintRule [private]
```

The rule for parsing an "or" constraint

Definition at line 119 of file SpatialMeasureCollectionGrammar.hpp.

```
6.181.4.13 template<typename Iterator> qi::rule<Iterator, PrimaryConstraintAttribute(), qi::space_type> multiscale::verification::SpatialMeasureCollectionGrammar<Iterator>::primaryConstraintRule [private]
```

The rule for parsing a primary constraint

Definition at line 96 of file SpatialMeasureCollectionGrammar.hpp.

```
6.181.4.14 template<typename Iterator> std::shared_ptr<-
    PrimaryNumericMeasureGrammar<Iterator> >
    multiscale::verification::SpatialMeasureCollectionGrammar< Iterator
    >::primaryNumericMeasureRule [private]
```

The grammar for parsing primary numeric measures

Definition at line 46 of file SpatialMeasureCollectionGrammar.hpp.

```
6.181.4.15 template<typename Iterator> qi::rule<Iterator, Primary-
    SpatialMeasureCollectionAttribute(), qi::space_type>
    multiscale::verification::SpatialMeasureCollectionGrammar< Iterator
    >::primarySpatialMeasureCollectionRule [private]
```

The rule for parsing a primary spatial measure collection

Definition at line 70 of file SpatialMeasureCollectionGrammar.hpp.

```
6.181.4.16 template<typename Iterator> ScaleAndSubsystemGrammar<Iterator>
    multiscale::verification::SpatialMeasureCollectionGrammar< Iterator
    >::scaleAndSubsystemRule [private]
```

The grammar for parsing scales and subsystems

Definition at line 60 of file SpatialMeasureCollectionGrammar.hpp.

```
6.181.4.17 template<typename Iterator> qi::rule<Iterator, Spatial-
    MeasureCollectionAttribute(), qi::space_type>
    multiscale::verification::SpatialMeasureCollectionGrammar< Iterator
    >::spatialMeasureCollectionRule [private]
```

The rule for parsing a spatial measure collection

Definition at line 66 of file SpatialMeasureCollectionGrammar.hpp.

```
6.181.4.18 template<typename Iterator> qi::rule<Iterator, SpatialMeasureAttribute(),
    qi::space_type> multiscale::verification::SpatialMeasure-
    CollectionGrammar< Iterator >::spatialMeasureRule
    [private]
```

The rule for parsing a spatial measure

Definition at line 80 of file SpatialMeasureCollectionGrammar.hpp.

```
6.181.4.19 template<typename Iterator> SpatialMeasureTypeParser
multiscale::verification::SpatialMeasureCollectionGrammar< Iterator
>::spatialMeasureTypeParser [private]
```

The spatial measure type parser

Definition at line 135 of file SpatialMeasureCollectionGrammar.hpp.

```
6.181.4.20 template<typename Iterator> SubsetOperationTypeParser
multiscale::verification::SpatialMeasureCollectionGrammar< Iterator
>::subsetOperationTypeParser [private]
```

The subset operation type parser

Definition at line 132 of file SpatialMeasureCollectionGrammar.hpp.

```
6.181.4.21 template<typename Iterator> qi::rule<Iterator, SubsetAttribute(), qi::space_
type> multiscale::verification::SpatialMeasureCollectionGrammar<
Iterator >::subsetRule [private]
```

The rule for parsing a subset

Definition at line 83 of file SpatialMeasureCollectionGrammar.hpp.

```
6.181.4.22 template<typename Iterator> qi::rule<Iterator, SubsetSpecificAttribute(),
qi::space_type> multiscale::verification::SpatialMeasure-
CollectionGrammar< Iterator >::subsetSpecificRule
[private]
```

The rule for parsing a specific subset

Definition at line 85 of file SpatialMeasureCollectionGrammar.hpp.

```
6.181.4.23 template<typename Iterator> SubsetSpecificTypeParser
multiscale::verification::SpatialMeasureCollectionGrammar< Iterator
>::subsetSpecificTypeParser [private]
```

The subset specific type parser

Definition at line 130 of file SpatialMeasureCollectionGrammar.hpp.

```
6.181.4.24 template<typename Iterator> qi::rule<Iterator,
SubsetSubsetOperationAttribute(), qi::space_type>
multiscale::verification::SpatialMeasureCollectionGrammar< Iterator
>::subsetSubsetOperationRule [private]
```

The rule for parsing a subset subset operation

Definition at line 89 of file SpatialMeasureCollectionGrammar.hpp.

**6.181.4.25 template<typename Iterator> qi::rule<Iterator, UnaryNumericFilterAttribute(), qi::space\_type> multiscale::verification::SpatialMeasureCollectionGrammar< Iterator >::unaryNumericFilterRule [private]**

The rule for parsing a unary numeric filter measure

Definition at line 110 of file SpatialMeasureCollectionGrammar.hpp.

**6.181.4.26 template<typename Iterator> UnaryNumericMeasureGrammar<Iterator> multiscale::verification::SpatialMeasureCollectionGrammar< Iterator >::unaryNumericMeasureRule [private]**

The grammar for parsing unary numeric measures

Definition at line 50 of file SpatialMeasureCollectionGrammar.hpp.

**6.181.4.27 template<typename Iterator> qi::rule<Iterator, UnaryNumericSpatialAttribute(), qi::space\_type> multiscale::verification::SpatialMeasureCollectionGrammar< Iterator >::unaryNumericSpatialMeasureCollectionRule [private]**

The rule for parsing a unary numeric spatial measure collection

Definition at line 73 of file SpatialMeasureCollectionGrammar.hpp.

**6.181.4.28 template<typename Iterator> qi::rule<Iterator, UnaryScaleAndSubsystemConstraintAttribute(), qi::space\_type> multiscale::verification::SpatialMeasureCollectionGrammar< Iterator >::unaryScaleAndSubsystemConstraintRule [private]**

The rule for parsing a unary scale and subsystem constraint

Definition at line 103 of file SpatialMeasureCollectionGrammar.hpp.

**6.181.4.29 template<typename Iterator> qi::rule<Iterator, UnarySpatialConstraintAttribute(), qi::space\_type> multiscale::verification::SpatialMeasureCollectionGrammar< Iterator >::unarySpatialConstraintRule [private]**

The rule for parsing a unary spatial constraint

Definition at line 100 of file SpatialMeasureCollectionGrammar.hpp.

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

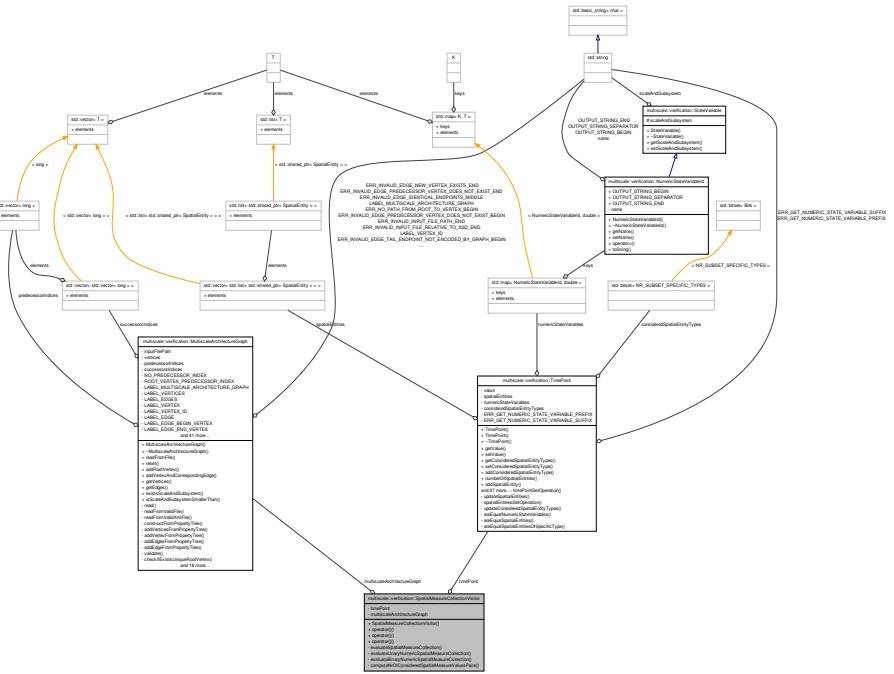
- [SpatialMeasureCollectionGrammar.hpp](#)
- [SpatialMeasureCollectionGrammarDefinition.hpp](#)

## 6.182 multiscale::verification::SpatialMeasureCollectionVisitor - Class Reference

Class for evaluating spatial measure collections.

```
#include <SpatialMeasureCollectionVisitor.hpp>
```

## Collaboration diagram for multiscale::verification::SpatialMeasureCollectionVisitor:



## Public Member Functions

- `SpatialMeasureCollectionVisitor` (`TimePoint &timePoint, const MultiscaleArchitectureGraph &multiscaleArchitectureGraph`)
  - `std::vector< double > operator()` (`const PrimarySpatialMeasureCollectionAttribute &primarySpatialMeasureCollection`) const
    - Overloading the "(" operator for the `PrimarySpatialMeasureCollectionAttribute` alternative.*
  - `std::vector< double > operator()` (`const UnaryNumericSpatialAttribute &unaryNumericSpatialMeasureCollection`) const
    - Overloading the "(" operator for the `UnaryNumericSpatialAttribute` alternative.*
  - `std::vector< double > operator()` (`const BinaryNumericSpatialAttribute &binaryNumericSpatialMeasureCollection`) const
    - Overloading the "(" operator for the `BinaryNumericSpatialAttribute` alternative.*

## **6.182 multiscale::verification::SpatialMeasureCollectionVisitor Class Reference**

---

### **Private Member Functions**

- std::vector< double > [\*\*evaluateSpatialMeasureCollection\*\*](#) (const **SpatialMeasureCollectionAttribute** &**spatialMeasureCollection**) const  
*Evaluate the given spatial measure collection.*
- std::vector< double > [\*\*evaluateUnaryNumericSpatialMeasureCollection\*\*](#) (const **UnaryNumericMeasureAttribute** &**unaryNumericMeasure**, const std::vector< double > &**spatialMeasureValues**) const  
*Evaluate the given unary numeric measure for each spatial measure value.*
- std::vector< double > [\*\*evaluateBinaryNumericSpatialMeasureCollection\*\*](#) (const **BinaryNumericMeasureAttribute** &**binaryNumericMeasure**, const std::vector< double > &**firstSpatialMeasureValuesCollection**, const std::vector< double > &**secondSpatialMeasureValuesCollection**) const  
*Evaluate the given binary numeric measure for each pair of spatial measure values.*
- std::size\_t [\*\*computeNrOfConsideredSpatialMeasureValuesPairs\*\*](#) (const std::vector< double > &**firstSpatialMeasureValuesCollection**, const std::vector< double > &**secondSpatialMeasureValuesCollection**) const  
*Compute the number of available spatial measure values pairs.*

### **Private Attributes**

- **TimePoint** & **timePoint**
- const **MultiscaleArchitectureGraph** & **multiscaleArchitectureGraph**

### **6.182.1 Detailed Description**

Class for evaluating spatial measure collections.

Definition at line 19 of file SpatialMeasureCollectionVisitor.hpp.

### **6.182.2 Constructor & Destructor Documentation**

#### **6.182.2.1 multiscale::verification::SpatialMeasureCollectionVisitor:: SpatialMeasureCollectionVisitor ( TimePoint & timePoint, const MultiscaleArchitectureGraph & multiscaleArchitectureGraph ) [inline]**

Definition at line 31 of file SpatialMeasureCollectionVisitor.hpp.

### **6.182.3 Member Function Documentation**

#### **6.182.3.1 std::size\_t multiscale::verification::SpatialMeasureCollectionVisitor:: ::computeNrOfConsideredSpatialMeasureValuesPairs ( const std::vector< double > & firstSpatialMeasureValuesCollection, const std::vector< double > & secondSpatialMeasureValuesCollection ) const [inline, private]**

Compute the number of available spatial measure values pairs.

The number of considered pairs is equal to the minimum number of values in the provided collections.

#### Parameters

|                                                 |                                                 |
|-------------------------------------------------|-------------------------------------------------|
| <i>firstSpatial-Measure-Values-Collection</i>   | The first collection of spatial measure values  |
| <i>second-Spatial-Measure-Values-Collection</i> | The second collection of spatial measure values |

Definition at line 201 of file SpatialMeasureCollectionVisitor.hpp.

Referenced by evaluateBinaryNumericSpatialMeasureCollection().

```
6.182.3.2 std::vector<double> multiscale::verification::SpatialMeasureCollection-
    Visitor::evaluateBinaryNumericSpatialMeasureCollection ( const
        BinaryNumericMeasureAttribute & binaryNumericMeasure, const std::vector<
            double > & firstSpatialMeasureValuesCollection, const std::vector< double > &
            secondSpatialMeasureValuesCollection ) const [inline, private]
```

Evaluate the given binary numeric measure for each pair of spatial measure values.

#### Parameters

|                                                 |                                                 |
|-------------------------------------------------|-------------------------------------------------|
| <i>binary-Numeric-Measure</i>                   | The provided binary numeric measure             |
| <i>firstSpatial-Measure-Values-Collection</i>   | The first collection of spatial measure values  |
| <i>second-Spatial-Measure-Values-Collection</i> | The second collection of spatial measure values |

Definition at line 165 of file SpatialMeasureCollectionVisitor.hpp.

References multiscale::verification::BinaryNumericMeasureAttribute::binaryNumericMeasureType, computeNrOfConsideredSpatialMeasureValuesPairs(), and multiscale::verification::NumericEvaluator::evaluate().

Referenced by operator()().

## **6.182 multiscale::verification::SpatialMeasureCollectionVisitor Class Reference**

```
6.182.3.3 std::vector<double> multiscale::verification::SpatialMeasure-
CollectionVisitor::evaluateSpatialMeasureCollection ( const
SpatialMeasureCollectionAttribute & spatialMeasureCollection ) const
[inline, private]
```

Evaluate the given spatial measure collection.

The timepoint and multiscale architecture graph provided in the constructor are employed during the evaluation of the spatial measure collection.

### Parameters

|                                                         |                                      |
|---------------------------------------------------------|--------------------------------------|
| <i>spatial-</i><br><i>Measure-</i><br><i>Collection</i> | The given spatial measure collection |
|---------------------------------------------------------|--------------------------------------|

Definition at line 122 of file SpatialMeasureCollectionVisitor.hpp.

References multiscale::verification::NumericMeasureCollectionEvaluator::evaluateSpatialMeasureCollection(), multiscaleArchitectureGraph, and timePoint.

Referenced by operator()().

```
6.182.3.4 std::vector<double> multiscale::verification::SpatialMeasureCollection-
Visitor::evaluateUnaryNumericSpatialMeasureCollection ( const
UnaryNumericMeasureAttribute & unaryNumericMeasure, const std::vector<
double > & spatialMeasureValues ) const [inline, private]
```

Evaluate the given unary numeric measure for each spatial measure value.

### Parameters

|                                                     |                                          |
|-----------------------------------------------------|------------------------------------------|
| <i>unary-</i><br><i>Numeric-</i><br><i>Measure</i>  | The provided unary numeric measure       |
| <i>spatial-</i><br><i>Measure-</i><br><i>Values</i> | The collection of spatial measure values |

Definition at line 139 of file SpatialMeasureCollectionVisitor.hpp.

References multiscale::verification::NumericEvaluator::evaluate(), and multiscale::verification::UnaryNumericMeasureAttribute::unaryNumericMeasureType.

Referenced by operator()().

---

```
6.182.3.5 std::vector<double> multiscale::verification::SpatialMeasureCollectionVisitor-
::operator() ( const PrimarySpatialMeasureCollectionAttribute &
primarySpatialMeasureCollection ) const [inline]
```

Overloading the "(") operator for the [PrimarySpatialMeasureCollectionAttribute](#) alternative.

#### Parameters

|                                                                            |                                        |
|----------------------------------------------------------------------------|----------------------------------------|
| <i>primary-</i><br><i>Spatial-</i><br><i>Measure-</i><br><i>Collection</i> | The primary spatial measure collection |
|----------------------------------------------------------------------------|----------------------------------------|

Definition at line 41 of file `SpatialMeasureCollectionVisitor.hpp`.

References `multiscale::verification::TimePointEvaluator::getSpatialMeasureValues()`, `multiscaleArchitectureGraph`, `multiscale::verification::PrimarySpatialMeasureCollectionAttribute::spatialMeasure`, `multiscale::verification::SpatialMeasureAttribute::spatialMeasureType`, `multiscale::verification::PrimarySpatialMeasureCollectionAttribute::subset`, `multiscale::verification::SubsetAttribute::subset`, and `timePoint`.

---

```
6.182.3.6 std::vector<double> multiscale::verification::SpatialMeasureCollection-
Visitor::operator() ( const UnaryNumericSpatialAttribute &
unaryNumericSpatialMeasureCollection ) const [inline]
```

Overloading the "(") operator for the [UnaryNumericSpatialAttribute](#) alternative.

#### Parameters

|                                                                                             |                                                                                   |
|---------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------|
| <i>unary-</i><br><i>Numeric-</i><br><i>Spatial-</i><br><i>Measure-</i><br><i>Collection</i> | The attribute containing a unary numeric measure and a spatial measure collection |
|---------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------|

Definition at line 70 of file `SpatialMeasureCollectionVisitor.hpp`.

References `evaluateSpatialMeasureCollection()`, `evaluateUnaryNumericSpatialMeasureCollection()`, `multiscale::verification::UnaryNumericSpatialAttribute::spatialMeasureCollection`, and `multiscale::verification::UnaryNumericSpatialAttribute::unaryNumericMeasure`.

---

```
6.182.3.7 std::vector<double> multiscale::verification::SpatialMeasureCollection-
Visitor::operator() ( const BinaryNumericSpatialAttribute &
binaryNumericSpatialMeasureCollection ) const [inline]
```

Overloading the "(") operator for the [BinaryNumericSpatialAttribute](#) alternative.

**Parameters**

|                                                  |                                                                                       |
|--------------------------------------------------|---------------------------------------------------------------------------------------|
| <i>binary-Numeric-Spatial-Measure-Collection</i> | The attribute containing a binary numeric measure and two spatial measure collections |
|--------------------------------------------------|---------------------------------------------------------------------------------------|

Definition at line 92 of file SpatialMeasureCollectionVisitor.hpp.

References multiscale::verification::BinaryNumericSpatialAttribute::binaryNumericMeasure, evaluateBinaryNumericSpatialMeasureCollection(), evaluateSpatialMeasureCollection(), multiscale::verification::BinaryNumericSpatialAttribute::firstSpatialMeasureCollection, and multiscale::verification::BinaryNumericSpatialAttribute::secondSpatialMeasureCollection.

#### 6.182.4 Member Data Documentation

**6.182.4.1 const MultiscaleArchitectureGraph& multiscale::verification::SpatialMeasureCollectionVisitor::multiscaleArchitectureGraph [private]**

The considered multiscale architecture graph

Definition at line 26 of file SpatialMeasureCollectionVisitor.hpp.

Referenced by evaluateSpatialMeasureCollection(), and operator()().

**6.182.4.2 TimePoint& multiscale::verification::SpatialMeasureCollectionVisitor::timePoint [private]**

The considered timepoint

Definition at line 24 of file SpatialMeasureCollectionVisitor.hpp.

Referenced by evaluateSpatialMeasureCollection(), and operator()().

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

- SpatialMeasureCollectionVisitor.hpp

## 6.183 multiscale::verification::SpatialMeasureEvaluator Class Reference

Class for evaluating spatial measures.

```
#include <SpatialMeasureEvaluator.hpp>
```

## Static Public Member Functions

- static double [evaluate](#) (const **SpatialEntity** &*spatialEntity*, const **SpatialMeasureType** &*type*)

*Return the value of the spatial measure for the given spatial entity.*

### 6.183.1 Detailed Description

Class for evaluating spatial measures.

Definition at line 13 of file **SpatialMeasureEvaluator.hpp**.

### 6.183.2 Member Function Documentation

- 6.183.2.1 static double multiscale::verification::SpatialMeasureEvaluator::evaluate**  
 ( const **SpatialEntity** & *spatialEntity*, const **SpatialMeasureType** & *type* )  
 [inline, static]

Return the value of the spatial measure for the given spatial entity.

#### Parameters

|                      |                                 |
|----------------------|---------------------------------|
| <i>spatialEntity</i> | The given spatial entity        |
| <i>type</i>          | The type of the spatial measure |

Definition at line 22 of file **SpatialMeasureEvaluator.hpp**.

References **multiscale::verification::SpatialEntity::getSpatialMeasureValue()**, and **multiscale::verification::spatialmeasure::validateSpatialMeasureType()**.

Referenced by **multiscale::verification::TimePointEvaluator::getSpatialMeasureValues()**, and **multiscale::verification::FilterNumericVisitor::operator()**.

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

- **SpatialMeasureEvaluator.hpp**

## 6.184 multiscale::verification::SpatialMeasureTypeParser Struct Reference

Symbol table and parser for the spatial measure type.

```
#include <SymbolTablesAutoGenerated.hpp>
```

### Public Member Functions

- [SpatialMeasureTypeParser \(\)](#)

### 6.184.1 Detailed Description

Symbol table and parser for the spatial measure type.

Definition at line 28 of file SymbolTablesAutoGenerated.hpp.

### 6.184.2 Constructor & Destructor Documentation

#### 6.184.2.1 multiscale::verification::SpatialMeasureTypeParser::SpatialMeasureTypeParser( ) [inline]

Definition at line 30 of file SymbolTablesAutoGenerated.hpp.

References multiscale::verification::Angle, multiscale::verification::CentroidX, multiscale::verification::Density, multiscale::verification::Perimeter, and multiscale::verification::RectangleMeasure.

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

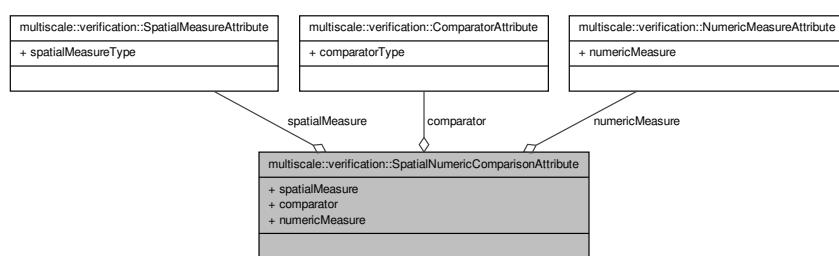
- SymbolTablesAutoGenerated.hpp

## 6.185 multiscale::verification::SpatialNumericComparisonAttribute Class Reference

Class for representing a spatial numeric comparison attribute.

```
#include <SpatialNumericComparisonAttribute.hpp>
```

Collaboration diagram for multiscale::verification::SpatialNumericComparisonAttribute:



### Public Attributes

- [SpatialMeasureAttribute spatialMeasure](#)
- [ComparatorAttribute comparator](#)
- [NumericMeasureAttribute numericMeasure](#)

### 6.185.1 Detailed Description

Class for representing a spatial numeric comparison attribute.

Definition at line 19 of file SpatialNumericComparisonAttribute.hpp.

### 6.185.2 Member Data Documentation

#### 6.185.2.1 ComparatorAttribute multiscale::verification::SpatialNumericComparisonAttribute::comparator

The comparator

Definition at line 24 of file SpatialNumericComparisonAttribute.hpp.

#### 6.185.2.2 NumericMeasureAttribute multiscale::verification::SpatialNumericComparisonAttribute::numericMeasure

The numeric measure

Definition at line 25 of file SpatialNumericComparisonAttribute.hpp.

#### 6.185.2.3 SpatialMeasureAttribute multiscale::verification::SpatialNumericComparisonAttribute::spatialMeasure

The spatial measure

Definition at line 23 of file SpatialNumericComparisonAttribute.hpp.

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

- SpatialNumericComparisonAttribute.hpp

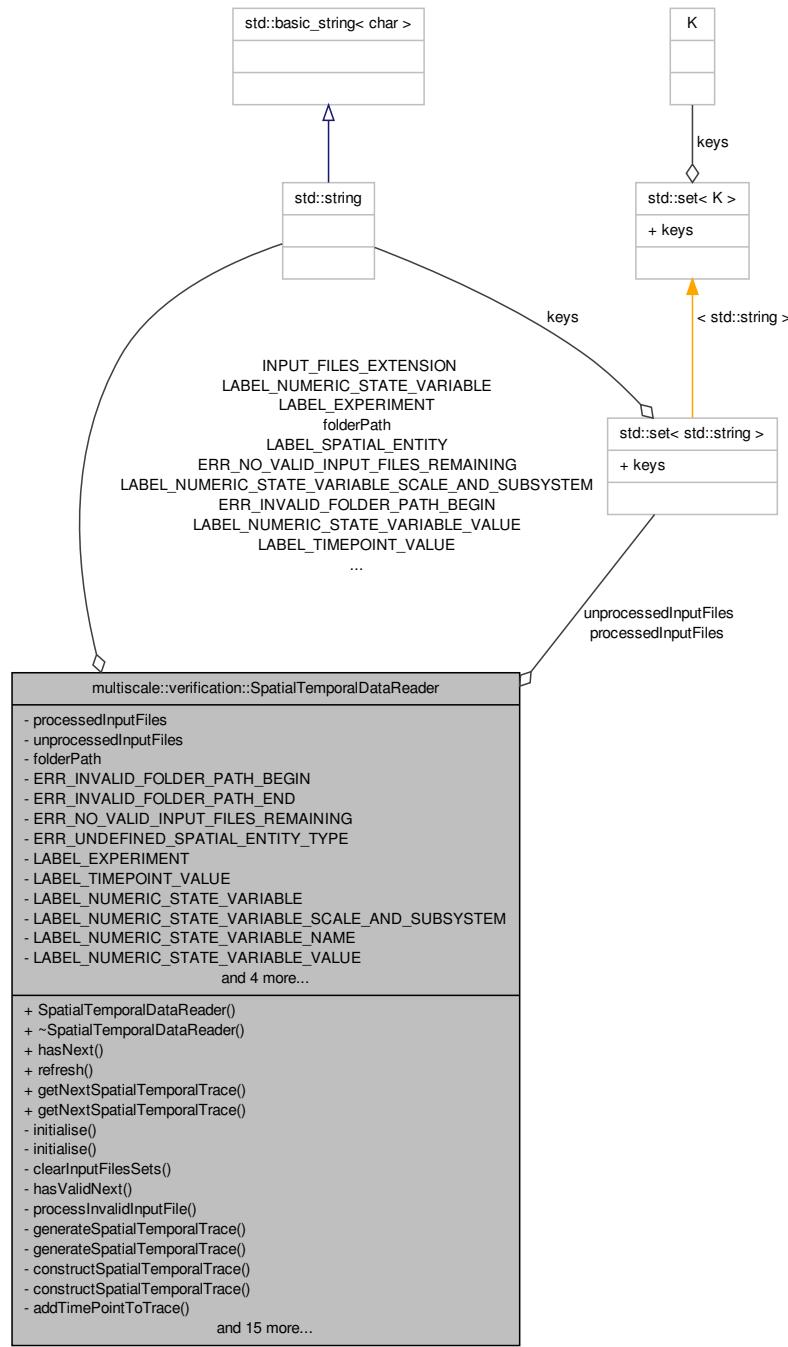
## 6.186 multiscale::verification::SpatialTemporalDataReader Class - Reference

Class for reading spatial temporal trace data from input files.

```
#include <SpatialTemporalDataReader.hpp>
```

## 6.186 multiscale::verification::SpatialTemporalDataReader Class Reference 1013

Collaboration diagram for multiscale::verification::SpatialTemporalDataReader:



## Public Member Functions

- `SpatialTemporalDataReader (const std::string &folderPath)`
- `~SpatialTemporalDataReader ()`
- `bool hasNext ()`  
*Check if there are any remaining valid unprocessed traces in the given folder.*
- `void refresh ()`  
*Refresh the sets of processed and unprocessed traces' input files considering the given folder.*
- `SpatialTemporalTrace getNextSpatialTemporalTrace ()`  
*Return the next spatial temporal trace.*
- `SpatialTemporalTrace getNextSpatialTemporalTrace (std::string &tracePath)`  
*Return the next spatial temporal trace and its path.*

## Private Member Functions

- `void initialise (const std::string &folderPath)`  
*Initialise the sets for storing processed and unprocessed input files.*
- `void initialise ()`  
*Initialise the sets for storing processed and unprocessed input files.*
- `void clearInputFilesSets ()`  
*Clear the contents of the sets of processed and unprocessed input files.*
- `bool hasValidNext ()`  
*Check if there are any remaining valid unprocessed traces in the given folder.*
- `std::set< std::string >::iterator processInvalidInputFile (const std::set< std::string >::iterator &invalidInputFileIterator)`  
*Process the invalid input file to which the given iterator points.*
- `SpatialTemporalTrace generateSpatialTemporalTrace ()`  
*Generate the spatial temporal trace corresponding to the first valid unprocessed input file.*
- `SpatialTemporalTrace generateSpatialTemporalTrace (std::string &tracePath)`  
*Generate the spatial temporal trace corresponding to the first valid unprocessed input file.*
- `SpatialTemporalTrace constructSpatialTemporalTrace (const std::string &inputFilepath)`  
*Construct the spatial temporal trace corresponding to the first valid unprocessed input file.*
- `SpatialTemporalTrace constructSpatialTemporalTrace (const pt::ptree &tree)`  
*Construct the spatial temporal trace corresponding to the given property tree.*
- `void addTimePointToTrace (const pt::ptree &timePointTree, SpatialTemporalTrace &trace)`  
*Add a timepoint corresponding to the given property tree to the spatial temporal trace.*
- `void convertTimePointPropertyTreeToTrace (const pt::ptree &timePointTree, - TimePoint &timePoint)`  
*Convert a time point from a property tree to a timepoint representation.*

- void `setTimePointValue` (const `pt::ptree` &timePointTree, `TimePoint` &timePoint)  
*Set the value of the timepoint considering the given timepoint tree.*
- bool `timePointHasValue` (const `pt::ptree` &propertyTree, unsigned long &value)  
*Check if the provided property tree contains the attribute "value".*
- void `addEntitiesToTimePoint` (const `pt::ptree` &timePointTree, `TimePoint` &timePoint)  
*Add the numeric state variables and spatial entities contained by the property tree to the given timepoint.*
- void `addNumericStateVariableToTimePoint` (const `pt::ptree` &numericStateVariableTree, `TimePoint` &timePoint)  
*Add the numeric state variable (provided as a tree) to the provided timepoint.*
- void `addSpatialEntityToTimePoint` (const `pt::ptree` &spatialEntityTree, `TimePoint` &timePoint)  
*Add the spatial entity contained by the property tree to the given timePoint.*
- void `createDerivedSpatialEntity` (const `pt::ptree` &spatialEntityTree, std::shared\_ptr< `SpatialEntity` > &spatialEntity, `SubsetSpecificType` &spatialEntityType)  
*Create a derived spatial entity considering the type specified in the given tree.*
- void `setSpatialEntityScaleAndSubsystem` (const `pt::ptree` &spatialEntityTree, const std::shared\_ptr< `SpatialEntity` > &spatialEntity)  
*Initialise the spatial entity scale and subsystem using the given spatialEntityTree.*
- void `setSpatialEntityMeasureValues` (const `pt::ptree` &spatialEntityTree, const std::shared\_ptr< `SpatialEntity` > &spatialEntity)  
*Initialise the spatial entity measure values using the given spatialEntityTree.*
- std::set< std::string >::iterator `getRandomValidUnprocessedInputFilepath` ()  
*Get an iterator pointing to a random valid unprocessed input file.*
- std::set< std::string >::iterator `getRandomUnprocessedInputFile` ()  
*Get an iterator pointing to a random unprocessed input file.*
- void `processValidInputFile` (const std::set< std::string >::iterator &validInputFileIterator)  
*Process the valid input file to which the given iterator points.*
- void `updateInputFilesSets` ()  
*Update the sets of processed and unprocessed files by checking if the folder contents have been updated.*
- std::vector< std::string > `getFilesInFolder` ()  
*Get the collection of files stored in the input folder.*
- bool `isValidInputFile` (const std::string &inputFilepath)  
*Check if the given input file is valid.*
- void `validateFolderPath` (const std::string &folderPath)  
*Check if the given folder path is valid.*

## Private Attributes

- std::set< std::string > `processedInputFiles`
- std::set< std::string > `unprocessedInputFiles`
- std::string `FolderPath`

## Static Private Attributes

- static const std::string `ERR_INVALID_FOLDER_PATH_BEGIN` = "The provided path ("
- static const std::string `ERR_INVALID_FOLDER_PATH_END` = ") does not point to a folder. Please change."
- static const std::string `ERR_NO_VALID_INPUT_FILES_REMAINING` = "There are no valid unprocessed input files remaining."
- static const std::string `ERR_UNDEFINED_SPATIAL_ENTITY_TYPE` = "The provided spatial entity type is invalid."
- static const std::string `LABEL_EXPERIMENT` = "experiment"
- static const std::string `LABEL_TIMEPOINT_VALUE` = "<xmattr>.value"
- static const std::string `LABEL_NUMERIC_STATE_VARIABLE` = "numericStateVariable"
- static const std::string `LABEL_NUMERIC_STATE_VARIABLE_SCALE_AND_SUBSYSTEM` = "<xmattr>.scaleAndSubsystem"
- static const std::string `LABEL_NUMERIC_STATE_VARIABLE_NAME` = "name"
- static const std::string `LABEL_NUMERIC_STATE_VARIABLE_VALUE` = "value"
- static const std::string `LABEL_SPATIAL_ENTITY` = "spatialEntity"
- static const std::string `LABEL_SPATIAL_ENTITY_SPATIAL_TYPE` = "<xmattr>.spatialType"
- static const std::string `LABEL_SPATIAL_ENTITY_SCALE_AND_SUBSYSTEM` = "<xmattr>.scaleAndSubsystem"
- static const std::string `INPUT_FILES_EXTENSION` = ".xml"
- static const std::string `INPUT_FILES_SCHEMA_PATH` = "/usr/local/share/mule/config/verification/spatial-temporal/schema/MSTML\_L1V1.xsd"

### 6.186.1 Detailed Description

Class for reading spatial temporal trace data from input files.

Definition at line 20 of file SpatialTemporalDataReader.hpp.

### 6.186.2 Constructor & Destructor Documentation

#### 6.186.2.1 `SpatialTemporalDataReader::SpatialTemporalDataReader ( const std::string & folderPath )`

Definition at line 17 of file SpatialTemporalDataReader.cpp.

References initialise().

#### 6.186.2.2 `SpatialTemporalDataReader::~SpatialTemporalDataReader ( )`

Definition at line 21 of file SpatialTemporalDataReader.cpp.

References processedInputFiles, and unprocessedInputFiles.

### 6.186.3 Member Function Documentation

**6.186.3.1 void SpatialTemporalDataReader::addEntitiesToTimePoint ( const pt::ptree & *timePointTree*, TimePoint & *timePoint* ) [private]**

Add the numeric state variables and spatial entities contained by the property tree to the given timepoint.

#### Parameters

|                       |                         |
|-----------------------|-------------------------|
| <i>timePoint-Tree</i> | The given property tree |
| <i>timePoint</i>      | The given timepoint     |

Definition at line 185 of file SpatialTemporalDataReader.cpp.

References addNumericStateVariableToTimePoint(), addSpatialEntityToTimePoint(), LABEL\_NUMERIC\_STATE\_VARIABLE, and LABEL\_SPATIAL\_ENTITY.

Referenced by convertTimePointPropertyTreeToTrace().

**6.186.3.2 void SpatialTemporalDataReader::addNumericStateVariableToTimePoint ( const pt::ptree & *numericStateVariableTree*, TimePoint & *timePoint* ) [private]**

Add the numeric state variable (provided as a tree) to the provided timepoint.

#### Parameters

|                                   |                                                   |
|-----------------------------------|---------------------------------------------------|
| <i>numeric-State-VariableTree</i> | The provided numeric state variable property tree |
| <i>timePoint</i>                  | The given timepoint                               |

Definition at line 197 of file SpatialTemporalDataReader.cpp.

References multiscale::verification::TimePoint::addNumericStateVariable(), multiscale::verification::ScaleAndSubsystem::DEFAULT\_VALUE, LABEL\_NUMERIC\_STATE\_VARIABLE\_NAME, LABEL\_NUMERIC\_STATE\_VARIABLE\_SCALE\_AND\_SUBSYSTEM, and LABEL\_NUMERIC\_STATE\_VARIABLE\_VALUE.

Referenced by addEntitiesToTimePoint().

**6.186.3.3 void SpatialTemporalDataReader::addSpatialEntityToTimePoint ( const pt::ptree & *spatialEntityTree*, TimePoint & *timePoint* ) [private]**

Add the spatial entity contained by the property tree to the given timePoint.

**Parameters**

|                           |                                                         |
|---------------------------|---------------------------------------------------------|
| <i>spatialEntity-Tree</i> | The given spatial entity represented as a property tree |
| <i>timePoint</i>          | The given timepoint                                     |

Definition at line 215 of file SpatialTemporalDataReader.cpp.

References multiscale::verification::TimePoint::addSpatialEntityAndType(), createDerivedSpatialEntity(), setSpatialEntityMeasureValues(), and setSpatialEntityScaleAndSubsystem().

Referenced by addEntitiesToTimePoint().

#### 6.186.3.4 void SpatialTemporalDataReader::addTimePointToTrace ( const pt::ptree & *timePointTree*, SpatialTemporalTrace & *trace* ) [private]

Add a timepoint corresponding to the given property tree to the spatial temporal trace.

**Parameters**

|                       |                                                  |
|-----------------------|--------------------------------------------------|
| <i>timePoint-Tree</i> | The property tree corresponding to the timepoint |
| <i>trace</i>          | The spatial temporal trace                       |

Definition at line 143 of file SpatialTemporalDataReader.cpp.

References multiscale::verification::SpatialTemporalTrace::addTimePoint(), and convertTimePointPropertyTreeToTrace().

Referenced by constructSpatialTemporalTrace().

#### 6.186.3.5 void SpatialTemporalDataReader::clearInputFilesSets ( ) [private]

Clear the contents of the sets of processed and unprocessed input files.

Definition at line 70 of file SpatialTemporalDataReader.cpp.

References processedInputFiles, and unprocessedInputFiles.

Referenced by initialise().

#### 6.186.3.6 SpatialTemporalTrace SpatialTemporalDataReader::construct-SpatialTemporalTrace ( const std::string & *inputFilepath* ) [private]

Construct the spatial temporal trace corresponding to the first valid unprocessed input file.

The unprocessed input file will be processed and returned as a property tree.

## **6.186 multiscale::verification::SpatialTemporalDataReader Class Reference 1019**

### Parameters

|                      |                                       |
|----------------------|---------------------------------------|
| <i>inputFilepath</i> | The valid unprocessed input file path |
|----------------------|---------------------------------------|

Definition at line 123 of file SpatialTemporalDataReader.cpp.

Referenced by generateSpatialTemporalTrace().

**6.186.3.7 SpatialTemporalTrace SpatialTemporalDataReader-  
::constructSpatialTemporalTrace ( const pt::ptree & tree )  
[private]**

Construct the spatial temporal trace corresponding to the given property tree.

Definition at line 132 of file SpatialTemporalDataReader.cpp.

References addTimePointToTrace(), and LABEL\_EXPERIMENT.

**6.186.3.8 void SpatialTemporalDataReader::convertTimePointProperty-  
TreeToTrace ( const pt::ptree & timePointTree, TimePoint & timePoint )  
[private]**

Convert a time point from a property tree to a timepoint representation.

### Parameters

|                            |                                                               |
|----------------------------|---------------------------------------------------------------|
| <i>timePoint-<br/>Tree</i> | Property tree representation of the timepoint                 |
| <i>timePoint</i>           | The <a href="#">TimePoint</a> representation of the timepoint |

Definition at line 152 of file SpatialTemporalDataReader.cpp.

References addEntitiesToTimePoint(), and setTimePointValue().

Referenced by addTimePointToTrace().

**6.186.3.9 void SpatialTemporalDataReader::createDerivedSpatialEntity ( const  
pt::ptree & spatialEntityTree, std::shared\_ptr< [SpatialEntity](#) > & spatialEntity,  
SubsetSpecificType & spatialEntityType ) [private]**

Create a derived spatial entity considering the type specified in the given tree.

### Parameters

|                                |                                                         |
|--------------------------------|---------------------------------------------------------|
| <i>spatialEntity-<br/>Tree</i> | The given spatial entity represented as a property tree |
| <i>spatialEntity</i>           | The created spatial entity                              |
| <i>spatialEntityType</i>       | The derived type of the spatial entity                  |

Definition at line 22 of file SpatialTemporalDataReaderAutoGenerated.cpp.

References multiscale::verification::Clusters, ERR\_UNDEFINED\_SPATIAL\_ENTITY\_TYPE, LABEL\_SPATIAL\_ENTITY\_SPATIAL\_TYPE, and multiscale::verification::Regions.

Referenced by addSpatialEntityToTimePoint().

#### 6.186.3.10 SpatialTemporalTrace SpatialTemporalDataReader::generateSpatialTemporalTrace( ) [private]

Generate the spatial temporal trace corresponding to the first valid unprocessed input file.

The unprocessed input file will be moved to the set of processed input files after creating the spatial temporal trace.

Definition at line 101 of file SpatialTemporalDataReader.cpp.

References constructSpatialTemporalTrace(), getRandomValidUnprocessedInputFilepath(), and processValidInputFile().

Referenced by getNextSpatialTemporalTrace().

#### 6.186.3.11 SpatialTemporalTrace SpatialTemporalDataReader::generateSpatialTemporalTrace( std::string & tracePath ) [private]

Generate the spatial temporal trace corresponding to the first valid unprocessed input file.

The unprocessed input file will be moved to the set of processed input files after creating the spatial temporal trace.

The path to the trace will be returned in the tracePath output parameter.

##### Parameters

|                  |                                        |
|------------------|----------------------------------------|
| <i>tracePath</i> | The path to the spatial temporal trace |
|------------------|----------------------------------------|

Definition at line 111 of file SpatialTemporalDataReader.cpp.

References constructSpatialTemporalTrace(), getRandomValidUnprocessedInputFilepath(), and processValidInputFile().

#### 6.186.3.12 std::vector< std::string > SpatialTemporalDataReader::getFilesInFolder( ) [private]

Get the collection of files stored in the input folder.

Definition at line 292 of file SpatialTemporalDataReader.cpp.

References folderPath, and INPUT\_FILES\_EXTENSION.

Referenced by updateInputFilesSets().

## **6.186 multiscale::verification::SpatialTemporalDataReader Class Reference 1021**

---

### **6.186.3.13 SpatialTemporalTrace SpatialTemporalDataReader::getNextSpatialTemporalTrace ( )**

Return the next spatial temporal trace.

Definition at line 37 of file SpatialTemporalDataReader.cpp.

References ERR\_NO\_VALID\_INPUT\_FILES\_REMAINING, generateSpatialTemporalTrace(), and hasNext().

Referenced by multiscale::verification::MSTMLSubfilesMerger::addSubtracesToResultingTrace(), and multiscale::verification::ModelCheckingManager::getNextSpatialTemporalTrace().

### **6.186.3.14 SpatialTemporalTrace SpatialTemporalDataReader::getNextSpatialTemporalTrace ( std::string & tracePath )**

Return the next spatial temporal trace and its path.

#### **Parameters**

|                  |                                        |
|------------------|----------------------------------------|
| <i>tracePath</i> | The path to the spatial temporal trace |
|------------------|----------------------------------------|

Definition at line 46 of file SpatialTemporalDataReader.cpp.

References ERR\_NO\_VALID\_INPUT\_FILES\_REMAINING, generateSpatialTemporalTrace(), and hasNext().

### **6.186.3.15 std::set< std::string >::iterator SpatialTemporalDataReader::getRandomUnprocessedInputFile ( ) [private]**

Get an iterator pointing to a random unprocessed input file.

Definition at line 260 of file SpatialTemporalDataReader.cpp.

References unprocessedInputFiles.

Referenced by getRandomValidUnprocessedInputFilepath().

### **6.186.3.16 std::set< std::string >::iterator SpatialTemporalDataReader::getRandomValidUnprocessedInputFilepath ( ) [private]**

Get an iterator pointing to a random valid unprocessed input file.

Definition at line 240 of file SpatialTemporalDataReader.cpp.

References ERR\_NO\_VALID\_INPUT\_FILES\_REMAINING, getRandomUnprocessedInputFile(), hasNext(), isValidInputFile(), and processInvalidInputFile().

Referenced by generateSpatialTemporalTrace().

**6.186.3.17 bool SpatialTemporalDataReader::hasNext( )**

Check if there are any remaining valid unprocessed traces in the given folder.

This method does not automatically refresh the sets of input files.

Definition at line 27 of file SpatialTemporalDataReader.cpp.

References hasValidNext().

Referenced by multiscale::verification::MSTMLSubfilesMerger::addSubtracesToResultingTrace(), getNextSpatialTemporalTrace(), getRandomValidUnprocessedInputFilepath(), and multiscale::verification::ModelCheckingManager::runModelCheckersForCurrentlyExistingTraces().

**6.186.3.18 bool SpatialTemporalDataReader::hasValidNext( ) [private]**

Check if there are any remaining valid unprocessed traces in the given folder.

Definition at line 76 of file SpatialTemporalDataReader.cpp.

References isValidInputFile(), processInvalidInputFile(), and unprocessedInputFiles.

Referenced by hasNext().

**6.186.3.19 void SpatialTemporalDataReader::initialise( const std::string & *FolderPath* ) [private]**

Initialise the sets for storing processed and unprocessed input files.

**Parameters**

|                   |                          |
|-------------------|--------------------------|
| <i>FolderPath</i> | Path to the input folder |
|-------------------|--------------------------|

Definition at line 55 of file SpatialTemporalDataReader.cpp.

References folderPath, initialise(), and validateFolderPath().

**6.186.3.20 void SpatialTemporalDataReader::initialise( ) [private]**

Initialise the sets for storing processed and unprocessed input files.

Definition at line 64 of file SpatialTemporalDataReader.cpp.

References clearInputFilesSets(), and updateInputFilesSets().

Referenced by initialise(), and SpatialTemporalDataReader().

**6.186.3.21 bool SpatialTemporalDataReader::isValidInputFile( const std::string & *inputfilepath* ) [private]**

Check if the given input file is valid.

## **6.186 multiscale::verification::SpatialTemporalDataReader Class Reference 1023**

An input file is valid if it is an xml file which conforms to the formal specification given in the xml schema (xsd file).

**WARNING:** The Timepoint class contains as members lists of spatial entities because the uniqueness of the spatial entities is determined using this method. If this method is no longer used then replace the lists in the Timepoint class with sets or unordered\_sets in order to ensure the uniqueness of the elements.

### **Parameters**

|                      |                            |
|----------------------|----------------------------|
| <i>inputFilepath</i> | The path to the input file |
|----------------------|----------------------------|

Definition at line 297 of file SpatialTemporalDataReader.cpp.

References INPUT\_FILES\_SCHEMA\_PATH, and multiscale::XmlValidator::isValidXmlFile().

Referenced by getRandomValidUnprocessedInputFilepath(), and hasValidNext().

**6.186.3.22 std::set< std::string >::iterator SpatialTemporalDataReader::processInvalidInputFile ( const std::set< std::string >::iterator & *invalidInputFileIterator* ) [private]**

Process the invalid input file to which the given iterator points.

The iterator corresponds to a position in the list of unprocessed input files

### **Parameters**

|                                 |                                                      |
|---------------------------------|------------------------------------------------------|
| <i>invalidInputFileIterator</i> | The iterator pointing to the invalid input file path |
|---------------------------------|------------------------------------------------------|

Definition at line 91 of file SpatialTemporalDataReader.cpp.

References processedInputFiles, and unprocessedInputFiles.

Referenced by getRandomValidUnprocessedInputFilepath(), and hasValidNext().

**6.186.3.23 void SpatialTemporalDataReader::processValidInputFile ( const std::set< std::string >::iterator & *validInputFileIterator* ) [private]**

Process the valid input file to which the given iterator points.

The iterator corresponds to a position in the list of unprocessed input files

### **Parameters**

|                               |                                                    |
|-------------------------------|----------------------------------------------------|
| <i>validInputFileIterator</i> | The iterator pointing to the valid input file path |
|-------------------------------|----------------------------------------------------|

Definition at line 271 of file SpatialTemporalDataReader.cpp.

References processedInputFiles, and unprocessedInputFiles.

Referenced by generateSpatialTemporalTrace().

#### 6.186.3.24 void SpatialTemporalDataReader::refresh ( )

Refresh the sets of processed and unprocessed traces' input files considering the given folder.

Definition at line 32 of file SpatialTemporalDataReader.cpp.

References updateInputFileSets().

Referenced by multiscale::verification::ModelCheckingManager::updateTraceReader().

#### 6.186.3.25 void SpatialTemporalDataReader::setSpatialEntityMeasureValues ( const pt::ptree & *spatialEntityTree*, const std::shared\_ptr< SpatialEntity > & *spatialEntity* ) [private]

Initialise the spatial entity measure values using the given spatialEntityTree.

##### Parameters

|                           |                                      |
|---------------------------|--------------------------------------|
| <i>spatialEntity-Tree</i> | The spatial entity tree              |
| <i>spatialEntity</i>      | The spatial entity to be initialised |

Definition at line 38 of file SpatialTemporalDataReaderAutoGenerated.cpp.

Referenced by addSpatialEntityToTimePoint().

#### 6.186.3.26 void SpatialTemporalDataReader::setSpatialEntityScaleAndSubsystem ( const pt::ptree & *spatialEntityTree*, const std::shared\_ptr< SpatialEntity > & *spatialEntity* ) [private]

Initialise the spatial entity scale and subsystem using the given spatialEntityTree.

If the value of the scale and subsystem is not provided in the spatialEntityTree the default empty string "" is used instead.

##### Parameters

|                           |                                      |
|---------------------------|--------------------------------------|
| <i>spatialEntity-Tree</i> | The spatial entity tree              |
| <i>spatialEntity</i>      | The spatial entity to be initialised |

Definition at line 228 of file SpatialTemporalDataReader.cpp.

References multiscale::verification::ScaleAndSubsystem::DEFAULT\_VALUE, and LABEL\_SPATIAL\_ENTITY\_SCALE\_AND\_SUBSYSTEM.

Referenced by addSpatialEntityToTimePoint().

## **6.186 multiscale::verification::SpatialTemporalDataReader Class Reference 1025**

**6.186.3.27 void SpatialTemporalDataReader::setTimePointValue ( const pt::ptree & timePointTree, TimePoint & timePoint ) [private]**

Set the value of the timepoint considering the given timepoint tree.

### Parameters

|                       |                                                               |
|-----------------------|---------------------------------------------------------------|
| <i>timePoint-Tree</i> | Property tree representation of the timepoint                 |
| <i>timePoint</i>      | The <a href="#">TimePoint</a> representation of the timepoint |

Definition at line 159 of file SpatialTemporalDataReader.cpp.

References multiscale::verification::TimePoint::setValue(), and timePointHasValue().

Referenced by convertTimePointPropertyTreeToTrace().

**6.186.3.28 bool SpatialTemporalDataReader::timePointHasValue ( const pt::ptree & propertyTree, unsigned long & value ) [private]**

Check if the provided property tree contains the attribute "value".

### Parameters

|                     |                            |
|---------------------|----------------------------|
| <i>propertyTree</i> | The provided property tree |
| <i>value</i>        | The value (if it exists)   |

Definition at line 170 of file SpatialTemporalDataReader.cpp.

References LABEL\_TIMEPOINT\_VALUE.

Referenced by setTimePointValue().

**6.186.3.29 void SpatialTemporalDataReader::updateInputFilesSets ( ) [private]**

Update the sets of processed and unprocessed files by checking if the folder contents have been updated.

Definition at line 280 of file SpatialTemporalDataReader.cpp.

References getFilesInFolder(), processedInputFiles, and unprocessedInputFiles.

Referenced by initialise(), and refresh().

**6.186.3.30 void SpatialTemporalDataReader::validateFolderPath ( const std::string & folderPath ) [private]**

Check if the given folder path is valid.

The folder path is valid if it is a path pointing to a folder.

**Parameters**

|                   |
|-------------------|
| <i>FolderPath</i> |
|-------------------|

Definition at line 302 of file SpatialTemporalDataReader.cpp.

References `ERR_INVALID_FOLDER_PATH_BEGIN`, `ERR_INVALID_FOLDER_PATH_END`, and `multiscale::Filesystem::isValidFolderPath()`.

Referenced by `initialise()`.

#### 6.186.4 Member Data Documentation

6.186.4.1 `const std::string SpatialTemporalDataReader::ERR_INVALID_FOLDER_PATH_BEGIN = "The provided path (" [static, private]`

Definition at line 230 of file SpatialTemporalDataReader.hpp.

Referenced by `validateFolderPath()`.

6.186.4.2 `const std::string SpatialTemporalDataReader::ERR_INVALID_FOLDER_PATH_END = ") does not point to a folder. Please change." [static, private]`

Definition at line 231 of file SpatialTemporalDataReader.hpp.

Referenced by `validateFolderPath()`.

6.186.4.3 `const std::string SpatialTemporalDataReader::ERR_NO_VALID_INPUT_FILES_REMAINING = "There are no valid unprocessed input files remaining." [static, private]`

Definition at line 233 of file SpatialTemporalDataReader.hpp.

Referenced by `getNextSpatialTemporalTrace()`, and `getRandomValidUnprocessedInputFilepath()`.

6.186.4.4 `const std::string SpatialTemporalDataReader::ERR_UNDEFINED_SPATIAL_ENTITY_TYPE = "The provided spatial entity type is invalid." [static, private]`

Definition at line 234 of file SpatialTemporalDataReader.hpp.

Referenced by `createDerivedSpatialEntity()`.

6.186.4.5 `std::string multiscale::verification::SpatialTemporalDataReader::FolderPath [private]`

The path to the folder where all input files are stored

## **6.186 multiscale::verification::SpatialTemporalDataReader Class Reference 1027**

---

Definition at line 27 of file SpatialTemporalDataReader.hpp.

Referenced by getFilesInFolder(), and initialise().

**6.186.4.6 const std::string SpatialTemporalDataReader::INPUT\_FILES\_EXTENSION = ".xml" [static, private]**

Definition at line 248 of file SpatialTemporalDataReader.hpp.

Referenced by getFilesInFolder().

**6.186.4.7 const std::string SpatialTemporalDataReader::INPUT\_FILES\_SCHEMA\_PATH = "/usr/local/share/mule/config/verification/spatial-temporal/schema/MSTML1-V1.xsd" [static, private]**

Definition at line 249 of file SpatialTemporalDataReader.hpp.

Referenced by isValidInputFile().

**6.186.4.8 const std::string SpatialTemporalDataReader::LABEL\_EXPERIMENT = "experiment" [static, private]**

Definition at line 236 of file SpatialTemporalDataReader.hpp.

Referenced by constructSpatialTemporalTrace().

**6.186.4.9 const std::string SpatialTemporalDataReader::LABEL\_NUMERIC\_STATE\_VARIABLE = "numericStateVariable" [static, private]**

Definition at line 239 of file SpatialTemporalDataReader.hpp.

Referenced by addEntitiesToTimePoint().

**6.186.4.10 const std::string SpatialTemporalDataReader::LABEL\_NUMERIC\_STATE\_VARIABLE\_NAME = "name" [static, private]**

Definition at line 241 of file SpatialTemporalDataReader.hpp.

Referenced by addNumericStateVariableToTimePoint().

**6.186.4.11 const std::string SpatialTemporalDataReader::LABEL\_NUMERIC\_STATE\_VARIABLE\_SCALE\_AND\_SUBSYSTEM = "<xmattr>.scaleAndSubsystem" [static, private]**

Definition at line 240 of file SpatialTemporalDataReader.hpp.

Referenced by addNumericStateVariableToTimePoint().

---

```
6.186.4.12 const std::string SpatialTemporalDataReader::LABEL_NUMERIC_STATE_VARIABLE_VALUE = "value" [static, private]
```

Definition at line 242 of file SpatialTemporalDataReader.hpp.

Referenced by addNumericStateVariableToTimePoint().

```
6.186.4.13 const std::string SpatialTemporalDataReader::LABEL_SPATIAL_ENTITY = "spatialEntity" [static, private]
```

Definition at line 244 of file SpatialTemporalDataReader.hpp.

Referenced by addEntitiesToTimePoint().

```
6.186.4.14 const std::string SpatialTemporalDataReader::LABEL_SPATIAL_ENTITY_SCALE_AND_SUBSYSTEM = "<xmllatr>.scaleAndSubsystem" [static, private]
```

Definition at line 246 of file SpatialTemporalDataReader.hpp.

Referenced by setSpatialEntityScaleAndSubsystem().

```
6.186.4.15 const std::string SpatialTemporalDataReader::LABEL_SPATIAL_ENTITY_SPATIAL_TYPE = "<xmllatr>.spatialType" [static, private]
```

Definition at line 245 of file SpatialTemporalDataReader.hpp.

Referenced by createDerivedSpatialEntity().

```
6.186.4.16 const std::string SpatialTemporalDataReader::LABEL_TIMEPOINT_VALUE = "<xmllatr>.value" [static, private]
```

Definition at line 237 of file SpatialTemporalDataReader.hpp.

Referenced by timePointHasValue().

```
6.186.4.17 std::set<std::string> multiscale::verification::-  
SpatialTemporalDataReader::processedInputFiles  
[private]
```

The set of processed input files

Definition at line 24 of file SpatialTemporalDataReader.hpp.

Referenced by clearInputFilesSets(), processInvalidInputFile(), processValidInputFile(), updateInputFilesSets(), and ~SpatialTemporalDataReader().

**6.186.4.18 std::set<std::string> multiscale::verification::SpatialTemporalDataReader::unprocessedInputFiles  
[private]**

The set of unprocessed input files

Definition at line 25 of file SpatialTemporalDataReader.hpp.

Referenced by clearInputFilesSets(), getRandomUnprocessedInputFile(), hasValidNext(), processInvalidInputFile(), processValidInputFile(), updateInputFilesSets(), and ~SpatialTemporalDataReader().

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

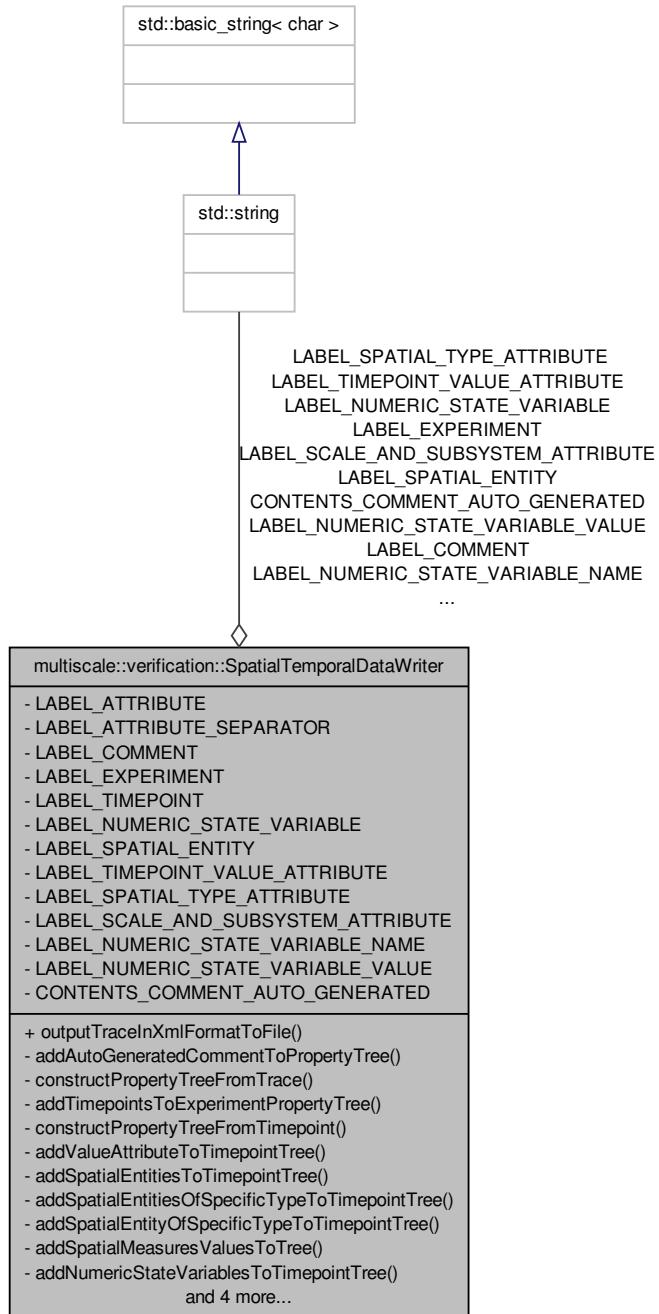
- SpatialTemporalDataReader.hpp
  
- SpatialTemporalDataReader.cpp
  
- SpatialTemporalDataReaderAutoGenerated.cpp

## **6.187 multiscale::verification::SpatialTemporalDataWriter Class Reference**

Class for writing spatial temporal traces to output files.

```
#include <SpatialTemporalDataWriter.hpp>
```

## Collaboration diagram for multiscale::verification::SpatialTemporalDataWriter:



### Static Public Member Functions

- static void [outputTraceInXmlFormatToFile](#) (const [SpatialTemporalTrace](#) &trace, const std::string &outputFilepath)

*Output the given spatial temporal trace in xml format to the specified file path.*

### Static Private Member Functions

- static void [addAutoGeneratedCommentToPropertyTree](#) (pt::ptree &propertyTree)

*Add a comment to the property tree indicating that the property tree was generated automatically.*

- static void [constructPropertyTreeFromTrace](#) (const [SpatialTemporalTrace](#) &trace, pt::ptree &propertyTree)

*Construct the property tree following the structure of the spatial temporal trace.*

- static void [addTimepointsToExperimentPropertyTree](#) (const [SpatialTemporalTrace](#) &trace, pt::ptree &experimentPropertyTree)

*Add timepoints from the trace to the provided experiment property tree.*

- static pt::ptree [constructPropertyTreeFromTimepoint](#) (const [TimePoint](#) &timepoint)

*Construct the property tree corresponding to the provided timepoint.*

- static void [addValueAttributeToTimepointTree](#) (const [TimePoint](#) &timepoint, pt::ptree &timepointTree)

*Add the timepoint value as an attribute to the timepoint tree.*

- static void [addSpatialEntitiesToTimepointTree](#) (const [TimePoint](#) &timepoint, pt::ptree &timepointTree)

*Add the spatial entities from the timepoint to the timepoint tree.*

- static void [addSpatialEntitiesOfSpecificTypeToTimepointTree](#) (const std::size\_t &spatialEntitiesType, const [TimePoint](#) &timepoint, pt::ptree &timepointTree)

*Add the spatial entities of the specified type to the timepoint tree.*

- static void [addSpatialEntityOfSpecificTypeToTimepointTree](#) (const std::string &spatialEntityType, const std::shared\_ptr< [SpatialEntity](#) > &spatialEntity, pt::ptree &timepointTree)

*Add a spatial entity of the specified type to the timepoint tree.*

- static void [addSpatialMeasuresValuesToTree](#) (const std::shared\_ptr< [SpatialEntity](#) > &spatialEntity, pt::ptree &spatialEntityTree)

*Add the spatial measures values for the given spatial entity to the provided spatial entity tree.*

- static void [addNumericStateVariablesToTimepointTree](#) (const [TimePoint](#) &timepoint, pt::ptree &timepointTree)

*Add the numeric state variables from the timepoint to the timepoint tree.*

- static pt::ptree [constructPropertyTreeFromNumericStateVariable](#) (const [NumericStateVariableId](#) &numericStateVariableId, double numericStateVariableValue)

*Construct the property tree corresponding to the provided numeric state variable.*

- static void [addSpatialTypeAttributeToTree](#) (const std::string &spatialType, pt::ptree &propertyTree)

*Add the provided spatial type string as an attribute to the property tree.*

- static void `addScaleAndSubsystemAttributeToTree` (const std::string &scaleAndSubsystem, pt::ptree &propertyTree)

*Add the provided scale and subsystem as an attribute to the property tree.*

- static void `addAttributeToTree` (const std::string &attributeName, const std::string &attributeValue, pt::ptree &propertyTree)

*Add the provided attribute to the property tree.*

- static void `outputPropertyTreeInXmlFormatToFile` (const pt::ptree &propertyTree, const std::string &outputFilepath)

*Output the property tree in xml format to the specified path.*

### Static Private Attributes

- static const std::string `LABEL_ATTRIBUTE` = "<xmلاtr>"
- static const std::string `LABEL_ATTRIBUTE_SEPARATOR` = ":"
- static const std::string `LABEL_COMMENT` = "<xmلاcomment>"
- static const std::string `LABEL_EXPERIMENT` = "experiment"
- static const std::string `LABEL_TIMEPOINT` = "timepoint"
- static const std::string `LABEL_NUMERIC_STATE_VARIABLE` = "numericStateVariable"
- static const std::string `LABEL_SPATIAL_ENTITY` = "spatialEntity"
- static const std::string `LABEL_TIMEPOINT_VALUE_ATTRIBUTE` = "value"
- static const std::string `LABEL_SPATIAL_TYPE_ATTRIBUTE` = "spatialType"
- static const std::string `LABEL_SCALE_AND_SUBSYSTEM_ATTRIBUTE` = "scaleAndSubsystem"
- static const std::string `LABEL_NUMERIC_STATE_VARIABLE_NAME` = "name"
- static const std::string `LABEL_NUMERIC_STATE_VARIABLE_VALUE` = "value"
- static const std::string `CONTENTS_COMMENT_AUTO_GENERATED` = "-Warning! This xml file was automatically generated by a C++ program using the Boost PropertyTree library."

#### 6.187.1 Detailed Description

Class for writing spatial temporal traces to output files.

Definition at line 19 of file SpatialTemporalDataWriter.hpp.

#### 6.187.2 Member Function Documentation

- ##### 6.187.2.1 void `SpatialTemporalDataWriter::addAttributeToTree` ( const std::string & attributeName, const std::string & attributeValue, pt::ptree & propertyTree ) [static, private]

Add the provided attribute to the property tree.

**Parameters**

|                        |                                                   |
|------------------------|---------------------------------------------------|
| <i>attribute-Name</i>  | The name of the attribute                         |
| <i>attribute-Value</i> | The value of the attribute                        |
| <i>propertyTree</i>    | The property tree to which the attribute is added |

Definition at line 173 of file SpatialTemporalDataWriter.cpp.

**6.187.2.2 void SpatialTemporalDataWriter::addAutoGeneratedCommentToPropertyTree ( pt::ptree & *propertyTree* ) [static, private]**

Add a comment to the property tree indicating that the property tree was generated automatically.

**Parameters**

|                     |                                                      |
|---------------------|------------------------------------------------------|
| <i>propertyTree</i> | The property tree to which the comment will be added |
|---------------------|------------------------------------------------------|

Definition at line 17 of file SpatialTemporalDataWriter.cpp.

**6.187.2.3 void SpatialTemporalDataWriter::addNumericStateVariablesToTimepointTree ( const TimePoint & *timepoint*, pt::ptree & *timepointTree* ) [static, private]**

Add the numeric state variables from the timepoint to the timepoint tree.

**Parameters**

|                       |                                                          |
|-----------------------|----------------------------------------------------------|
| <i>timepoint</i>      | The considered timepoint                                 |
| <i>timepoint-Tree</i> | The timepoint tree to which the value attribute is added |

Definition at line 114 of file SpatialTemporalDataWriter.cpp.

References multiscale::verification::TimePoint::getNumericStateVariablesBeginIterator(), and multiscale::verification::TimePoint::getNumericStateVariablesEndIterator().

**6.187.2.4 void SpatialTemporalDataWriter::addScaleAndSubsystemAttributeToTree ( const std::string & *scaleAndSubsystem*, pt::ptree & *propertyTree* ) [static, private]**

Add the provided scale and subsystem as an attribute to the property tree.

**Parameters**

|                           |                                                                       |
|---------------------------|-----------------------------------------------------------------------|
| <i>scaleAnd-Subsystem</i> | The scale and subsystem provided as a string                          |
| <i>propertyTree</i>       | The property tree to which the scale and subsystem attribute is added |

Definition at line 159 of file SpatialTemporalDataWriter.cpp.

```
6.187.2.5 void SpatialTemporalDataWriter::addSpatialEntitiesOfSpecificTypeTo-
TimepointTree ( const std::size_t & spatialEntitiesType, const TimePoint &
timepoint, pt::ptree & timepointTree ) [static, private]
```

Add the spatial entities of the specified type to the timepoint tree.

**Parameters**

|                             |                                                            |
|-----------------------------|------------------------------------------------------------|
| <i>spatial-EntitiesType</i> | The considered spatial entities type                       |
| <i>timepoint</i>            | The considered timepoint                                   |
| <i>timepoint-Tree</i>       | The timepoint tree to which the spatial entities are added |

Definition at line 75 of file SpatialTemporalDataWriter.cpp.

References multiscale::verification::TimePoint::getSpatialEntitiesBeginIterator(), and multiscale::verification::TimePoint::getSpatialEntitiesEndIterator().

```
6.187.2.6 void SpatialTemporalDataWriter::addSpatialEntitiesToTimepointTree
( const TimePoint & timepoint, pt::ptree & timepointTree ) [static,
private]
```

Add the spatial entities from the timepoint to the timepoint tree.

**Parameters**

|                       |                                                            |
|-----------------------|------------------------------------------------------------|
| <i>timepoint</i>      | The considered timepoint                                   |
| <i>timepoint-Tree</i> | The timepoint tree to which the spatial entities are added |

Definition at line 67 of file SpatialTemporalDataWriter.cpp.

```
6.187.2.7 void SpatialTemporalDataWriter::addSpatialEntityOfSpecificTypeTo-
TimepointTree ( const std::string & spatialEntityType, const std::shared_ptr<
SpatialEntity > & spatialEntity, pt::ptree & timepointTree ) [static,
private]
```

Add a spatial entity of the specified type to the timepoint tree.

**Parameters**

|                           |                                                         |
|---------------------------|---------------------------------------------------------|
| <i>spatialEntity-Type</i> | The type of the spatial entity                          |
| <i>spatialEntity</i>      | The considered spatial entity                           |
| <i>timepoint-Tree</i>     | The timepoint tree to which the spatial entity is added |

Definition at line 92 of file SpatialTemporalDataWriter.cpp.

```
6.187.2.8 void SpatialTemporalDataWriter::addSpatialMeasuresValuesToTree (
    const std::shared_ptr< SpatialEntity > & spatialEntity, pt::ptree & spatialEntityTree
) [static, private]
```

Add the spatial measures values for the given spatial entity to the provided spatial entity tree.

**Parameters**

|                           |                                                                       |
|---------------------------|-----------------------------------------------------------------------|
| <i>spatialEntity</i>      | The considered spatial entity                                         |
| <i>spatialEntity-Tree</i> | The spatial entity tree to which the spatial measure values are added |

Definition at line 21 of file SpatialTemporalDataWriterAutoGenerated.cpp.

```
6.187.2.9 void SpatialTemporalDataWriter::addSpatialTypeAttributeToTree ( const
    std::string & spatialType, pt::ptree & propertyTree ) [static, private]
```

Add the provided spatial type string as an attribute to the property tree.

**Parameters**

|                     |                                                                |
|---------------------|----------------------------------------------------------------|
| <i>spatialType</i>  | The spatial type value provided as a string                    |
| <i>propertyTree</i> | The property tree to which the spatial type attribute is added |

Definition at line 150 of file SpatialTemporalDataWriter.cpp.

```
6.187.2.10 void SpatialTemporalDataWriter::addTimepointsToExperimentProperty-
Tree ( const SpatialTemporalTrace & trace, pt::ptree & experimentPropertyTree
) [static, private]
```

Add timepoints from the trace to the provided experiment property tree.

**Parameters**

|                                 |                                                                                   |
|---------------------------------|-----------------------------------------------------------------------------------|
| <i>trace</i>                    | The provided spatial temporal trace                                               |
| <i>experiment-Property-Tree</i> | The experiment property tree which corresponds to the experiment root xml element |

Definition at line 32 of file SpatialTemporalDataWriter.cpp.

References multiscale::verification::SpatialTemporalTrace::getTimePointReference(), and multiscale::verification::SpatialTemporalTrace::length().

```
6.187.2.11 void SpatialTemporalDataWriter::addValueAttributeToTimepointTree (
    const TimePoint & timepoint, pt::ptree & timepointTree ) [static,
    private]
```

Add the timepoint value as an attribute to the timepoint tree.

#### Parameters

|                       |                                                          |
|-----------------------|----------------------------------------------------------|
| <i>timepoint</i>      | The considered timepoint                                 |
| <i>timepoint-Tree</i> | The timepoint tree to which the value attribute is added |

Definition at line 57 of file SpatialTemporalDataWriter.cpp.

References multiscale::verification::TimePoint::getValue(), and multiscale::StringManipulator::toString().

```
6.187.2.12 pt::ptree SpatialTemporalDataWriter::constructPropertyTree-
    FromNumericStateVariable ( const NumericStateVariableId &
    numericStateVariableId, double numericStateVariableValue ) [static,
    private]
```

Construct the property tree corresponding to the provided numeric state variable.

#### Parameters

|                                     |                                                                                |
|-------------------------------------|--------------------------------------------------------------------------------|
| <i>numeric-State-VariableId</i>     | The id of the numeric state variable to which the property tree corresponds    |
| <i>numeric-State-Variable-Value</i> | The value of the numeric state variable to which the property tree corresponds |

Definition at line 133 of file SpatialTemporalDataWriter.cpp.

References multiscale::verification::NumericStateVariableId::getName(), and multiscale::verification::StateVariable::getScaleAndSubsystem().

```
6.187.2.13 pt::ptree SpatialTemporalDataWriter::constructPropertyTree-
    FromTimepoint ( const TimePoint & timepoint ) [static,
    private]
```

Construct the property tree corresponding to the provided timepoint.

**Parameters**

|                  |                        |
|------------------|------------------------|
| <i>timepoint</i> | The provided timepoint |
|------------------|------------------------|

Definition at line 47 of file SpatialTemporalDataWriter.cpp.

**6.187.2.14 void SpatialTemporalDataWriter::constructPropertyTreeFromTrace ( const SpatialTemporalTrace & trace, pt::ptree & propertyTree ) [static, private]**

Construct the property tree following the structure of the spatial temporal trace.

**Parameters**

|                     |                                                                   |
|---------------------|-------------------------------------------------------------------|
| <i>trace</i>        | The provided spatial temporal trace                               |
| <i>propertyTree</i> | The property tree which corresponds to the spatial temporal trace |

Definition at line 21 of file SpatialTemporalDataWriter.cpp.

**6.187.2.15 void SpatialTemporalDataWriter::outputPropertyTreeInXmlFormatToFile ( const pt::ptree & propertyTree, const std::string & outputfilepath ) [static, private]**

Output the property tree in xml format to the specified path.

**Parameters**

|                        |                                                                   |
|------------------------|-------------------------------------------------------------------|
| <i>propertyTree</i>    | The property tree which corresponds to the spatial temporal trace |
| <i>output-Filepath</i> | The path to the file where the property tree will be written      |

Definition at line 183 of file SpatialTemporalDataWriter.cpp.

**6.187.2.16 void SpatialTemporalDataWriter::outputTraceInXmlFormatToFile ( const SpatialTemporalTrace & trace, const std::string & outputfilepath ) [static]**

Output the given spatial temporal trace in xml format to the specified file path.

**Parameters**

|                        |                                                      |
|------------------------|------------------------------------------------------|
| <i>trace</i>           | The provided spatial temporal trace                  |
| <i>output-Filepath</i> | The path of the file where the trace will be written |

Definition at line 8 of file SpatialTemporalDataWriter.cpp.

Referenced by multiscale::verification::MSTMLSubfilesMerger::outputResultingMSTM-File().

### 6.187.3 Member Data Documentation

6.187.3.1 `const std::string SpatialTemporalDataWriter::CONTENTS_COMMENT_AUTO_GENERATED = "Warning! This xml file was automatically generated by a C++ program using the Boost PropertyTree library." [static, private]`

Definition at line 191 of file SpatialTemporalDataWriter.hpp.

6.187.3.2 `const std::string SpatialTemporalDataWriter::LABEL_ATTRIBUTE = "<xmlattr>" [static, private]`

Definition at line 175 of file SpatialTemporalDataWriter.hpp.

6.187.3.3 `const std::string SpatialTemporalDataWriter::LABEL_ATTRIBUTE_SEPARATOR = ":" [static, private]`

Definition at line 176 of file SpatialTemporalDataWriter.hpp.

6.187.3.4 `const std::string SpatialTemporalDataWriter::LABEL_COMMENT = "<xmlcomment>" [static, private]`

Definition at line 177 of file SpatialTemporalDataWriter.hpp.

6.187.3.5 `const std::string SpatialTemporalDataWriter::LABEL_EXPERIMENT = "experiment" [static, private]`

Definition at line 179 of file SpatialTemporalDataWriter.hpp.

6.187.3.6 `const std::string SpatialTemporalDataWriter::LABEL_NUMERIC_STATE_VARIABLE = "numericStateVariable" [static, private]`

Definition at line 181 of file SpatialTemporalDataWriter.hpp.

6.187.3.7 `const std::string SpatialTemporalDataWriter::LABEL_NUMERIC_STATE_VARIABLE_NAME = "name" [static, private]`

Definition at line 188 of file SpatialTemporalDataWriter.hpp.

```
6.187.3.8 const std::string SpatialTemporalDataWriter::LABEL_NUM-
ERIC_STATE_VARIABLE_VALUE = "value" [static,
private]
```

Definition at line 189 of file SpatialTemporalDataWriter.hpp.

```
6.187.3.9 const std::string SpatialTemporalDataWriter::LABEL_SCALE_AN-
D_SUBSYSTEM_ATTRIBUTE = "scaleAndSubsystem" [static,
private]
```

Definition at line 186 of file SpatialTemporalDataWriter.hpp.

```
6.187.3.10 const std::string SpatialTemporalDataWriter::LABEL_SPATIAL_ENTITY =
"spatialEntity" [static, private]
```

Definition at line 182 of file SpatialTemporalDataWriter.hpp.

```
6.187.3.11 const std::string SpatialTemporalDataWriter::LABEL_S-
PATIAL_TYPE_ATTRIBUTE = "spatialType" [static,
private]
```

Definition at line 185 of file SpatialTemporalDataWriter.hpp.

```
6.187.3.12 const std::string SpatialTemporalDataWriter::LABEL_TIMEPOINT =
"timepoint" [static, private]
```

Definition at line 180 of file SpatialTemporalDataWriter.hpp.

```
6.187.3.13 const std::string SpatialTemporalDataWriter::LABEL_T-
IMEPOINT_VALUE_ATTRIBUTE = "value" [static,
private]
```

Definition at line 184 of file SpatialTemporalDataWriter.hpp.

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

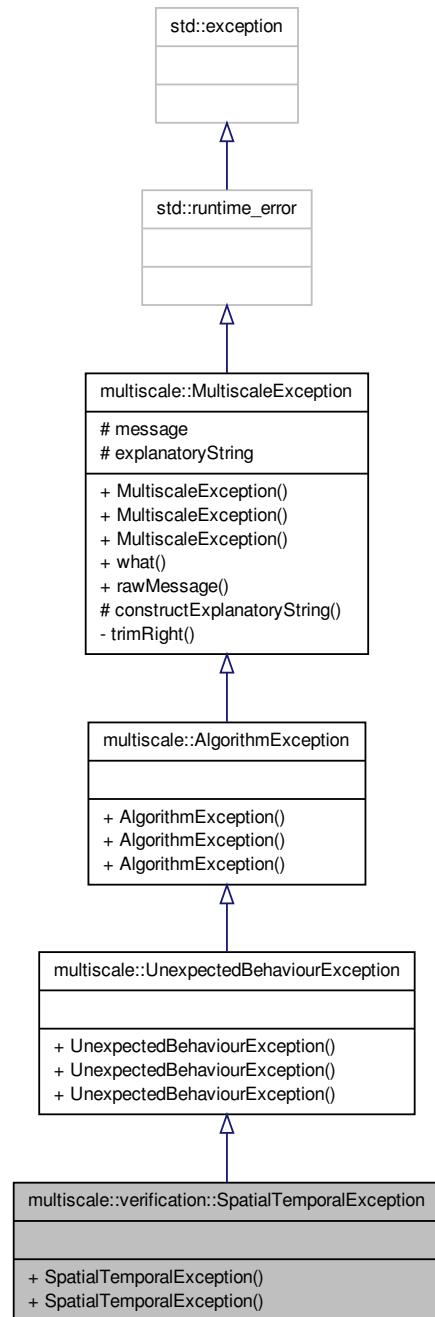
- SpatialTemporalDataWriter.hpp
- SpatialTemporalDataWriter.cpp
- SpatialTemporalDataWriterAutoGenerated.cpp

## **6.188 multiscale::verification::SpatialTemporalException Class - Reference**

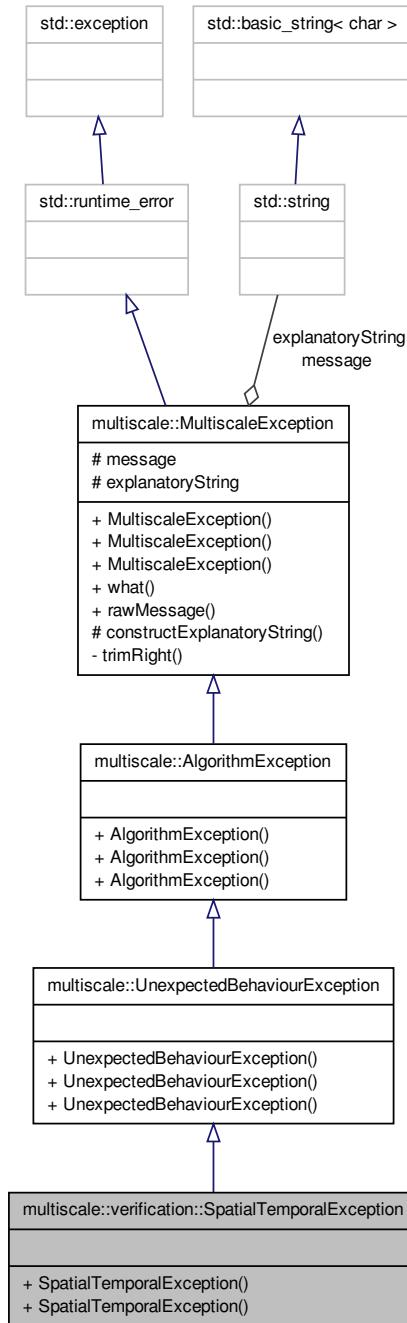
Class for representing a spatial temporal exception.

```
#include <SpatialTemporalException.hpp>
```

Inheritance diagram for multiscale::verification::SpatialTemporalException:



Collaboration diagram for multiscale::verification::SpatialTemporalException:



## Public Member Functions

- [SpatialTemporalException \(const std::string &file, int line, const std::string &msg\)](#)
- [SpatialTemporalException \(const std::string &file, int line, const char \\*msg\)](#)

### 6.188.1 Detailed Description

Class for representing a spatial temporal exception.

Definition at line 14 of file SpatialTemporalException.hpp.

### 6.188.2 Constructor & Destructor Documentation

6.188.2.1 **multiscale::verification::SpatialTemporalException::SpatialTemporalException ( const std::string & file, int line, const std::string & msg ) [inline, explicit]**

Definition at line 18 of file SpatialTemporalException.hpp.

6.188.2.2 **multiscale::verification::SpatialTemporalException::SpatialTemporalException ( const std::string & file, int line, const char \* msg ) [inline, explicit]**

Definition at line 23 of file SpatialTemporalException.hpp.

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

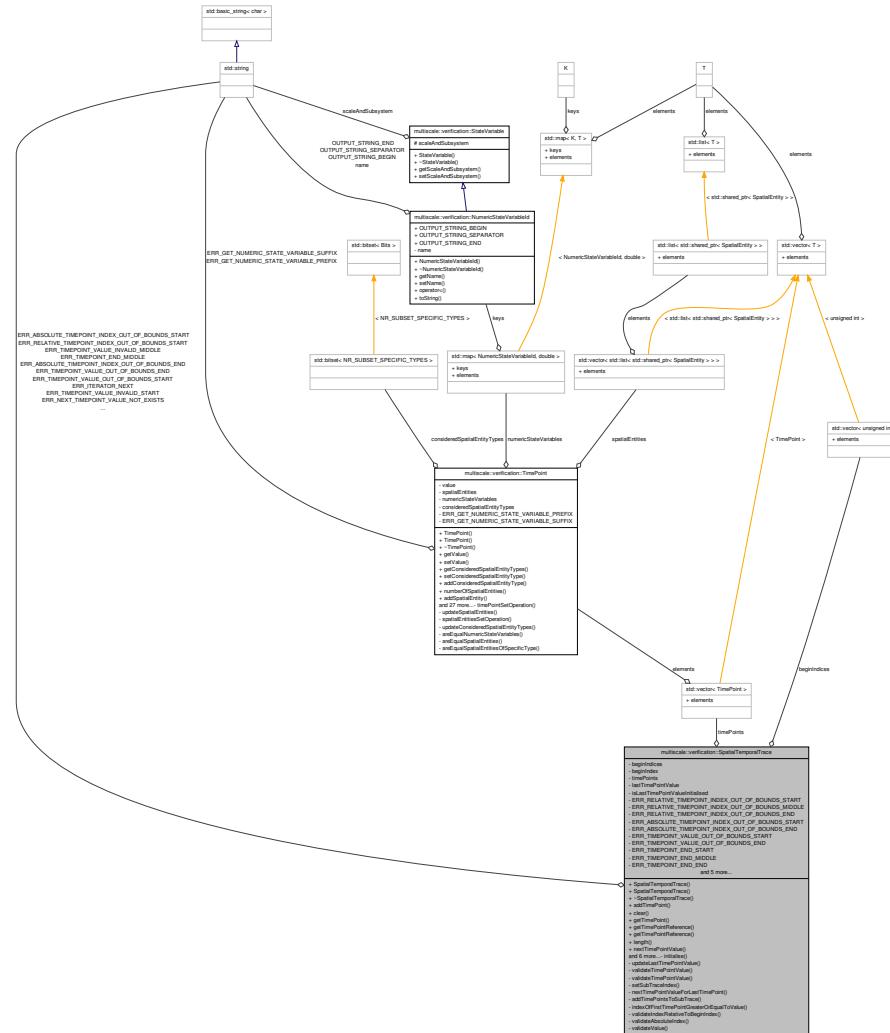
- [SpatialTemporalException.hpp](#)

## 6.189 multiscale::verification::SpatialTemporalTrace Class Reference -

Class for representing a spatial temporal trace.

```
#include <SpatialTemporalTrace.hpp>
```

## Collaboration diagram for multiscale::verification::SpatialTemporalTrace:



# Public Member Functions

- `SpatialTemporalTrace ()`
  - `SpatialTemporalTrace (const SpatialTemporalTrace &trace)`
  - `~SpatialTemporalTrace ()`
  - `void addTimePoint (const TimePoint &timePoint)`

*Add a time point to the array*

- void clear()

*Clear all the stored timepoints and reinitialise.*

- `TimePoint getTimePoint (unsigned int index) const`

- `TimePoint & getTimePointReference (unsigned int index)`

*Get the time point at the given index in the array.*
- `const TimePoint & getTimePointReference (unsigned int index) const`

*Get the reference to the time point at the given index in the array.*
- `unsigned int length () const`

*Get the reference to the time point at the given index in the array.*
- `unsigned long nextTimePointValue () const`

*Get the length of the spatial temporal trace (i.e. number of timepoints)*
- `void setSubTraceWithTimepointsValuesGreaterOrEqualTo (unsigned long startValue)`

*Set the subtrace containing timepoints with values greater or equal to the given start value.*
- `void advanceTraceBeginIndex (unsigned long advanceValue)`

*Advance the trace begin index by the given value.*
- `void setTraceBeginIndex (unsigned int newBeginIndex)`

*Set the trace begin index to the given value.*
- `void storeSubTraceBeginIndex ()`

*Add the current value of beginIndex to the beginIndices stack.*
- `unsigned int getMostRecentlyStoredSubTraceBeginIndex ()`

*Retrieve the value of the most recent stored subtrace begin index.*
- `void restoreSubTraceBeginIndex ()`

*Restore the subtrace beginIndex from the beginIndices stack.*
- `bool operator== (const SpatialTemporalTrace &rhsSpatialTemporalTrace)`

### Private Member Functions

- `void initialise ()`

*Initialise the member fields.*
- `void updateLastTimePointValue (TimePoint &timePoint)`

*Update the last timepoint value.*
- `void validateTimePointValue (const TimePoint &timePoint)`

*Check if the provided time point value is greater than the last time point value.*
- `void validateTimePointValue (unsigned long timePointValue)`

*Check if the provided time point value is greater than the last time point value.*
- `void setSubTraceIndex (unsigned long startValue)`

*Set the begin index for the subtrace starting with the given value.*
- `unsigned long nextTimePointValueForLastTimePoint () const`

*Get the value of the next timepoint when beginIndex is the index of the last timepoint.*
- `void addTimePointsToSubTrace (SpatialTemporalTrace &subTrace, int startIndex, int endIndex) const`

*Add the timepoints starting and ending with the given indices to the subtrace.*
- `int indexOfFirstTimePointGreaterOrEqualToValue (unsigned long value) const`

*Get the index of the first timepoint which has a value greater than or equal to the given value.*

- void [validateIndexRelativeToBeginIndex](#) (unsigned int index) const
 

*Check if relative to beginIndex the provided index is smaller than the number of timepoints.*
- void [validateAbsoluteIndex](#) (unsigned int index) const
 

*Check if the provided absolute index is smaller than the number of timepoints.*
- void [validateValue](#) (unsigned long value) const
 

*Check if the provided value is smaller than or equal to the maximum timepoint value.*

### Private Attributes

- std::vector< unsigned int > [beginIndices](#)
- unsigned int [beginIndex](#)
- std::vector< TimePoint > [timePoints](#)
- unsigned long [lastTimePointValue](#)
- bool [isLastTimePointValueInitialised](#)

### Static Private Attributes

- static const std::string [ERR\\_RELATIVE\\_TIMEPOINT\\_INDEX\\_OUT\\_OF\\_BOUNDS\\_START](#) = "Relative to the begin index ("
- static const std::string [ERR\\_RELATIVE\\_TIMEPOINT\\_INDEX\\_OUT\\_OF\\_BOUNDS\\_MIDDLE](#) = ") the provided timepoint index ("
- static const std::string [ERR\\_RELATIVE\\_TIMEPOINT\\_INDEX\\_OUT\\_OF\\_BOUNDS\\_END](#) = ") is out of bounds for the given spatial temporal trace."
- static const std::string [ERR\\_ABSOLUTE\\_TIMEPOINT\\_INDEX\\_OUT\\_OF\\_BOUNDS\\_START](#) = "The provided timepoint index ("
- static const std::string [ERR\\_ABSOLUTE\\_TIMEPOINT\\_INDEX\\_OUT\\_OF\\_BOUNDS\\_END](#) = ") is out of bounds for the given spatial temporal trace."
- static const std::string [ERR\\_TIMEPOINT\\_VALUE\\_OUT\\_OF\\_BOUNDS\\_START](#) = "The provided timepoint value ("
- static const std::string [ERR\\_TIMEPOINT\\_VALUE\\_OUT\\_OF\\_BOUNDS\\_END](#) = ") is out of bounds for the given spatial temporal trace."
- static const std::string [ERR\\_TIMEPOINT\\_END\\_START](#) = "The provided end timepoint ("
- static const std::string [ERR\\_TIMEPOINT\\_END\\_MIDDLE](#) = ") should be greater or equal to the start timepoint ("
- static const std::string [ERR\\_TIMEPOINT\\_END\\_END](#) = ")."
- static const std::string [ERR\\_TIMEPOINT\\_VALUE\\_INVALID\\_START](#) = "The current timepoint value ("
- static const std::string [ERR\\_TIMEPOINT\\_VALUE\\_INVALID\\_MIDDLE](#) = ") should be greater than the previously added timepoint value ("
- static const std::string [ERR\\_TIMEPOINT\\_VALUE\\_INVALID\\_END](#) = ")."

- static const std::string `ERR_NEXT_TIMEPOINT_VALUE_NOT_EXISTS` = "The value of the last timepoint is the maximum value which can be represented by an unsigned long. Therefore a next timepoint value, which is greater than the value of the last timepoint, does not exist."
- static const std::string `ERR_ITERATOR_NEXT` = " before calling the next() method."
- static const int `TIMEPOINT_INDEX_NOT_FOUND` = -1

### 6.189.1 Detailed Description

Class for representing a spatial temporal trace.

Definition at line 15 of file SpatialTemporalTrace.hpp.

### 6.189.2 Constructor & Destructor Documentation

#### 6.189.2.1 SpatialTemporalTrace::SpatialTemporalTrace ( )

Definition at line 11 of file SpatialTemporalTrace.cpp.

References initialise().

#### 6.189.2.2 SpatialTemporalTrace::SpatialTemporalTrace ( const SpatialTemporalTrace & trace )

Definition at line 15 of file SpatialTemporalTrace.cpp.

#### 6.189.2.3 SpatialTemporalTrace::~SpatialTemporalTrace ( )

Definition at line 20 of file SpatialTemporalTrace.cpp.

### 6.189.3 Member Function Documentation

#### 6.189.3.1 void SpatialTemporalTrace::addTimePoint ( const TimePoint & timePoint )

Add a time point to the array.

##### Parameters

|                        |                               |
|------------------------|-------------------------------|
| <code>timePoint</code> | Time point added to the array |
|------------------------|-------------------------------|

Definition at line 22 of file SpatialTemporalTrace.cpp.

References timePoints, updateLastTimePointValue(), and validateTimePointValue().

Referenced by multiscale::verification::MSTMLSubfilesMerger::addSubtraceState-VariablesToEmptyResultingTrace(), addTimePointsToSubTrace(), multiscale::verification-

::SpatialTemporalDataReader::addTimePointToTrace(), multiscale::verification::TemporalDataReader::createTimePointFromTokens(), and multiscaletest::ModelCheckerTest::InitialiseSpatioTemporalTraceWithAreaValues().

```
6.189.3.2 void SpatialTemporalTrace::addTimePointsToSubTrace (
    SpatialTemporalTrace & subTrace, int startIndex, int endIndex ) const
    [private]
```

Add the timepoints starting and ending with the given indices to the subtrace.

#### Parameters

|                   |                              |
|-------------------|------------------------------|
| <i>subTrace</i>   | The resulting subtrace       |
| <i>startIndex</i> | The starting timepoint index |
| <i>endIndex</i>   | The end timepoint index      |

Definition at line 193 of file SpatialTemporalTrace.cpp.

References addTimePoint(), and timePoints.

```
6.189.3.3 void SpatialTemporalTrace::advanceTraceBeginIndex ( unsigned long
    advanceValue )
```

Advance the trace begin index by the given value.

#### Parameters

|                      |                                                             |
|----------------------|-------------------------------------------------------------|
| <i>advance-Value</i> | The value by which the trace begin index should be advanced |
|----------------------|-------------------------------------------------------------|

Definition at line 69 of file SpatialTemporalTrace.cpp.

References beginIndex, and validateIndexRelativeToBeginIndex().

Referenced by multiscale::verification::LogicPropertyVisitor::evaluateNextKLogicProperty(), and multiscale::verification::LogicPropertyVisitor::evaluateTemporalNumericMeasure().

```
6.189.3.4 void SpatialTemporalTrace::clear ( )
```

Clear all the stored timepoints and reinitialise.

Definition at line 30 of file SpatialTemporalTrace.cpp.

References initialise().

```
6.189.3.5 unsigned int SpatialTemporalTrace::getMostRecentlyStoredSubTrace-
    BeginIndex ( )
```

Retrieve the value of the most recent stored subtrace begin index.

If the beginIndices stack is non-empty then retrieve the top value in the stack. Otherwise return 0.

Definition at line 90 of file SpatialTemporalTrace.cpp.

References beginIndices.

#### 6.189.3.6 TimePoint SpatialTemporalTrace::getTimePoint ( *unsigned int index* ) const

Get the time point at the given index in the array.

##### Parameters

|              |                                        |
|--------------|----------------------------------------|
| <i>index</i> | The index of the position in the array |
|--------------|----------------------------------------|

Definition at line 34 of file SpatialTemporalTrace.cpp.

References beginIndex, timePoints, and validateIndexRelativeToBeginIndex().

Referenced by multiscale::verification::MSTMLSubfilesMerger::addSubtraceState-VariablesToEmptyResultingTrace().

#### 6.189.3.7 TimePoint & SpatialTemporalTrace::getTimePointReference ( *unsigned int index* )

Get the reference to the time point at the given index in the array.

##### Parameters

|              |                                        |
|--------------|----------------------------------------|
| <i>index</i> | The index of the position in the array |
|--------------|----------------------------------------|

Definition at line 40 of file SpatialTemporalTrace.cpp.

References beginIndex, timePoints, and validateIndexRelativeToBeginIndex().

Referenced by multiscale::verification::MSTMLSubfilesMerger::addSubtraceState-VariablesToNonEmptyResultingTrace(), multiscale::verification::SpatialTemporal-DataWriter::addTimepointsToExperimentPropertyTree(), multiscale::verification::M-STMLSubfilesMerger::areMismatchingTimepointValues(), multiscale::verification::-LogicPropertyVisitor::evaluateChangeLhsTemporalNumericMeasure(), multiscale-::verification::NumericMeasureCollectionVisitor::evaluateTemporalNumericMeasure-Collection(), multiscale::verification::TemporalNumericVisitor::operator()(), and multiscale::verification::MSTMLSubfilesMerger::updateResultingTraceTimepoints-Values().

#### 6.189.3.8 const TimePoint & SpatialTemporalTrace::getTimePointReference ( *unsigned int index* ) const

Get the reference to the time point at the given index in the array.

**Parameters**

|              |                                        |
|--------------|----------------------------------------|
| <i>index</i> | The index of the position in the array |
|--------------|----------------------------------------|

Definition at line 46 of file SpatialTemporalTrace.cpp.

References beginIndex, timePoints, and validateIndexRelativeToBeginIndex().

```
6.189.3.9 int SpatialTemporalTrace::indexOfFirstTimePoint-
GreaterOrEqualToValue ( unsigned long value ) const
[private]
```

Get the index of the first timepoint which has a value greater than or equal to the given value.

**Parameters**

|              |                 |
|--------------|-----------------|
| <i>value</i> | The given value |
|--------------|-----------------|

Definition at line 200 of file SpatialTemporalTrace.cpp.

References beginIndex, TIMEPOINT\_INDEX\_NOT\_FOUND, and timePoints.

Referenced by setSubTraceIndex().

```
6.189.3.10 void SpatialTemporalTrace::initialise ( ) [private]
```

Initialise the member fields.

Definition at line 133 of file SpatialTemporalTrace.cpp.

References beginIndex, isLastTimePointValueInitialised, lastTimePointValue, and timePoints.

Referenced by clear(), and SpatialTemporalTrace().

```
6.189.3.11 unsigned int SpatialTemporalTrace::length ( ) const
```

Get the length of the spatial temporal trace (i.e. number of timepoints)

Definition at line 52 of file SpatialTemporalTrace.cpp.

References beginIndex, and timePoints.

Referenced by multiscale::verification::MSTMLSubfilesMerger::addSubtraceState-VariablesToResultingTrace(), multiscale::verification::SpatialTemporalDataWriter::add-TimepointsToExperimentPropertyTree(), multiscale::verification::MSTMLSubfiles-Merger::areMismatchingTimepointValues(), multiscale::verification::ProbabilisticLogic-PropertyAttribute::evaluate(), multiscale::verification::MSTMLSubfilesMerger::output-ResultingMSTMLFile(), multiscale::verification::MSTMLSubfilesMerger::validate-NumberOfTimepointsInResultingTrace(), multiscale::verification::MSTMLSubfiles-Merger::validateSubtraceNumberOfTimepoints(), and multiscale::verification::MST-MLSSubfilesMerger::validateSubtraceTimepointsValues().

**6.189.3.12 unsigned long SpatialTemporalTrace::nextTimePointValue( ) const**

Get the value of the next timepoint considering beginIndex.

Definition at line 56 of file SpatialTemporalTrace.cpp.

References beginIndex, nextTimePointValueForLastTimePoint(), and timePoints.

Referenced by multiscale::verification::LogicPropertyVisitor::evaluateTemporalLogicPropertyWithStartAndEndTimepoints(), multiscale::verification::NumericMeasureCollectionVisitor::evaluateTemporalNumericMeasureCollection(), and multiscale::verification::NumericMeasureCollectionVisitor::evaluateTemporalNumericMeasureCollectionTimepoints().

**6.189.3.13 unsigned long SpatialTemporalTrace::nextTimePointValueForLastTimePoint( ) const [private]**

Get the value of the next timepoint when beginIndex is the index of the last timepoint.

Return maximum unsigned long value if the value of the last timepoint is smaller than the maximum unsigned long value. Otherwise throw an exception.

Definition at line 185 of file SpatialTemporalTrace.cpp.

References ERR\_NEXT\_TIMEPOINT\_VALUE\_NOT\_EXISTS, and timePoints.

Referenced by nextTimePointValue().

**6.189.3.14 bool SpatialTemporalTrace::operator==( const SpatialTemporalTrace & rhsSpatialTemporalTrace )****Parameters**

|                                  |                                                                          |
|----------------------------------|--------------------------------------------------------------------------|
| <i>rhsSpatial-Temporal-Trace</i> | The provided spatial temporal trace against which this trace is compared |
|----------------------------------|--------------------------------------------------------------------------|

Definition at line 110 of file SpatialTemporalTrace.cpp.

References timePoints.

**6.189.3.15 void SpatialTemporalTrace::restoreSubTraceBeginIndex( )**

Restore the subtrace beginIndex from the beginIndices stack.

If the beginIndices stack is non-empty then the value of beginIndex will be equal to the top value in the stack. Otherwise the value of beginIndex will be set to 0.

If the value of beginIndex is updated from the beginIndices stack then the top value is popped out of the stack.

Definition at line 98 of file SpatialTemporalTrace.cpp.

References beginIndex, and beginIndices.

Referenced by multiscale::verification::LogicPropertyVisitor::evaluateNextKLogicProperty(), multiscale::verification::LogicPropertyVisitor::evaluateTemporalLogicPropertyWithStartEndTimepoints(), multiscale::verification::LogicPropertyVisitor::evaluateTemporalNumericMeasure(), multiscale::verification::NumericMeasureCollectionVisitor::evaluateTemporalNumericMeasureCollection(), and multiscale::verification::NumericMeasureCollectionVisitor::evaluateTemporalNumericMeasureCollectionTimepoints().

**6.189.3.16 void SpatialTemporalTrace::setSubTraceIndex ( unsigned long *startValue* ) [private]**

Set the begin index for the subtrace starting with the given value.

**Parameters**

|                   |                                              |
|-------------------|----------------------------------------------|
| <i>startValue</i> | The starting timepoint value of the subtrace |
|-------------------|----------------------------------------------|

Definition at line 175 of file SpatialTemporalTrace.cpp.

References beginIndex, indexOfFirstTimePointGreaterOrEqualToValue(), TIMEPOINT\_INDEX\_NOT\_FOUND, and timePoints.

Referenced by setSubTraceWithTimepointsValuesGreaterOrEqualTo().

**6.189.3.17 void SpatialTemporalTrace::setSubTraceWithTimepoints-ValuesGreaterOrEqualTo ( unsigned long *startValue* )**

Set the subtrace containing timepoints with values greater or equal to the given start value.

**Parameters**

|                   |                                    |
|-------------------|------------------------------------|
| <i>startValue</i> | The starting value of the subtrace |
|-------------------|------------------------------------|

Definition at line 64 of file SpatialTemporalTrace.cpp.

References setSubTraceIndex(), and validateValue().

Referenced by multiscale::verification::LogicPropertyVisitor::evaluateTemporalLogicPropertyWithStartAndEndTimepoints(), multiscale::verification::NumericMeasureCollectionVisitor::evaluateTemporalNumericMeasureCollection(), and multiscale::verification::NumericMeasureCollectionVisitor::evaluateTemporalNumericMeasureCollectionTimepoints().

**6.189.3.18 void SpatialTemporalTrace::setTraceBeginIndex ( unsigned int *newBeginIndex* )**

Set the trace begin index to the given value.

**Parameters**

|                       |                                                 |
|-----------------------|-------------------------------------------------|
| <i>newBegin-Index</i> | The new value of the trace starting/begin index |
|-----------------------|-------------------------------------------------|

Definition at line 77 of file SpatialTemporalTrace.cpp.

References beginIndex, and validateAbsoluteIndex().

**6.189.3.19 void SpatialTemporalTrace::storeSubTraceBeginIndex( )**

Add the current value of beginIndex to the beginIndices stack.

Definition at line 85 of file SpatialTemporalTrace.cpp.

References beginIndex, and beginIndices.

Referenced by multiscale::verification::LogicPropertyVisitor::evaluateNextKLogicProperty(), multiscale::verification::LogicPropertyVisitor::evaluateTemporalLogicPropertyWithStartTimepoints(), multiscale::verification::LogicPropertyVisitor::evaluateTemporalNumericMeasure(), multiscale::verification::NumericMeasureCollectionVisitor::evaluateTemporalNumericMeasureCollection(), and multiscale::verification::NumericMeasureCollectionVisitor::evaluateTemporalNumericMeasureCollectionTimepoints().

**6.189.3.20 void SpatialTemporalTrace::updateLastTimePointValue( TimePoint & timePoint ) [private]**

Update the last timepoint value.

**Parameters**

|                  |                          |
|------------------|--------------------------|
| <i>timePoint</i> | The last added timepoint |
|------------------|--------------------------|

Definition at line 142 of file SpatialTemporalTrace.cpp.

References multiscale::verification::TimePoint::getValue(), lastTimePointValue, and multiscale::verification::TimePoint::setValue().

Referenced by addTimePoint().

**6.189.3.21 void SpatialTemporalTrace::validateAbsoluteIndex( unsigned int index ) const [private]**

Check if the provided absolute index is smaller than the number of timepoints.

**Parameters**

|              |                    |
|--------------|--------------------|
| <i>index</i> | The provided index |
|--------------|--------------------|

Definition at line 228 of file SpatialTemporalTrace.cpp.

References `ERR_ABSOLUTE_TIMEPOINT_INDEX_OUT_OF_BOUNDS_END`, `ERR_ABSOLUTE_TIMEPOINT_INDEX_OUT_OF_BOUNDS_START`, `timePoints`, and `multiscale::StringManipulator::toString()`.

Referenced by `setTraceBeginIndex()`.

#### 6.189.3.22 void SpatialTemporalTrace::validateIndexRelativeToBeginIndex ( unsigned int *index* ) const [private]

Check if relative to beginIndex the provided index is smaller than the number of timepoints.

##### Parameters

|              |                    |
|--------------|--------------------|
| <i>index</i> | The provided index |
|--------------|--------------------|

Definition at line 215 of file `SpatialTemporalTrace.cpp`.

References `beginIndex`, `ERR_RELATIVE_TIMEPOINT_INDEX_OUT_OF_BOUNDS_END`, `ERR_RELATIVE_TIMEPOINT_INDEX_OUT_OF_BOUNDS_MIDDLE`, `ERR_RELATIVE_TIMEPOINT_INDEX_OUT_OF_BOUNDS_START`, `timePoints`, and `multiscale::StringManipulator::toString()`.

Referenced by `advanceTraceBeginIndex()`, `getTimePoint()`, and `getTimePointReference()`.

#### 6.189.3.23 void SpatialTemporalTrace::validateTimePointValue ( const TimePoint & *timePoint* ) [private]

Check if the provided time point value is greater than the last time point value.

The timepoint is considered to be uninitialised if the value is equal to the maximum value which can be represented as an unsigned long. Otherwise if the timepoint value is less or equal to the lastTimePointValue then an exception is thrown.

##### Parameters

|                  |                      |
|------------------|----------------------|
| <i>timePoint</i> | The given time point |
|------------------|----------------------|

Definition at line 152 of file `SpatialTemporalTrace.cpp`.

References `multiscale::verification::TimePoint::getValue()`.

Referenced by `addTimePoint()`.

#### 6.189.3.24 void SpatialTemporalTrace::validateTimePointValue ( unsigned long *timePointValue* ) [private]

Check if the provided time point value is greater than the last time point value.

The timepoint is considered to be uninitialised if the value is equal to the maximum value

which can be represented as an unsigned long. Otherwise if the timepoint value is less or equal to the lastTimePointValue then an exception is thrown.

**Parameters**

|                        |                            |
|------------------------|----------------------------|
| <i>timePoint-Value</i> | The value of the timepoint |
|------------------------|----------------------------|

Definition at line 158 of file SpatialTemporalTrace.cpp.

References ERR\_TIMEPOINT\_VALUE\_INVALID\_END, ERR\_TIMEPOINT\_VALUE\_INVALID\_MIDDLE, ERR\_TIMEPOINT\_VALUE\_INVALID\_START, isLastTimePointValueInitialised, and lastTimePointValue.

**6.189.3.25 void SpatialTemporalTrace::validateValue ( unsigned long *value* ) const [private]**

Check if the provided value is smaller than or equal to the maximum timepoint value.

**Parameters**

|              |                    |
|--------------|--------------------|
| <i>value</i> | The provided value |
|--------------|--------------------|

Definition at line 239 of file SpatialTemporalTrace.cpp.

References ERR\_TIMEPOINT\_VALUE\_OUT\_OF\_BOUNDS\_END, ERR\_TIMEPOINT\_VALUE\_OUT\_OF\_BOUNDS\_START, and timePoints.

Referenced by setSubTraceWithTimepointsValuesGreaterOrEqualTo().

**6.189.4 Member Data Documentation**

**6.189.4.1 unsigned int multiscale::verification::SpatialTemporalTrace::beginIndex [private]**

The currently employed trace begin index

Definition at line 20 of file SpatialTemporalTrace.hpp.

Referenced by advanceTraceBeginIndex(), getTimePoint(), getTimePointReference(), indexOfFirstTimePointGreaterOrEqualToValue(), initialise(), length(), nextTimePointValue(), restoreSubTraceBeginIndex(), setSubTraceIndex(), setTraceBeginIndex(), storeSubTraceBeginIndex(), and validateIndexRelativeToBeginIndex().

**6.189.4.2 std::vector<unsigned int> multiscale::verification::SpatialTemporalTrace::beginIndices [private]**

The stack of stored trace begin indices

Definition at line 19 of file SpatialTemporalTrace.hpp.

Referenced by getMostRecentlyStoredSubTraceBeginIndex(), restoreSubTraceBeginIndex(), and storeSubTraceBeginIndex().

6.189.4.3 `const std::string SpatialTemporalTrace::ERR_ABSOLUTE_TIMEPOINT_INDEX_OUT_OF_BOUNDS_END = "The provided timepoint index ("`  
[static, private]

Definition at line 194 of file SpatialTemporalTrace.hpp.

Referenced by validateAbsoluteIndex().

6.189.4.4 `const std::string SpatialTemporalTrace::ERR_ABSOLUTE_TIMEPOINT_INDEX_OUT_OF_BOUNDS_START = "The provided timepoint index ("`  
[static, private]

Definition at line 193 of file SpatialTemporalTrace.hpp.

Referenced by validateAbsoluteIndex().

6.189.4.5 `const std::string SpatialTemporalTrace::ERR_ITERATOR_NEXT = "before calling the next() method."` [static, private]

Definition at line 209 of file SpatialTemporalTrace.hpp.

6.189.4.6 `const std::string SpatialTemporalTrace::ERR_NEXT_TIMEPOINT_VALUE_NOT_EXISTS = "The value of the last timepoint is the maximum value which can be represented by an unsigned long. Therefore a next timepoint value, which is greater than the value of the last timepoint, does not exist."` [static, private]

Definition at line 207 of file SpatialTemporalTrace.hpp.

Referenced by nextTimePointValueForLastTimePoint().

6.189.4.7 `const std::string SpatialTemporalTrace::ERR_RELATIVE_TIMEPOINT_INDEX_OUT_OF_BOUNDS_END = "is out of bounds for the given spatial temporal trace."` [static, private]

Definition at line 191 of file SpatialTemporalTrace.hpp.

Referenced by validateIndexRelativeToBeginIndex().

6.189.4.8 `const std::string SpatialTemporalTrace::ERR_RELATIVE_TIMEPOINT_INDEX_OUT_OF_BOUNDS_MIDDLE = "the provided timepoint index ("`  
[static, private]

Definition at line 190 of file SpatialTemporalTrace.hpp.

Referenced by validateIndexRelativeToBeginIndex().

```
6.189.4.9 const std::string SpatialTemporalTrace::ERR_RELATIVE_TIMEPOINT_INDEX_OUT_OF_BOUNDS_START = "Relative to the begin index (" [static, private]
```

Definition at line 189 of file SpatialTemporalTrace.hpp.

Referenced by validateIndexRelativeToBeginIndex().

```
6.189.4.10 const std::string SpatialTemporalTrace::ERR_TIMEPOINT_END_END = ")" [static, private]
```

Definition at line 201 of file SpatialTemporalTrace.hpp.

```
6.189.4.11 const std::string SpatialTemporalTrace::ERR_TIMEPOINT_END_MIDDLE = ") should be greater or equal to the start timepoint (" [static, private]
```

Definition at line 200 of file SpatialTemporalTrace.hpp.

```
6.189.4.12 const std::string SpatialTemporalTrace::ERR_TIMEPOINT_END_START = "The provided end timepoint (" [static, private]
```

Definition at line 199 of file SpatialTemporalTrace.hpp.

```
6.189.4.13 const std::string SpatialTemporalTrace::ERR_TIMEPOINT_VALUE_INVALID_END = ")" [static, private]
```

Definition at line 205 of file SpatialTemporalTrace.hpp.

Referenced by validateTimePointValue().

```
6.189.4.14 const std::string SpatialTemporalTrace::ERR_TIMEPOINT_VALUE_INVALID_MIDDLE = ") should be greater than the previously added timepoint value (" [static, private]
```

Definition at line 204 of file SpatialTemporalTrace.hpp.

Referenced by validateTimePointValue().

```
6.189.4.15 const std::string SpatialTemporalTrace::ERR_TIMEPOINT_VALUE_INVALID_START = "The current timepoint value (" [static, private]
```

Definition at line 203 of file SpatialTemporalTrace.hpp.

Referenced by validateTimePointValue().

6.189.4.16 `const std::string SpatialTemporalTrace::ERR_TIMEPOINT_VALUE_OUT_OF_BOUNDS_END = ") is out of bounds for the given spatial temporal trace."` [static, private]

Definition at line 197 of file SpatialTemporalTrace.hpp.

Referenced by validateValue().

6.189.4.17 `const std::string SpatialTemporalTrace::ERR_TIMEPOINT_VALUE_OUT_OF_BOUNDS_START = "The provided timepoint value ("` [static, private]

Definition at line 196 of file SpatialTemporalTrace.hpp.

Referenced by validateValue().

6.189.4.18 `bool multiscale::verification::SpatialTemporalTrace::isLastTimePointValueInitialised` [private]

Flag to indicate if the last time point value was initialised

Definition at line 25 of file SpatialTemporalTrace.hpp.

Referenced by initialise(), and validateTimePointValue().

6.189.4.19 `unsigned long multiscale::verification::SpatialTemporalTrace::lastTimePointValue` [private]

The value of the last added timepoint

Definition at line 23 of file SpatialTemporalTrace.hpp.

Referenced by initialise(), updateLastTimePointValue(), and validateTimePointValue().

6.189.4.20 `const int SpatialTemporalTrace::TIMEPOINT_INDEX_NOT_FOUND = -1` [static, private]

Definition at line 211 of file SpatialTemporalTrace.hpp.

Referenced by indexOfFirstTimePointGreaterOrEqualToValue(), and setSubTraceIndex().

6.189.4.21 `std::vector<TimePoint> multiscale::verification::SpatialTemporalTrace::timePoints` [private]

The array of time points

Definition at line 22 of file SpatialTemporalTrace.hpp.

Referenced by addTimePoint(), addTimePointsToSubTrace(), getTimePoint(), getTimePointReference(), indexOfFirstTimePointGreaterOrEqualToValue(), initialise(), length(), nextTimePointValue(), nextTimePointValueForLastTimePoint(), operator==(), setSubTraceIndex(), validateAbsoluteIndex(), validateIndexRelativeToBeginIndex(), and validateValue().

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

- SpatialTemporalTrace.hpp

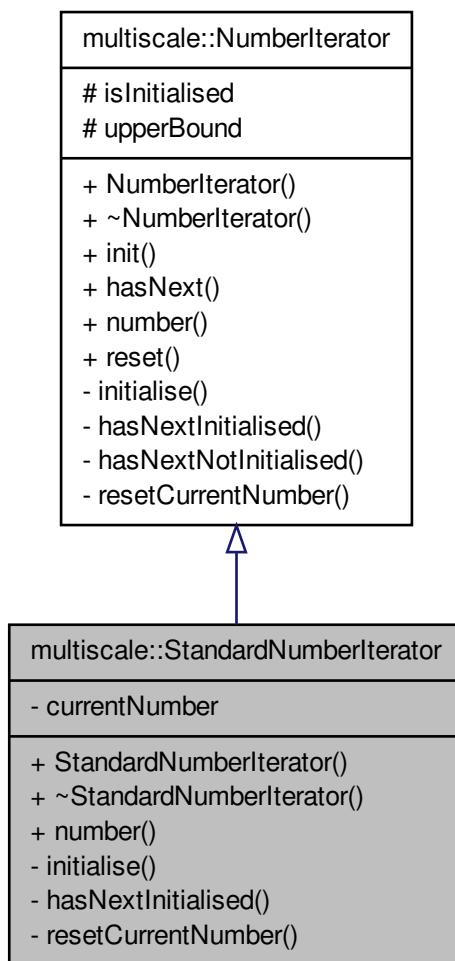
- SpatialTemporalTrace.cpp

## 6.190 multiscale::StandardNumberIterator Class Reference

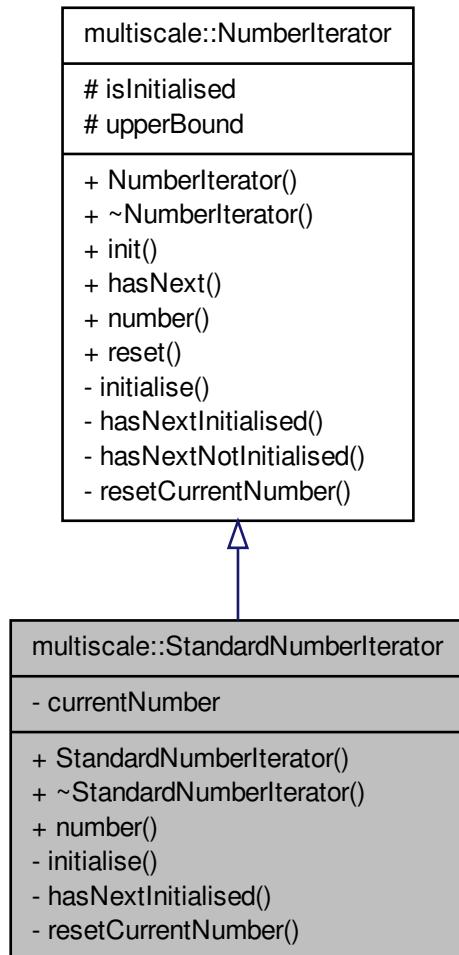
Iterator class starting at 1 and iterating over all natural numbers until the provided upper bound is reached.

```
#include <StandardNumberIterator.hpp>
```

Inheritance diagram for multiscale::StandardNumberIterator:



Collaboration diagram for multiscale::StandardNumberIterator:



## Public Member Functions

- `StandardNumberIterator (unsigned int upperBound)`
- `~StandardNumberIterator ()`
- `unsigned int number ()`

*Get the number pointed by the iterator.*

## Private Member Functions

- void [initialise \(\)](#)  
*Initialise the value of the current number.*
- bool [hasNextInitialised \(\)](#)  
*Check if there is a next number when in initialised state.*
- void [resetCurrentNumber \(\)](#)  
*Reset the current number to the initial value.*

## Private Attributes

- unsigned int [currentNumber](#)

### 6.190.1 Detailed Description

Iterator class starting at 1 and iterating over all natural numbers until the provided upper bound is reached.

Definition at line 10 of file StandardNumberIterator.hpp.

### 6.190.2 Constructor & Destructor Documentation

#### 6.190.2.1 StandardNumberIterator::StandardNumberIterator ( unsigned int *upperBound* )

Definition at line 6 of file StandardNumberIterator.cpp.

References initialise(), and multiscale::NumberIterator::reset().

#### 6.190.2.2 StandardNumberIterator::~StandardNumberIterator ( )

Definition at line 11 of file StandardNumberIterator.cpp.

### 6.190.3 Member Function Documentation

#### 6.190.3.1 bool StandardNumberIterator::hasNextInitialised ( ) [private, virtual]

Check if there is a next number when in initialised state.

Implements [multiscale::NumberIterator](#).

Definition at line 19 of file StandardNumberIterator.cpp.

References currentNumber, and multiscale::NumberIterator::upperBound.

**6.190.3.2 void StandardNumberIterator::initialise( ) [private, virtual]**

Initialise the value of the current number.

Implements [multiscale::NumberIterator](#).

Definition at line 17 of file StandardNumberIterator.cpp.

Referenced by StandardNumberIterator().

**6.190.3.3 unsigned int StandardNumberIterator::number( ) [virtual]**

Get the number pointed by the iterator.

Implements [multiscale::NumberIterator](#).

Definition at line 13 of file StandardNumberIterator.cpp.

References currentNumber.

**6.190.3.4 void StandardNumberIterator::resetCurrentNumber( ) [private, virtual]**

Reset the current number to the initial value.

Implements [multiscale::NumberIterator](#).

Definition at line 29 of file StandardNumberIterator.cpp.

References currentNumber.

#### 6.190.4 Member Data Documentation

**6.190.4.1 unsigned int multiscale::StandardNumberIterator::currentNumber [private]**

The current number

Definition at line 14 of file StandardNumberIterator.hpp.

Referenced by hasNextInitialised(), number(), and resetCurrentNumber().

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

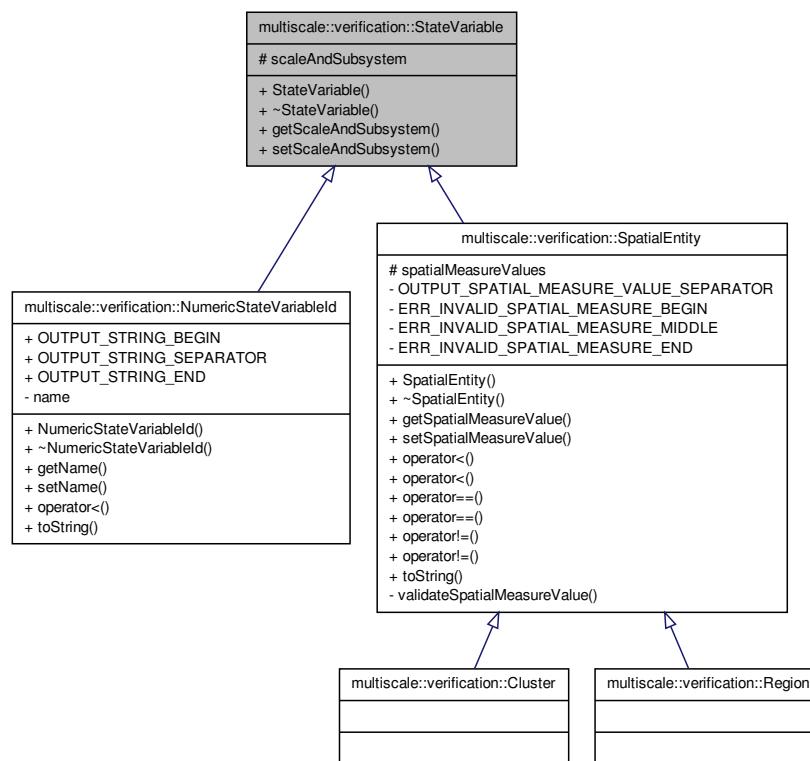
- StandardNumberIterator.hpp
- StandardNumberIterator.cpp

## 6.191 multiscale::verification::StateVariable Class Reference

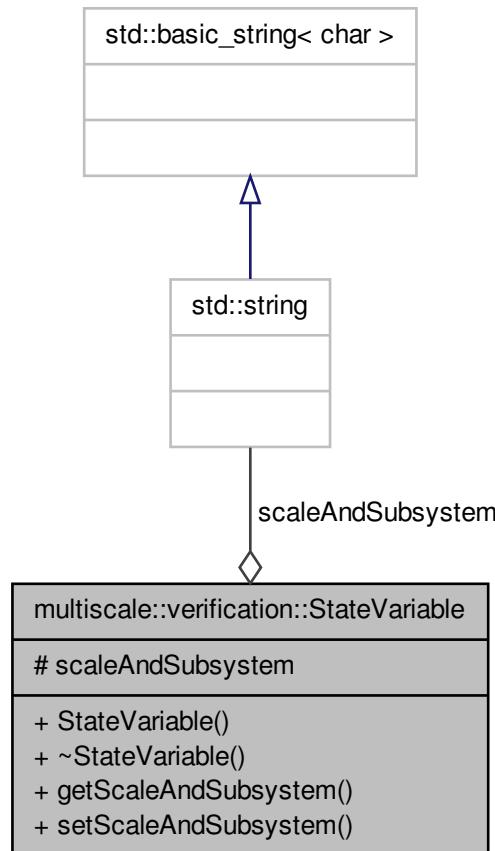
Class for representing a state variable.

```
#include <StateVariable.hpp>
```

Inheritance diagram for multiscale::verification::StateVariable:



Collaboration diagram for multiscale::verification::StateVariable:



## Public Member Functions

- `StateVariable ()`
- `~StateVariable ()`
- `std::string getScaleAndSubsystem () const`  
*Get the scale and subsystem of the state variable.*
- `void setScaleAndSubsystem (const std::string &scaleAndSubsystem)`  
*Set the scale and subsystem of the state variable.*

## Protected Attributes

- std::string [scaleAndSubsystem](#)

### 6.191.1 Detailed Description

Class for representing a state variable.

Definition at line 12 of file StateVariable.hpp.

### 6.191.2 Constructor & Destructor Documentation

#### 6.191.2.1 StateVariable::StateVariable( )

Definition at line 6 of file StateVariable.cpp.

#### 6.191.2.2 StateVariable::~StateVariable( )

Definition at line 8 of file StateVariable.cpp.

### 6.191.3 Member Function Documentation

#### 6.191.3.1 std::string StateVariable::getScaleAndSubsystem( ) const

Get the scale and subsystem of the state variable.

Definition at line 10 of file StateVariable.cpp.

References scaleAndSubsystem.

Referenced by multiscale::verification::SpatialTemporalDataWriter::constructPropertyTreeFromNumericStateVariable().

#### 6.191.3.2 void StateVariable::setScaleAndSubsystem( const std::string & [scaleAndSubsystem](#) )

Set the scale and subsystem of the state variable.

##### Parameters

|                                   |                                                                          |
|-----------------------------------|--------------------------------------------------------------------------|
| <a href="#">scaleAndSubsystem</a> | The scale and subsystem which will be associated with the state variable |
|-----------------------------------|--------------------------------------------------------------------------|

Definition at line 14 of file StateVariable.cpp.

References scaleAndSubsystem.

#### 6.191.4 Member Data Documentation

##### 6.191.4.1 std::string multiscale::verification::StateVariable::scaleAndSubsystem [protected]

The scale and subsystem of the state variable

Definition at line 16 of file StateVariable.hpp.

Referenced by getScaleAndSubsystem(), multiscale::verification::NumericStateVariableId::NumericStateVariableId(), multiscale::verification::NumericStateVariableId::operator<(), multiscale::verification::SpatialEntity::operator<(), multiscale::verification::SpatialEntity::operator==( ), setScaleAndSubsystem(), and multiscale::verification::NumericStateVariableId::toString().

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

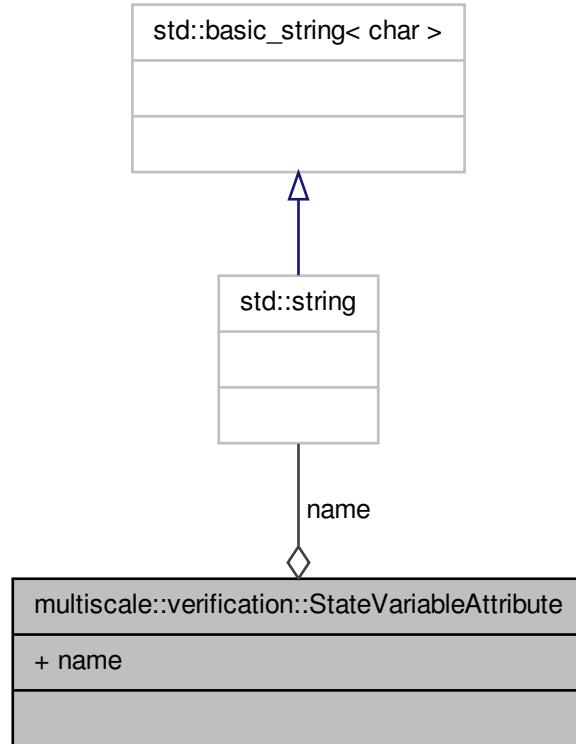
- StateVariable.hpp
  
- StateVariable.cpp

## 6.192 multiscale::verification::StateVariableAttribute Class Reference

Class for representing a state variable attribute.

```
#include <StateVariableAttribute.hpp>
```

Collaboration diagram for multiscale::verification::StateVariableAttribute:



## Public Attributes

- `std::string name`

### 6.192.1 Detailed Description

Class for representing a state variable attribute.

Definition at line 14 of file `StateVariableAttribute.hpp`.

### 6.192.2 Member Data Documentation

**6.192.2.1 std::string multiscale::verification::StateVariableAttribute::name**

Name of the state variable

Definition at line 18 of file StateVariableAttribute.hpp.

Referenced by multiscale::verification::NumericStateVariableEvaluator::evaluate().

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

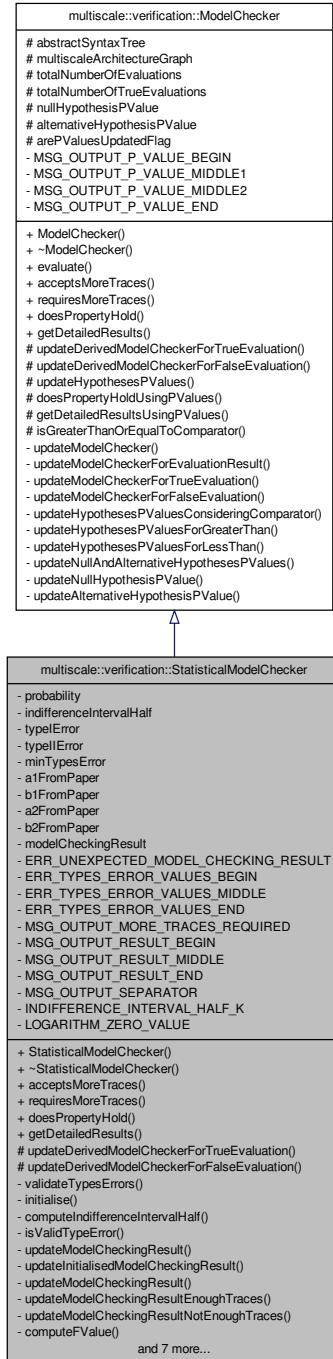
- StateVariableAttribute.hpp

**6.193 multiscale::verification::StatisticalModelChecker Class Reference**

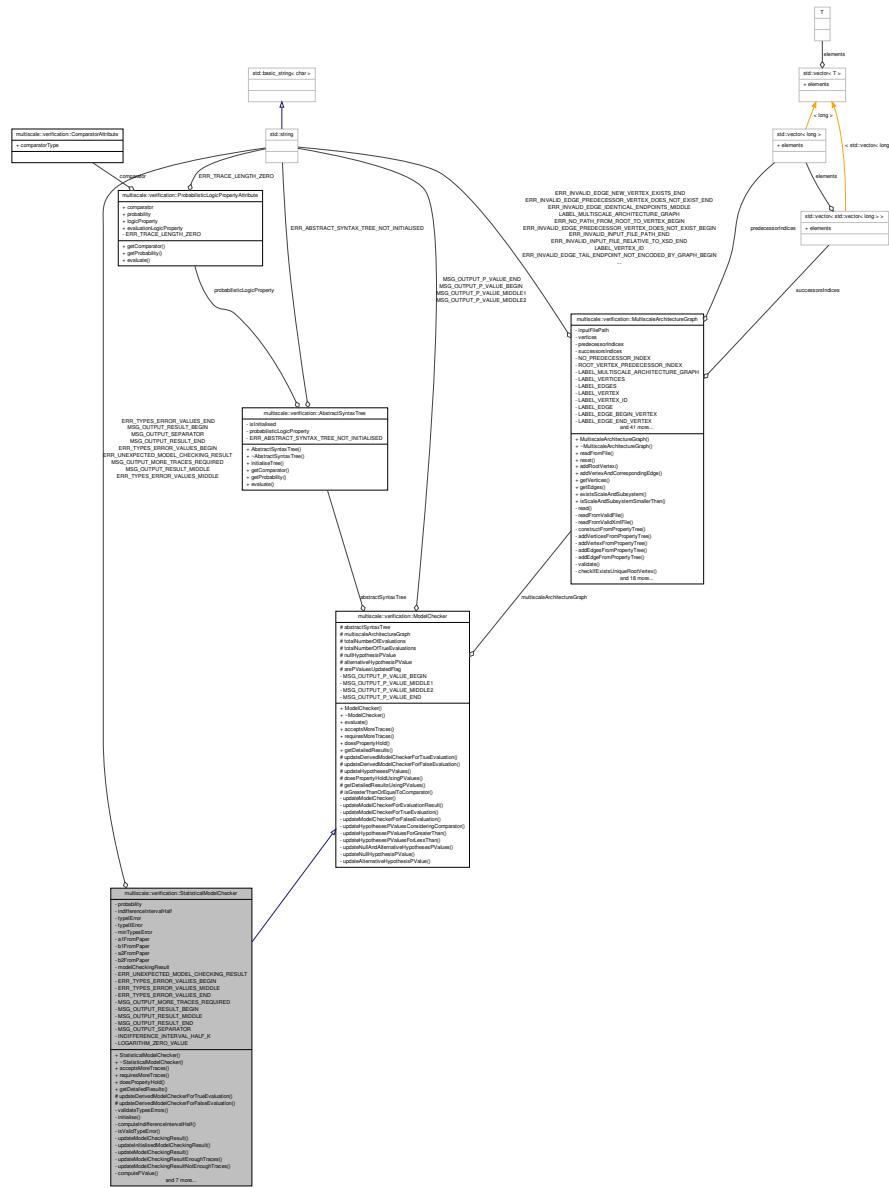
Class used to run statistical model checking tasks.

```
#include <StatisticalModelChecker.hpp>
```

Inheritance diagram for multiscale::verification::StatisticalModelChecker:



Collaboration diagram for multiscale::verification::StatisticalModelChecker:



## Public Member Functions

- `StatisticalModelChecker` (`const AbstractSyntaxTree &abstractSyntaxTree, const MultiscaleArchitectureGraph &multiscaleArchitectureGraph, double typeIError, double typeIIError`)
  - `~StatisticalModelChecker` ()

- bool `acceptsMoreTraces ()` override  
*Check if more traces are accepted for evaluating the logic property.*
- bool `requiresMoreTraces ()` override  
*Check if more traces are required for evaluating the logic property.*
- bool `doesPropertyHold ()` override  
*Check if the given property holds.*
- std::string `getDetailedResults ()` override  
*Get the detailed description of the results.*

### Protected Member Functions

- void `updateDerivedModelCheckerForTrueEvaluation ()` override  
*Update the results of the derived model checker type considering that the logic property was evaluated to true for the last trace.*
- void `updateDerivedModelCheckerForFalseEvaluation ()` override  
*Update the results of the derived model checker type considering that the logic property was evaluated to false for the last trace.*

### Private Member Functions

- void `validateTypesErrors (double typeIError, double typeIIError)`  
*Validate the probability of type I and type II errors to occur.*
- void `initialise ()`  
*Initialisation of some of the class members.*
- double `computeIndifferenceIntervalHalf (double probability)`  
*Compute the value of the indifference interval half considering the given probability.*
- bool `isValidTypeError (double typeError)`  
*Check if the given type I/II error probability is valid.*
- void `updateModelCheckingResult ()`  
*Update the result of the model checking task.*
- void `updateInitialisedModelCheckingResult ()`  
*Update the result of the model checking task which was already initialised.*
- void `updateModelCheckingResult (double f, double fPrime)`  
*Update the result of the model checking task considering the given values.*
- void `updateModelCheckingResultEnoughTraces (double f, double fPrime)`  
*Update the result of the model checking task considering the given values when enough traces have been provided.*
- void `updateModelCheckingResultNotEnoughTraces ()`  
*Update the result of the model checking task when not enough traces were provided.*
- double `computeFValue ()`  
*Compute the value of f (from original paper)*
- double `computeFValueFirstTerm ()`  
*Compute the value of the first term of f (from original paper)*

- double `computeFValueSecondTerm ()`  
*Compute the value of the second term of f (from original paper)*
- double `computeFPrimeValue ()`  
*Compute the value of f' (from original paper)*
- double `computeFPrimeValueFirstTerm ()`  
*Compute the value of the first term of f' (from original paper)*
- double `computeFPrimeValueSecondTerm ()`  
*Compute the value of the second term of f' (from original paper)*
- bool `doesPropertyHoldConsideringResult ()`  
*Check if the given property holds considering the obtained model checking result.*
- bool `doesPropertyHoldConsideringProbabilityComparator (bool isNullHypothesis-True)`  
*Check if the given property holds considering the obtained answer and probability comparator (i.e. <=, >=)*
- std::string `getDetailedUpdatedResults ()`  
*Get the detailed description of the updated results.*

### Private Attributes

- double `probability`
- double `indifferenceIntervalHalf`
- double `typeIError`
- double `typeIIError`
- double `minTypesError`
- double `a1FromPaper`
- double `b1FromPaper`
- double `a2FromPaper`
- double `b2FromPaper`
- StatisticalModelCheckingResult `modelCheckingResult`

### Static Private Attributes

- static const std::string `ERR_UNEXPECTED_MODEL_CHECKING_RESULT` = "- An invalid statistical model checking result was obtained. Please check source code."
- static const std::string `ERR_TYPES_ERROR_VALUES_BEGIN` = "The provided probabilities of type I and type II errors ("
- static const std::string `ERR_TYPES_ERROR_VALUES_MIDDLE` = ", "
- static const std::string `ERR_TYPES_ERROR_VALUES_END` = ") should be greater than zero and less or equal to 1. Please change."
- static const std::string `MSG_OUTPUT_MORE_TRACES_REQUIRED` = "More traces are required to provide a true/false answer assuming the given probabilities of type I and type II errors. Probabilistic black-box model checking was used instead to provide an answer."

- static const std::string **MSG\_OUTPUT\_RESULT\_BEGIN** = "The provided answer is given for the [probability](#) of type I errors = "
- static const std::string **MSG\_OUTPUT\_RESULT\_MIDDLE** = " and the [probability](#) of type II errors = "
- static const std::string **MSG\_OUTPUT\_RESULT\_END** = ""
- static const std::string **MSG\_OUTPUT\_SEPARATOR** = " "
- static const unsigned int **INDIFFERENCE\_INTERVAL\_HALF\_K** = (std::numeric\_limits<unsigned int>::max() >> 1)  
*The value of this constant should be much greater than 1.*
- static const double **LOGARITHM\_ZERO\_VALUE** = (std::numeric\_limits<double>::lowest() / 1E+10)  
*The value of this constant should be a large negative number.*

### 6.193.1 Detailed Description

Class used to run statistical model checking tasks.

The implementation of this class is (partially) based on the algorithms described in the following paper:

C. H. Koh, S. K. Palaniappan, P. S. Thiagarajan, and L. Wong, 'Improved statistical model checking methods for pathway analysis', BMC Bioinformatics, vol. 13, no. Suppl 17, p. S15, Dec. 2012.

In our implementation the variables in the original paper (right hand side of the assignments) have been given the following new names (left hand side of assignments):

`probability =  $\theta$`

`indifference =  $\delta$`

`typeIError =  $\alpha$`

`typeIIError =  $\beta$`

`minTypesError =  $\gamma$`

`totalNumberOfEvaluations =  $n$`

`totalNumberOfTrueEvaluations =  $d$`

Definition at line 50 of file StatisticalModelChecker.hpp.

### 6.193.2 Constructor & Destructor Documentation

6.193.2.1 **StatisticalModelChecker::StatisticalModelChecker**  
`( const AbstractSyntaxTree & abstractSyntaxTree, const MultiscaleArchitectureGraph & multiscaleArchitectureGraph, double typeIError, double typeIIError )`

Definition at line 13 of file StatisticalModelChecker.cpp.

References `initialise()`, `minTypesError`, `typeIError`, `typeIIError`, and `validateTypesErrors()`.

**6.193.2.2 StatisticalModelChecker::~StatisticalModelChecker( )**

Definition at line 30 of file StatisticalModelChecker.cpp.

**6.193.3 Member Function Documentation****6.193.3.1 bool StatisticalModelChecker::acceptsMoreTraces( ) [override, virtual]**

Check if more traces are accepted for evaluating the logic property.

Implements [multiscale::verification::ModelChecker](#).

Definition at line 32 of file StatisticalModelChecker.cpp.

References [modelCheckingResult](#), and [updateModelCheckingResult\(\)](#).

Referenced by [requiresMoreTraces\(\)](#).

**6.193.3.2 double StatisticalModelChecker::computeFPrimeValue( ) [private]**

Compute the value of f' (from original paper)

Definition at line 165 of file StatisticalModelChecker.cpp.

References [computeFPrimeValueFirstTerm\(\)](#), [computeFPrimeValueSecondTerm\(\)](#), [multiscale::verification::ModelChecker::totalNumberOfEvaluations](#), and [multiscale::verification::ModelChecker::totalNumberOfTrueEvaluations](#).

Referenced by [updateInitialisedModelCheckingResult\(\)](#).

**6.193.3.3 double StatisticalModelChecker::computeFPrimeValueFirstTerm( ) [private]**

Compute the value of the first term of f' (from original paper)

If the value inside the logarithm is equal to zero than the returned value is equal to -LOGARITHM\_ZERO\_VALUE. Otherwise the value of the logarithm is computed and returned.

Definition at line 175 of file StatisticalModelChecker.cpp.

References [multiscale::Numeric::almostEqual\(\)](#), [indifferenceIntervalHalf](#), LOGARITHM\_ZERO\_VALUE, and [probability](#).

Referenced by [computeFPrimeValue\(\)](#).

**6.193.3.4 double StatisticalModelChecker::computeFPrimeValueSecondTerm( ) [private]**

Compute the value of the second term of f' (from original paper)

If the value inside the logarithm is equal to zero than the returned value is equal to -LOGARITHM\_ZERO\_VALUE. Otherwise the value of the logarithm is computed and returned.

Definition at line 184 of file StatisticalModelChecker.cpp.

References multiscale::Numeric::almostEqual(), indifferenceIntervalHalf, LOGARITHM\_ZERO\_VALUE, and probability.

Referenced by computeFPrimeValue().

#### 6.193.3.5 double StatisticalModelChecker::computeFValue( ) [private]

Compute the value of f (from original paper)

Definition at line 137 of file StatisticalModelChecker.cpp.

References computeFValueFirstTerm(), computeFValueSecondTerm(), multiscale::verification::ModelChecker::totalNumberOfEvaluations, and multiscale::verification::ModelChecker::totalNumberOfTrueEvaluations.

Referenced by updateInitialisedModelCheckingResult().

#### 6.193.3.6 double StatisticalModelChecker::computeFValueFirstTerm( ) [private]

Compute the value of the first term of f (from original paper)

If the value inside the logarithm is equal to zero than the returned value is equal to -LOGARITHM\_ZERO\_VALUE. Otherwise the value of the logarithm is computed and returned.

Definition at line 147 of file StatisticalModelChecker.cpp.

References multiscale::Numeric::almostEqual(), indifferenceIntervalHalf, LOGARITHM\_ZERO\_VALUE, and probability.

Referenced by computeFValue().

#### 6.193.3.7 double StatisticalModelChecker::computeFValueSecondTerm( ) [private]

Compute the value of the second term of f (from original paper)

If the value inside the logarithm is equal to zero than the returned value is equal to -LOGARITHM\_ZERO\_VALUE. Otherwise the value of the logarithm is computed and returned.

Definition at line 156 of file StatisticalModelChecker.cpp.

References multiscale::Numeric::almostEqual(), indifferenceIntervalHalf, LOGARITHM\_ZERO\_VALUE, and probability.

Referenced by computeFValue().

6.193.3.8 double StatisticalModelChecker::computeIndifferenceIntervalHalf ( double *probability* ) [private]

Compute the value of the indifference interval half considering the given probability.

*indifferenceIntervalHalf* = max(0, min(*probability*, 1 - *probability*) - *eps*)

Parameters

|                    |                              |
|--------------------|------------------------------|
| <i>probability</i> | The value of the probability |
|--------------------|------------------------------|

Definition at line 79 of file StatisticalModelChecker.cpp.

References INDIFFERENCE\_INTERVAL\_HALF\_K.

Referenced by initialise().

6.193.3.9 bool StatisticalModelChecker::doesPropertyHold ( ) [override, virtual]

Check if the given property holds.

Implements [multiscale::verification::ModelChecker](#).

Definition at line 42 of file StatisticalModelChecker.cpp.

References doesPropertyHoldConsideringResult(), and updateModelCheckingResult().

6.193.3.10 bool StatisticalModelChecker::doesPropertyHoldConsideringProbabilityComparator ( bool *isNullHypothesisTrue* ) [private]

Check if the given property holds considering the obtained answer and probability comparator (i.e.  $\leq$ ,  $\geq$ )

For queries of type : a)  $P \geq \theta[\phi]$  the *isNullHypothesisTrue* flag value is returned b)  $P \leq \theta[\phi]$  the *!isNullHypothesisTrue* flag value is returned

Parameters

|                             |                                                                                    |
|-----------------------------|------------------------------------------------------------------------------------|
| <i>isNullHypothesisTrue</i> | Flag indicating if the null hypothesis is true considering a $P \geq [\phi]$ query |
|-----------------------------|------------------------------------------------------------------------------------|

Definition at line 212 of file StatisticalModelChecker.cpp.

References [multiscale::verification::ModelChecker::isGreaterThanOrEqualToComparator\(\)](#).

Referenced by doesPropertyHoldConsideringResult().

6.193.3.11 **bool StatisticalModelChecker::doesPropertyHoldConsideringResult( )**  
[private]

Check if the given property holds considering the obtained model checking result.

Definition at line 193 of file StatisticalModelChecker.cpp.

References doesPropertyHoldConsideringProbabilityComparator(), multiscale::verification::ModelChecker::doesPropertyHoldUsingPValues(), ERR\_UNEXPECTED\_MODEL\_CHECKING\_RESULT, and modelCheckingResult.

Referenced by doesPropertyHold().

6.193.3.12 **std::string StatisticalModelChecker::getDetailedResults( )**  
[override, virtual]

Get the detailed description of the results.

Implements [multiscale::verification::ModelChecker](#).

Definition at line 48 of file StatisticalModelChecker.cpp.

References getDetailedUpdatedResults(), and updateModelCheckingResult().

6.193.3.13 **std::string StatisticalModelChecker::getDetailedUpdatedResults( )**  
[private]

Get the detailed description of the updated results.

Definition at line 220 of file StatisticalModelChecker.cpp.

References multiscale::verification::ModelChecker::getDetailedResultsUsingPValues(), modelCheckingResult, MSG\_OUTPUT\_MORE\_TRACES\_REQUIRED, MSG\_OUTPUT\_RESULT\_BEGIN, MSG\_OUTPUT\_RESULT\_END, MSG\_OUTPUT\_RESULT\_MIDDLE, MSG\_OUTPUT\_SEPARATOR, multiscale::StringManipulator::toString(), typeError, and typeIIError.

Referenced by getDetailedResults().

6.193.3.14 **void StatisticalModelChecker::initialise( ) [private]**

Initialisation of some of the class members.

Definition at line 69 of file StatisticalModelChecker.cpp.

References a1FromPaper, a2FromPaper, multiscale::verification::ModelChecker::abstractSyntaxTree, b1FromPaper, b2FromPaper, computeIndifferenceIntervalHalf(), multiscale::verification::AbstractSyntaxTree::getProbability(), indifferenceIntervalHalf, minTypesError, probability, typeError, and typeIIError.

Referenced by StatisticalModelChecker().

**6.193.3.15 bool StatisticalModelChecker::isValidTypeError ( double typeError )**  
 [private]

Check if the given type I/II error probability is valid.

The probability of the type I/II error to occur should be greater than zero and less than one

**Parameters**

|                  |                                               |
|------------------|-----------------------------------------------|
| <i>typeError</i> | The probability of a type I/II error to occur |
|------------------|-----------------------------------------------|

Definition at line 86 of file StatisticalModelChecker.cpp.

Referenced by validateTypesErrors().

**6.193.3.16 bool StatisticalModelChecker::requiresMoreTraces ( )** [override,  
 virtual]

Check if more traces are required for evaluating the logic property.

Implements [multiscale::verification::ModelChecker](#).

Definition at line 38 of file StatisticalModelChecker.cpp.

References acceptsMoreTraces().

**6.193.3.17 void StatisticalModelChecker::updateDerivedModelChecker-  
 ForFalseEvaluation ( )** [override, protected,  
 virtual]

Update the results of the derived model checker type considering that the logic property was evaluated to false for the last trace.

Do not do anything

Implements [multiscale::verification::ModelChecker](#).

Definition at line 56 of file StatisticalModelChecker.cpp.

**6.193.3.18 void StatisticalModelChecker::updateDerivedModelChecker-  
 ForTrueEvaluation ( )** [override, protected,  
 virtual]

Update the results of the derived model checker type considering that the logic property was evaluated to true for the last trace.

Do not do anything

Implements [multiscale::verification::ModelChecker](#).

Definition at line 54 of file StatisticalModelChecker.cpp.

---

**6.193.3.19 void StatisticalModelChecker::updateInitialisedModelCheckingResult ( ) [private]**

Update the result of the model checking task which was already initialised.

The name and semantics of the local variables a1, b1, a2, b2, f, fPrime, n, d correspond to the name and semantics of the variables used in the original paper.

Definition at line 101 of file StatisticalModelChecker.cpp.

References computeFPrimeValue(), computeFValue(), and updateModelCheckingResult().

Referenced by updateModelCheckingResult().

**6.193.3.20 void StatisticalModelChecker::updateModelCheckingResult ( ) [private]**

Update the result of the model checking task.

Definition at line 93 of file StatisticalModelChecker.cpp.

References modelCheckingResult, multiscale::verification::UNDECIDED, and updateInitialisedModelCheckingResult().

Referenced by acceptsMoreTraces(), doesPropertyHold(), getDetailedResults(), and updateInitialisedModelCheckingResult().

**6.193.3.21 void StatisticalModelChecker::updateModelCheckingResult ( double *f*, double *fPrime* ) [private]**

Update the result of the model checking task considering the given values.

#### Parameters

|               |                                           |
|---------------|-------------------------------------------|
| <i>f</i>      | The value of f (from the original paper)  |
| <i>fPrime</i> | The value of f' (from the original paper) |

Definition at line 108 of file StatisticalModelChecker.cpp.

References a1FromPaper, a2FromPaper, b1FromPaper, b2FromPaper, updateModelCheckingResultEnoughTraces(), and updateModelCheckingResultNotEnoughTraces().

**6.193.3.22 void StatisticalModelChecker::updateModelCheckingResultEnoughTraces ( double *f*, double *fPrime* ) [private]**

Update the result of the model checking task considering the given values when enough traces have been provided.

**Parameters**

|               |                                           |
|---------------|-------------------------------------------|
| <i>f</i>      | The value of f (from the original paper)  |
| <i>fPrime</i> | The value of f' (from the original paper) |

Definition at line 121 of file StatisticalModelChecker.cpp.

References a1FromPaper, a2FromPaper, b1FromPaper, b2FromPaper, multiscale::verification::FALSE, indifferenceIntervalHalf, modelCheckingResult, multiscale::verification::TRUE, and multiscale::verification::UNDECIDED.

Referenced by updateModelCheckingResult().

#### 6.193.3.23 void StatisticalModelChecker::updateModelCheckingResultNotEnoughTraces( ) [private]

Update the result of the model checking task when not enough traces were provided.

Definition at line 133 of file StatisticalModelChecker.cpp.

References modelCheckingResult, and multiscale::verification::MORE\_TRACES\_REQUIRED.

Referenced by updateModelCheckingResult().

#### 6.193.3.24 void StatisticalModelChecker::validateTypesErrors ( double typeIError, double typeIIError ) [private]

Validate the probability of type I and type II errors to occur.

The probability of type I and type II errors to occur should be greater than zero and less than one

**Parameters**

|                    |                                             |
|--------------------|---------------------------------------------|
| <i>typeIError</i>  | The probability of a type I error to occur  |
| <i>typeIIError</i> | The probability of a type II error to occur |

Definition at line 58 of file StatisticalModelChecker.cpp.

References ERR\_TYPES\_ERROR\_VALUES\_BEGIN, ERR\_TYPES\_ERROR\_VALUES\_END, ERR\_TYPES\_ERROR\_VALUES\_MIDDLE, isValidTypeError(), and multiscale::StringManipulator::toString().

Referenced by StatisticalModelChecker().

### 6.193.4 Member Data Documentation

#### 6.193.4.1 double multiscale::verification::StatisticalModelChecker::a1FromPaper [private]

The variable A1 (from the original paper)

Definition at line 64 of file StatisticalModelChecker.hpp.

Referenced by initialise(), updateModelCheckingResult(), and updateModelCheckingResultEnoughTraces().

**6.193.4.2 double multiscale::verification::StatisticalModelChecker::a2FromPaper  
[private]**

The variable A2 (from the original paper)

Definition at line 66 of file StatisticalModelChecker.hpp.

Referenced by initialise(), updateModelCheckingResult(), and updateModelCheckingResultEnoughTraces().

**6.193.4.3 double multiscale::verification::StatisticalModelChecker::b1FromPaper  
[private]**

The variable B1 (from the original paper)

Definition at line 65 of file StatisticalModelChecker.hpp.

Referenced by initialise(), updateModelCheckingResult(), and updateModelCheckingResultEnoughTraces().

**6.193.4.4 double multiscale::verification::StatisticalModelChecker::b2FromPaper  
[private]**

The variable B2 (from the original paper)

Definition at line 67 of file StatisticalModelChecker.hpp.

Referenced by initialise(), updateModelCheckingResult(), and updateModelCheckingResultEnoughTraces().

**6.193.4.5 const std::string StatisticalModelChecker::ERR\_TYPES\_ERROR\_VALUE-  
S\_BEGIN = "The provided probabilities of type I and type II errors (" [static,  
private]**

Definition at line 216 of file StatisticalModelChecker.hpp.

Referenced by validateTypesErrors().

**6.193.4.6 const std::string StatisticalModelChecker::ERR\_TYPES\_ERROR\_VALUE-  
S\_END = ") should be greater than zero and less or equal to 1. Please change." [static,  
private]**

Definition at line 218 of file StatisticalModelChecker.hpp.

Referenced by validateTypesErrors().

```
6.193.4.7 const std::string StatisticalModelChecker::ERR_TYP-
ES_ERROR_VALUES_MIDDLE = ", " [static,
private]
```

Definition at line 217 of file StatisticalModelChecker.hpp.

Referenced by validateTypesErrors().

```
6.193.4.8 const std::string StatisticalModelChecker::ERR_UNEXPECTED_MODEL_-
CHECKING_RESULT = "An invalid statistical model checking result was obtained.
Please check source code." [static, private]
```

Definition at line 214 of file StatisticalModelChecker.hpp.

Referenced by doesPropertyHoldConsideringResult().

```
6.193.4.9 const unsigned int StatisticalModelChecker::INDIFFERENCE_INTERVAL_-
HALF_K = (std::numeric_limits<unsigned int>::max() >> 1) [static,
private]
```

The value of this constant should be much greater than 1.

Definition at line 228 of file StatisticalModelChecker.hpp.

Referenced by computeIndifferenceIntervalHalf().

```
6.193.4.10 double multiscale::verification::StatisticalModelChecker::indifference-
IntervalHalf [private]
```

Half of the size of the indifference interval

Definition at line 57 of file StatisticalModelChecker.hpp.

Referenced by computeFPrimeValueFirstTerm(), computeFPrimeValueSecondTerm(),  
computeFValueFirstTerm(), computeFValueSecondTerm(), initialise(), and update-  
ModelCheckingResultEnoughTraces().

```
6.193.4.11 const double StatisticalModelChecker::LOGARITHM_ZERO_VALUE =
(std::numeric_limits<double>::lowest() / 1E+10) [static, private]
```

The value of this constant should be a large negative number.

The value obtained when computing log(0)

This value will be further multiplied by non-negative integer numbers. In order to avoid  
overflow the lowest double value is divided by 1E10.

Definition at line 230 of file StatisticalModelChecker.hpp.

Referenced by computeFPrimeValueFirstTerm(), computeFPrimeValueSecondTerm(),  
computeFValueFirstTerm(), and computeFValueSecondTerm().

6.193.4.12 **double multiscale::verification::StatisticalModelChecker::minTypes-Error** [private]

The minimum probability of type I and type II errors to occur

Definition at line 62 of file StatisticalModelChecker.hpp.

Referenced by initialise(), and StatisticalModelChecker().

6.193.4.13 **StatisticalModelCheckingResult multiscale::verification-::StatisticalModelChecker::modelCheckingResult** [private]

The result of the model checking task

Definition at line 70 of file StatisticalModelChecker.hpp.

Referenced by acceptsMoreTraces(), doesPropertyHoldConsideringResult(), getDetailedUpdatedResults(), updateModelCheckingResult(), updateModelCheckingResultEnoughTraces(), and updateModelCheckingResultNotEnoughTraces().

6.193.4.14 **const std::string StatisticalModelChecker::MSG\_OUTPUT\_MORE\_-TRACES\_REQUIRED** = "More traces are required to provide a true/false answer assuming the given probabilities of type I and type II errors. Probabilistic black-box model checking was used instead to provide an answer." [static, private]

Definition at line 220 of file StatisticalModelChecker.hpp.

Referenced by getDetailedUpdatedResults().

6.193.4.15 **const std::string StatisticalModelChecker::MSG\_OUTPUT\_RESULT\_B-BEGIN** = "The provided answer is given for the probability of type I errors = " [static, private]

Definition at line 222 of file StatisticalModelChecker.hpp.

Referenced by getDetailedUpdatedResults().

6.193.4.16 **const std::string StatisticalModelChecker::MSG\_OUTPUT\_RESULT\_END** = "" [static, private]

Definition at line 224 of file StatisticalModelChecker.hpp.

Referenced by getDetailedUpdatedResults().

6.193.4.17 `const std::string StatisticalModelChecker::MSG_OUTPUT_RESULT_MIDDLE = " and the probability of type II errors = " [static, private]`

Definition at line 223 of file StatisticalModelChecker.hpp.

Referenced by `getDetailedUpdatedResults()`.

6.193.4.18 `const std::string StatisticalModelChecker::MSG_OUTPUT_SEPARATOR = " " [static, private]`

Definition at line 226 of file StatisticalModelChecker.hpp.

Referenced by `getDetailedUpdatedResults()`.

6.193.4.19 `double multiscale::verification::StatisticalModelChecker::probability [private]`

The probability specified by the user for the logic property to be evaluated

Definition at line 54 of file StatisticalModelChecker.hpp.

Referenced by `computeFPrimeValueFirstTerm()`, `computeFPrimeValueSecondTerm()`, `computeFValueFirstTerm()`, `computeFValueSecondTerm()`, and `initialise()`.

6.193.4.20 `double multiscale::verification::StatisticalModelChecker::typeIError [private]`

The probability of type I errors to occur

Definition at line 59 of file StatisticalModelChecker.hpp.

Referenced by `getDetailedUpdatedResults()`, `initialise()`, and `StatisticalModelChecker()`.

6.193.4.21 `double multiscale::verification::StatisticalModelChecker::typeIIError [private]`

The probability of type II errors to occur

Definition at line 60 of file StatisticalModelChecker.hpp.

Referenced by `getDetailedUpdatedResults()`, `initialise()`, and `StatisticalModelChecker()`.

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

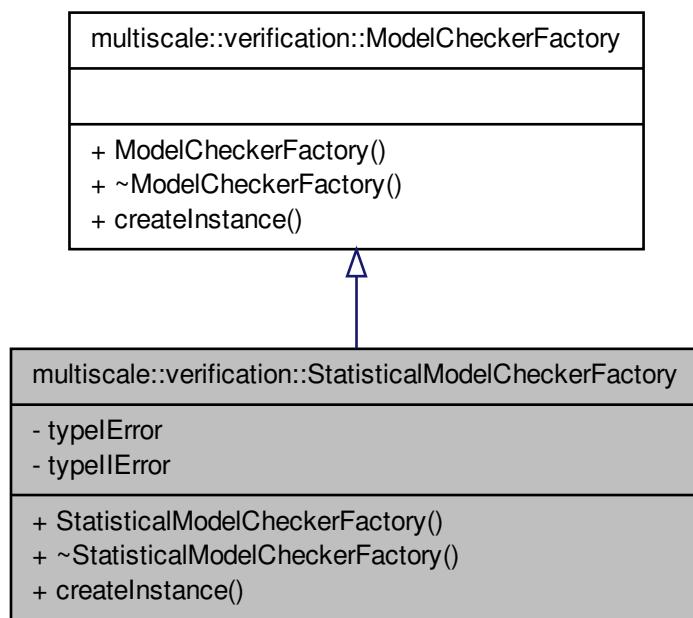
- StatisticalModelChecker.hpp
- StatisticalModelChecker.cpp

## 6.194 multiscale::verification::StatisticalModelCheckerFactory - Class Reference

Class for creating [StatisticalModelChecker](#) instances.

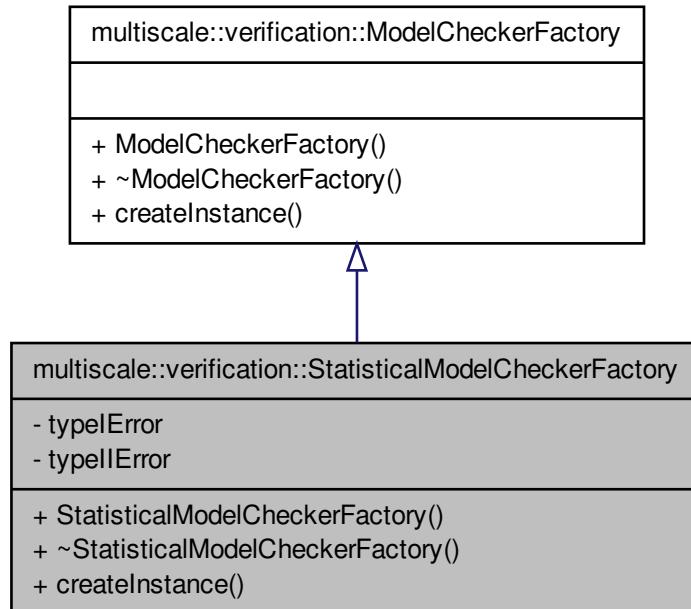
```
#include <StatisticalModelCheckerFactory.hpp>
```

Inheritance diagram for multiscale::verification::StatisticalModelCheckerFactory:



## **6.194 multiscale::verification::StatisticalModelCheckerFactory Class Reference**

Collaboration diagram for multiscale::verification::StatisticalModelCheckerFactory:



### **Public Member Functions**

- `StatisticalModelCheckerFactory (double typeIError, double typeIIError)`
- `~StatisticalModelCheckerFactory ()`
- `std::shared_ptr< ModelChecker > createInstance (const AbstractSyntaxTree &abstractSyntaxTree, const MultiscaleArchitectureGraph &multiscaleArchitectureGraph) override`

*Create an instance of [StatisticalModelChecker](#).*

### **Private Attributes**

- `double typeIError`
- `double typeIIError`

#### **6.194.1 Detailed Description**

Class for creating [StatisticalModelChecker](#) instances.

Definition at line 12 of file StatisticalModelCheckerFactory.hpp.

#### 6.194.2 Constructor & Destructor Documentation

6.194.2.1 **StatisticalModelCheckerFactory::StatisticalModelCheckerFactory ( double typeIError, double typeIIError )**

Definition at line 7 of file StatisticalModelCheckerFactory.cpp.

6.194.2.2 **StatisticalModelCheckerFactory::~StatisticalModelCheckerFactory ( )**

Definition at line 12 of file StatisticalModelCheckerFactory.cpp.

#### 6.194.3 Member Function Documentation

6.194.3.1 **std::shared\_ptr< ModelChecker > StatisticalModelCheckerFactory-  
::createInstance ( const AbstractSyntaxTree & abstractSyntaxTree,  
const MultiscaleArchitectureGraph & multiscaleArchitectureGraph )  
[override, virtual]**

Create an instance of [StatisticalModelChecker](#).

##### Parameters

|                                      |                                                                                                   |
|--------------------------------------|---------------------------------------------------------------------------------------------------|
| <i>abstract-SyntaxTree</i>           | The abstract syntax tree representing the logic property to be checked                            |
| <i>multiscale-Architecture-Graph</i> | The multiscale architecture graph encoding the hierarchical organization of scales and subsystems |

Implements [multiscale::verification::ModelCheckerFactory](#).

Definition at line 15 of file StatisticalModelCheckerFactory.cpp.

References typeIError, and typeIIError.

#### 6.194.4 Member Data Documentation

6.194.4.1 **double multiscale::verification::StatisticalModelCheckerFactory::typeI-  
Error [private]**

The probability of a type I error

Definition at line 16 of file StatisticalModelCheckerFactory.hpp.

Referenced by `createInstance()`.

6.194.4.2 double multiscale::verification::StatisticalModelCheckerFactory::typell-  
Error [private]

The probability of a type II error

Definition at line 17 of file StatisticalModelCheckerFactory.hpp.

Referenced by createInstance().

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

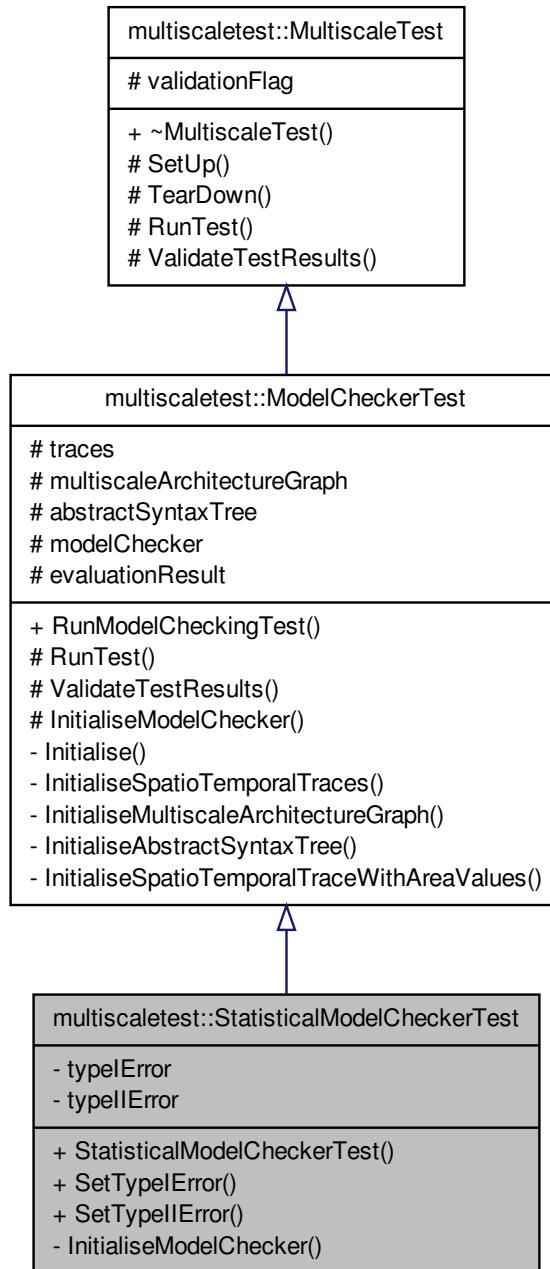
- StatisticalModelCheckerFactory.hpp
  
- StatisticalModelCheckerFactory.cpp

## 6.195 multiscaletest::StatisticalModelCheckerTest Class Reference

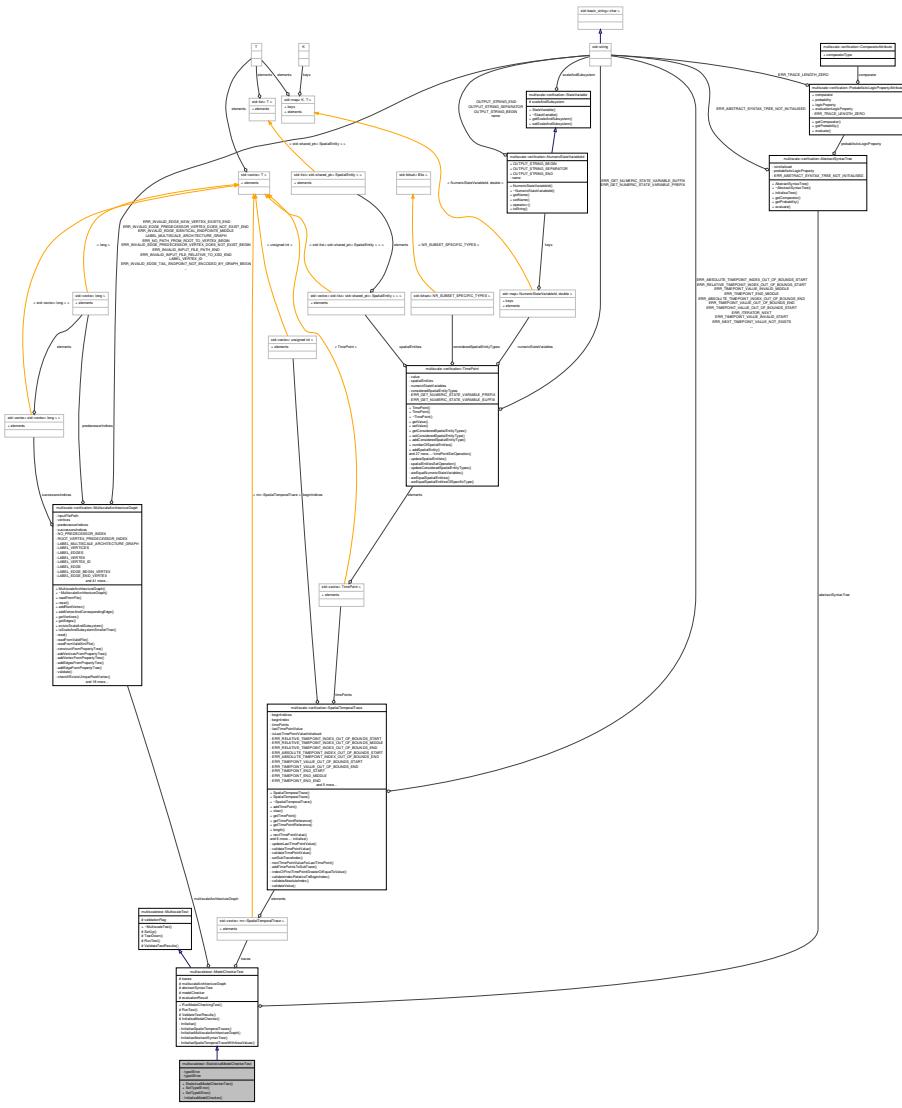
Class for testing the statistical model checker.

```
#include <StatisticalModelCheckerTest.hpp>
```

Inheritance diagram for multiscaletest::StatisticalModelCheckerTest:



## Collaboration diagram for multiscaletest::StatisticalModelCheckerTest:



## Public Member Functions

- `StatisticalModelCheckerTest ()`
  - `void SetTypeIError (double typeIError)`

*Set the value of the type I error*

- void SetTypeIIError (double typeIIError)

*Set the value of the type II error*

### Private Member Functions

- void [InitialiseModelChecker \(\) override](#)  
*Initialise the model checker.*

### Private Attributes

- double [typeIError](#)
- double [typeIIError](#)

#### 6.195.1 Detailed Description

Class for testing the statistical model checker.

Definition at line 15 of file StatisticalModelCheckerTest.hpp.

#### 6.195.2 Constructor & Destructor Documentation

##### 6.195.2.1 [multiscaletest::StatisticalModelCheckerTest::StatisticalModelCheckerTest\(\) \[inline\]](#)

Definition at line 24 of file StatisticalModelCheckerTest.hpp.

#### 6.195.3 Member Function Documentation

##### 6.195.3.1 [void multiscaletest::StatisticalModelCheckerTest::InitialiseModelChecker\(\) \[override, private, virtual\]](#)

Initialise the model checker.

Implements [multiscaletest::ModelCheckerTest](#).

Definition at line 55 of file StatisticalModelCheckerTest.hpp.

##### 6.195.3.2 [void multiscaletest::StatisticalModelCheckerTest::SetTypeIError\( double typeIError \)](#)

Set the value of the type I error.

#### Parameters

|                            |                                            |
|----------------------------|--------------------------------------------|
| <a href="#">typeIError</a> | The probability of type I errors occurring |
|----------------------------|--------------------------------------------|

Definition at line 47 of file StatisticalModelCheckerTest.hpp.

6.195.3.3 void multiscaletest::StatisticalModelCheckerTest::SetTypeIIError ( double typeIIError )

Set the value of the type II error.

Parameters

|                    |                                             |
|--------------------|---------------------------------------------|
| <i>typeIIError</i> | The probability of type II errors occurring |
|--------------------|---------------------------------------------|

Definition at line 51 of file StatisticalModelCheckerTest.hpp.

#### 6.195.4 Member Data Documentation

6.195.4.1 double multiscaletest::StatisticalModelCheckerTest::typeIError  
[private]

The probability of type I errors

Definition at line 19 of file StatisticalModelCheckerTest.hpp.

6.195.4.2 double multiscaletest::StatisticalModelCheckerTest::typeIIError  
[private]

The probability of type II errors

Definition at line 20 of file StatisticalModelCheckerTest.hpp.

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

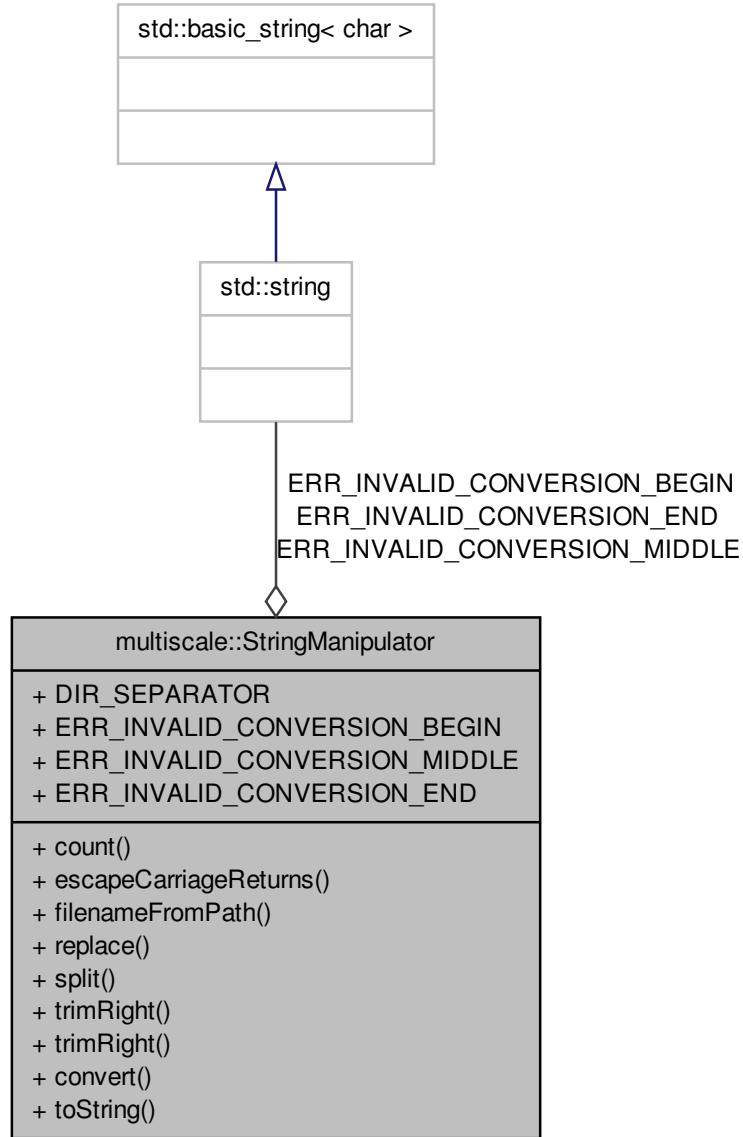
- StatisticalModelCheckerTest.hpp

## 6.196 multiscale::StringManipulator Class Reference

Class for manipulating strings.

```
#include <StringManipulator.hpp>
```

Collaboration diagram for multiscale::StringManipulator:



### Static Public Member Functions

- static unsigned long `count` (char searchChar, const std::string &inputString)

*Count how many times character occurs in the given string.*

- static std::string **escapeCarriageReturns** (const std::string &initialString)
 

*Escape carriage return characters in the provided string.*
- static std::string **filenameFromPath** (const std::string &filepath)
 

*Obtain the file name from the given file path.*
- static std::string **replace** (const std::string &initialString, const std::string &replaceWhat, const std::string &replaceTo)
 

*Replace a substd::string of the given std::string with another string.*
- static std::vector< std::string > **split** (const std::string &initialString, const std::string &delimiter)
 

*Split the given std::string into a std::vector of strings considering the given delimiter.*
- static std::string **trimRight** (std::string &inputString)
 

*Remove the trailing "new line" characters from the end of the string.*
- static std::string **trimRight** (const std::string &inputString)
 

*Remove the trailing "new line" characters from the end of the string.*
- template<typename T >
 static T **convert** (const std::string &inputString)
 

*Convert the string to the given type.*
- template<typename T >
 static std::string **toString** (T variable)
 

*Convert the variable to a string.*

## Static Public Attributes

- static const char **DIR\_SEPARATOR** = '/'
- static const std::string **ERR\_INVALID\_CONVERSION\_BEGIN** = "The provided string ("
- static const std::string **ERR\_INVALID\_CONVERSION\_MIDDLE** = ") could not be converted to a "
- static const std::string **ERR\_INVALID\_CONVERSION\_END** = ". Please change."

### 6.196.1 Detailed Description

Class for manipulating strings.

Definition at line 15 of file StringManipulator.hpp.

### 6.196.2 Member Function Documentation

#### 6.196.2.1 template<typename T > static T multiscale::StringManipulator::convert (const std::string & *inputString*) [inline, static]

Convert the string to the given type.

**Parameters**

|                    |                        |
|--------------------|------------------------|
| <i>inputString</i> | The given input string |
|--------------------|------------------------|

Definition at line 72 of file StringManipulator.hpp.

References ERR\_INVALID\_CONVERSION\_BEGIN, ERR\_INVALID\_CONVERSION-END, ERR\_INVALID\_CONVERSION\_MIDDLE, and escapeCarriageReturns().

**6.196.2.2 unsigned long StringManipulator::count ( char *searchChar*, const std::string & *inputString* ) [static]**

Count how many times character occurs in the given string.

**Parameters**

|                    |                                                              |
|--------------------|--------------------------------------------------------------|
| <i>searchChar</i>  | The search character of interest                             |
| <i>inputString</i> | The input string in which the character will be searched for |

Definition at line 9 of file StringManipulator.cpp.

**6.196.2.3 std::string StringManipulator::escapeCarriageReturns ( const std::string & *initialString* ) [static]**

Escape carriage return characters in the provided string.

**Parameters**

|                      |                             |
|----------------------|-----------------------------|
| <i>initialString</i> | The provided initial string |
|----------------------|-----------------------------|

Definition at line 22 of file StringManipulator.cpp.

References replace().

Referenced by convert().

**6.196.2.4 std::string StringManipulator::filenameFromPath ( const std::string & *filepath* ) [static]**

Obtain the file name from the given file path.

**Parameters**

|                 |           |
|-----------------|-----------|
| <i>filepath</i> | File path |
|-----------------|-----------|

Definition at line 26 of file StringManipulator.cpp.

References DIR\_SEPARATOR.

Referenced by multiscale::visualisation::PolarGnuplotScriptGenerator::generate-Header(), and multiscale::visualisation::RectangularGnuplotScriptGenerator::generate-

Header().

**6.196.2.5 std::string StringManipulator::replace ( const std::string & *initialString*, const std::string & *replaceWhat*, const std::string & *replaceTo* ) [static]**

Replace a substd::string of the given std::string with another string.

**Parameters**

|                      |                                                                 |
|----------------------|-----------------------------------------------------------------|
| <i>initialString</i> | Initial string                                                  |
| <i>replaceWhat</i>   | Substring which will be replaced                                |
| <i>replaceTo</i>     | String which will be inserted instead of the replaceWhat string |

Definition at line 36 of file StringManipulator.cpp.

Referenced by escapeCarriageReturns(), multiscale::visualisation::PolarGnuplotScriptGenerator::outputContent(), multiscale::visualisation::PolarGnuplotScriptGenerator::outputHeader(), and multiscale::visualisation::RectangularGnuplotScriptGenerator::outputHeader().

**6.196.2.6 std::vector< std::string > StringManipulator::split ( const std::string & *initialString*, const std::string & *delimiter* ) [static]**

Split the given std::string into a std::vector of strings considering the given delimiter.

**Parameters**

|                      |                |
|----------------------|----------------|
| <i>initialString</i> | Initial string |
| <i>delimiter</i>     | Delimiter      |

Definition at line 50 of file StringManipulator.cpp.

Referenced by multiscale::verification::TemporalDataReader::readInputFileContents(), multiscale::verification::TemporalDataReader::readInputFileHeader(), multiscale::visualisation::PolarCsvToInputFilesConverter::splitLineInConcentrations(), multiscale::visualisation::RectangularCsvToInputFilesConverter::splitLineInConcentrations(), multiscale::visualisation::RectangularEntityCsvToInputFilesConverter::splitLineInCoordinates(), multiscale::visualisation::RectangularEntityCsvToInputFilesConverter::validateInputLine(), multiscale::visualisation::PolarCsvToInputFilesConverter::validateInputLine(), and multiscale::visualisation::RectangularCsvToInputFilesConverter::validateInputLine().

**6.196.2.7 template<typename T > static std::string multiscale::StringManipulator::toString ( T *variable* ) [inline, static]**

Convert the variable to a string.

**Parameters**

|                 |          |
|-----------------|----------|
| <i>variable</i> | Variable |
|-----------------|----------|

Definition at line 97 of file StringManipulator.hpp.

Referenced by multiscale::verification::SpatialTemporalDataWriter::addValueAttributeToTimepointTree(), multiscale::Numeric::combinations(), multiscale::XmlValidator::XmlValidationErrorHandler::constructExceptionMessage(), multiscale::verification::ApproximateProbabilisticModelChecker::getDetailedResults(), multiscale::verification::ModelChecker::getDetailedResultsUsingPValues(), multiscale::verification::ApproximateBayesianModelChecker::getDetailedUpdatedResults(), multiscale::verification::BayesianModelChecker::getDetailedUpdatedResults(), multiscale::verification::StatisticalModelChecker::getDetailedUpdatedResults(), multiscale::verification::CommandLineModelChecking::initialiseApproximateBayesianModelChecker(), multiscale::verification::CommandLineModelChecking::initialiseApproximateProbabilisticModelChecker(), multiscale::verification::CommandLineModelChecking::initialiseBayesianModelChecker(), multiscale::verification::CommandLineModelChecking::initialiseStatisticalModelChecker(), multiscale::visualisation::RectangularEntityCsvToInputFilesConverter::initOutputFile(), multiscale::visualisation::PolarCsvToInputFilesConverter::initOutputFile(), multiscale::visualisation::RectangularCsvToInputFilesConverter::initOutputFile(), multiscale::verification::ModelCheckingOutputWriter::printEvaluationResultsSummary(), multiscale::verification::ModelCheckingOutputWriter::printInitialisationMessage(), multiscale::verification::ModelCheckingOutputWriter::printTimeoutMessage(), multiscale::ConsolePrinter::unixColourCodeToString(), multiscale::verification::SpatialTemporalTrace::validateAbsoluteIndex(), multiscale::verification::BayesianModelChecker::validateBayesFactorThreshold(), multiscale::verification::SpatialTemporalTrace::validateIndexRelativeToBeginIndex(), multiscale::verification::ApproximateProbabilisticModelChecker::validateInput(), multiscale::Numeric::validateLogBase(), multiscale::Numeric::validateLogNumber(), multiscale::BinomialDistribution::validateNrOfSuccesses(), multiscale::verification::MSTM::LSubfilesMerger::validateNumberOfTimepointsInResultingTrace(), multiscale::Distribution::validateProbability(), multiscale::BetaDistribution::validateShapeParameters(), multiscale::verification::BayesianModelChecker::validateShapeParameters(), multiscale::verification::ApproximateBayesianModelChecker::validateShapeParameters(), multiscale::verification::StatisticalModelChecker::validateTypesErrors(), and multiscale::verification::ApproximateBayesianModelChecker::validateVarianceThreshold().

#### 6.196.2.8 `std::string StringManipulator::trimRight ( std::string & inputString ) [static]`

Remove the trailing "new line" characters from the end of the string.

**Parameters**

|                    |                        |
|--------------------|------------------------|
| <i>inputString</i> | The given input string |
|--------------------|------------------------|

Definition at line 58 of file StringManipulator.cpp.

Referenced by multiscale::verification::ModelCheckingOutputWriter::printLogicPropertyWithTag(), multiscale::verification::ModelCheckingOutputWriter::printParsings-

LogicPropertyMessage(), trimRight(), and multiscale::verification::ParserGrammarExceptionHandler::trimRight().

#### 6.196.2.9 `std::string StringManipulator::trimRight ( const std::string & inputString ) [static]`

Remove the trailing "new line" characters from the end of the string.

##### Parameters

|                                 |                        |
|---------------------------------|------------------------|
| <code><i>inputString</i></code> | The given input string |
|---------------------------------|------------------------|

Definition at line 71 of file StringManipulator.cpp.

References trimRight().

### 6.196.3 Member Data Documentation

#### 6.196.3.1 `const char StringManipulator::DIR_SEPARATOR = '/' [static]`

Definition at line 108 of file StringManipulator.hpp.

Referenced by filenameFromPath().

#### 6.196.3.2 `const std::string StringManipulator::ERR_INVALID_CONVERSION_BEGIN = "The provided string (" [static]`

Definition at line 110 of file StringManipulator.hpp.

Referenced by convert().

#### 6.196.3.3 `const std::string StringManipulator::ERR_INVALID_CONVERSION_END = ". Please change." [static]`

Definition at line 112 of file StringManipulator.hpp.

Referenced by convert().

#### 6.196.3.4 `const std::string StringManipulator::ERR_INVALID_C-ONVERSION_MIDDLE = ") could not be converted to a " [static]`

Definition at line 111 of file StringManipulator.hpp.

Referenced by convert().

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

- StringManipulator.hpp

- StringManipulator.cpp

## 6.197 multiscale::verification::SubsetAttribute Class Reference

Class for representing a subset attribute.

```
#include <SubsetAttribute.hpp>
```

### Public Attributes

- [SubsetAttributeType subset](#)

#### 6.197.1 Detailed Description

Class for representing a subset attribute.

Definition at line 29 of file SubsetAttribute.hpp.

#### 6.197.2 Member Data Documentation

##### 6.197.2.1 [SubsetAttributeType multiscale::verification::SubsetAttribute::subset](#)

The considered subset

Definition at line 33 of file SubsetAttribute.hpp.

Referenced by [multiscale::verification::SubsetVisitor::operator\(\)\(\)](#), and [multiscale::verification::SpatialMeasureCollectionVisitor::operator\(\)\(\)](#).

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

- [SubsetAttribute.hpp](#)

## 6.198 multiscale::verification::SubsetOperationAttribute Class - Reference

Class for representing a subset operation attribute.

```
#include <SubsetOperationAttribute.hpp>
```

### Public Attributes

- [SubsetOperationType subsetOperationType](#)

## **6.199 multiscale::verification::SubsetOperationTypeParser Struct Reference 1101**

---

### **6.198.1 Detailed Description**

Class for representing a subset operation attribute.

Definition at line 29 of file SubsetOperationAttribute.hpp.

### **6.198.2 Member Data Documentation**

#### **6.198.2.1 SubsetOperationType multiscale::verification::SubsetOperationAttribute::subsetOperationType**

The subset operation type

Definition at line 33 of file SubsetOperationAttribute.hpp.

Referenced by multiscale::verification::SubsetVisitor::operator()().

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

- [SubsetOperationAttribute.hpp](#)

## **6.199 multiscale::verification::SubsetOperationTypeParser Struct - Reference**

Symbol table and parser for the subset operation type.

```
#include <SymbolTables.hpp>
```

### **Public Member Functions**

- [SubsetOperationTypeParser \(\)](#)

### **6.199.1 Detailed Description**

Symbol table and parser for the subset operation type.

Definition at line 169 of file SymbolTables.hpp.

### **6.199.2 Constructor & Destructor Documentation**

#### **6.199.2.1 multiscale::verification::SubsetOperationTypeParser::SubsetOperationTypeParser( ) [inline]**

Definition at line 172 of file SymbolTables.hpp.

References multiscale::verification::Intersection.

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

- SymbolTables.hpp

## 6.200 multiscale::verification::SubsetSpecificAttribute Class - Reference

Class for representing a subset specific attribute.

```
#include <SubsetSpecificAttribute.hpp>
```

### Public Attributes

- [SubsetSpecificType subsetSpecificType](#)

#### 6.200.1 Detailed Description

Class for representing a subset specific attribute.

Definition at line 67 of file SubsetSpecificAttribute.hpp.

#### 6.200.2 Member Data Documentation

##### 6.200.2.1 [SubsetSpecificType multiscale::verification::SubsetSpecificAttribute-::subsetSpecificType](#)

The specific subset type

Definition at line 71 of file SubsetSpecificAttribute.hpp.

Referenced by [multiscale::verification::SubsetVisitor::operator\(\)\(\)](#).

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

- [SubsetSpecificAttribute.hpp](#)

## 6.201 multiscale::verification::SubsetSpecificTypeParser Struct - Reference

Symbol table and parser for a specific subset type.

```
#include <SymbolTablesAutoGenerated.hpp>
```

### Public Member Functions

- [SubsetSpecificTypeParser \(\)](#)

## **6.202 multiscale::verification::SubsetSubsetOperationAttribute Class Reference**

### **6.201.1 Detailed Description**

Symbol table and parser for a specific subset type.

Definition at line 49 of file SymbolTablesAutoGenerated.hpp.

### **6.201.2 Constructor & Destructor Documentation**

#### **6.201.2.1 multiscale::verification::SubsetSpecificTypeParser::SubsetSpecificTypeParser( ) [inline]**

Definition at line 51 of file SymbolTablesAutoGenerated.hpp.

References multiscale::verification::Regions.

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

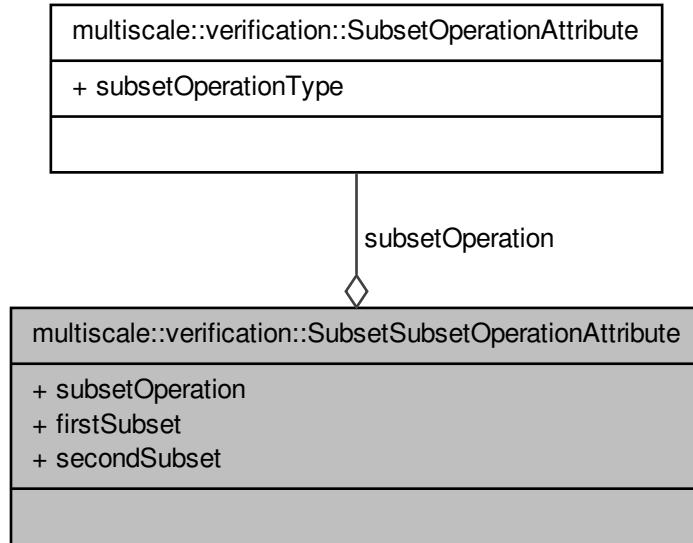
- SymbolTablesAutoGenerated.hpp

## **6.202 multiscale::verification::SubsetSubsetOperationAttribute - Class Reference**

Class for representing a subset subset operation attribute.

```
#include <SubsetSubsetOperationAttribute.hpp>
```

Collaboration diagram for multiscale::verification::SubsetSubsetOperationAttribute:



## Public Attributes

- [SubsetOperationAttribute](#) `subsetOperation`
- [SubsetAttributeType](#) `firstSubset`
- [SubsetAttributeType](#) `secondSubset`

### 6.202.1 Detailed Description

Class for representing a subset subset operation attribute.

Definition at line 15 of file `SubsetSubsetOperationAttribute.hpp`.

### 6.202.2 Member Data Documentation

#### 6.202.2.1 [SubsetAttributeType](#) `multiscale::verification::SubsetSubsetOperationAttribute::firstSubset`

The first considered subset

Definition at line 20 of file `SubsetSubsetOperationAttribute.hpp`.

Referenced by multiscale::verification::SubsetVisitor::operator()().

#### 6.202.2.2 SubsetAttributeType multiscale::verification::SubsetSubsetOperation-Attribute::secondSubset

The second considered subset

Definition at line 21 of file SubsetSubsetOperationAttribute.hpp.

Referenced by multiscale::verification::SubsetVisitor::operator()().

#### 6.202.2.3 SubsetOperationAttribute multiscale::verification::SubsetSubset-OperationAttribute::subsetOperation

The employed subset operation

Definition at line 19 of file SubsetSubsetOperationAttribute.hpp.

Referenced by multiscale::verification::SubsetVisitor::operator()().

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

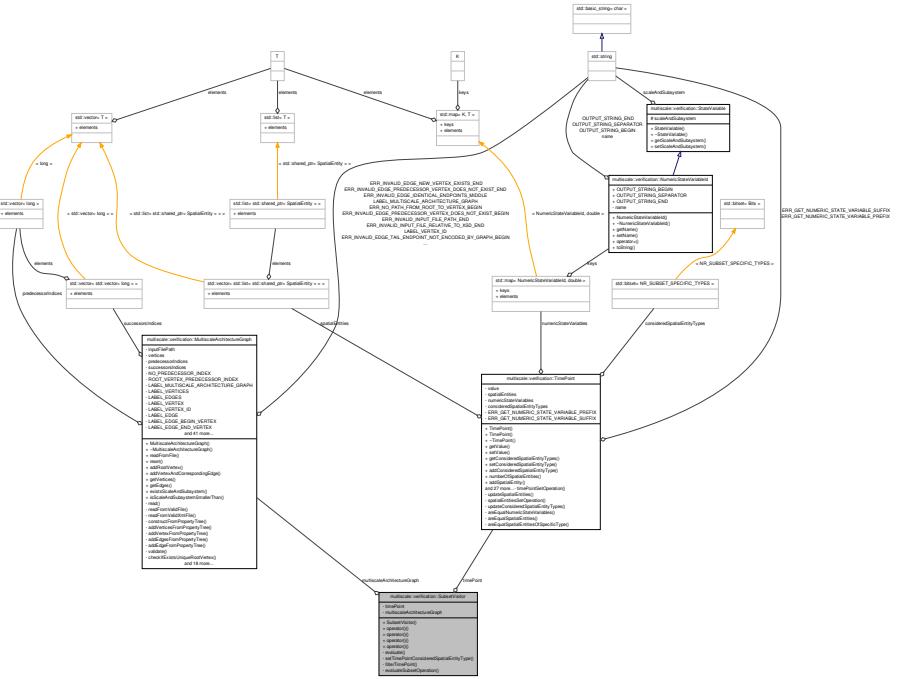
- SubsetSubsetOperationAttribute.hpp

## 6.203 multiscale::verification::SubsetVisitor Class Reference

Class used to evaluate subsets.

```
#include <SubsetVisitor.hpp>
```

## Collaboration diagram for multiscale::verification::SubsetVisitor::



## Public Member Functions

- `SubsetVisitor` (`TimePoint &timePoint, const MultiscaleArchitectureGraph &multiscaleArchitectureGraph`)
  - `TimePoint operator()` (`const SubsetAttribute &subset`) const
    - Overloading the "(" operator for the `SubsetAttribute` alternative.*
  - `TimePoint operator()` (`const SubsetSpecificAttribute &subset`) const
    - Overloading the "(" operator for the `SubsetSpecificAttribute` alternative.*
  - `TimePoint operator()` (`const FilterSubsetAttribute &subset`) const
    - Overloading the "(" operator for the `FilterSubsetAttribute` alternative.*
  - `TimePoint operator()` (`const SubsetSubsetOperationAttribute &subset`) const
    - Overloading the "(" operator for the `SubsetSubsetOperationAttribute` alternative.*

## Private Member Functions

- `TimePoint evaluate (const SubsetAttributeType &subset, TimePoint &timePoint)`  
const  
  
*Evaluate the subset considering the given timepoint.*
  - `void setTimePointConsideredSpatialEntityType (TimePoint &timePoint, const -`  
`SubsetSpecificType &subsetType) const`

*Set the considered spatial entity type for the given timepoint using the specific subset type.*

- `TimePoint filterTimePoint (TimePoint &timePoint, const ConstraintAttributeType &constraint) const`

*Filter the given timepoint considering the provided constraint.*

- `TimePoint evaluateSubsetOperation (const SubsetOperationType &subsetOperation, const TimePoint &firstSubsetTimePoint, const TimePoint &secondSubsetTimePoint) const`

*Evaluate subsetOperation against the given subsets timepoints.*

## Private Attributes

- `TimePoint & timePoint`
- `const MultiscaleArchitectureGraph & multiscaleArchitectureGraph`

### 6.203.1 Detailed Description

Class used to evaluate subsets.

Definition at line 14 of file SubsetVisitor.hpp.

### 6.203.2 Constructor & Destructor Documentation

- 6.203.2.1 `multiscale::verification::SubsetVisitor::SubsetVisitor ( TimePoint & timePoint, const MultiscaleArchitectureGraph & multiscaleArchitectureGraph ) [inline]`

Definition at line 25 of file SubsetVisitor.hpp.

Referenced by `evaluate()`.

### 6.203.3 Member Function Documentation

- 6.203.3.1 `TimePoint multiscale::verification::SubsetVisitor::evaluate ( const SubsetAttributeType & subset, TimePoint & timePoint ) const [inline, private]`

Evaluate the subset considering the given timepoint.

#### Parameters

|                        |                     |
|------------------------|---------------------|
| <code>subset</code>    | The subset          |
| <code>timePoint</code> | The given timepoint |

Definition at line 96 of file SubsetVisitor.hpp.

References `multiscaleArchitectureGraph`, and `SubsetVisitor()`.

Referenced by operator()().

**6.203.3.2 TimePoint multiscale::verification::SubsetVisitor::evaluateSubset-Operation ( const SubsetOperationType & subsetOperation, const TimePoint & firstSubsetTimePoint, const TimePoint & secondSubsetTimePoint ) const [inline, private]**

Evaluate subsetOperation against the given subsets timepoints.

**Parameters**

|                                |                                                  |
|--------------------------------|--------------------------------------------------|
| <i>subset-Operation</i>        | The considered subset operation                  |
| <i>firstSubset-TimePoint</i>   | The timepoint corresponding to the first subset  |
| <i>second-SubsetTime-Point</i> | The timepoint corresponding to the second subset |

Definition at line 132 of file SubsetVisitor.hpp.

References multiscale::ERR\_UNDEFINED\_ENUM\_VALUE, multiscale::verification::TimePoint::timePointDifference(), multiscale::verification::TimePoint::timePointIntersection(), and multiscale::verification::TimePoint::timePointUnion().

Referenced by operator()().

**6.203.3.3 multiscale::verification::TimePoint multiscale::verification::-SubsetVisitor::filterTimePoint ( TimePoint & timePoint, const ConstraintAttributeType & constraint ) const [inline, private]**

Filter the given timepoint considering the provided constraint.

**Parameters**

|                   |                         |
|-------------------|-------------------------|
| <i>timePoint</i>  | The given timepoint     |
| <i>constraint</i> | The provided constraint |

Definition at line 172 of file SubsetVisitor.hpp.

References multiscaleArchitectureGraph.

Referenced by operator()().

**6.203.3.4 TimePoint multiscale::verification::SubsetVisitor::operator() ( const SubsetAttribute & subset ) const [inline]**

Overloading the "(" operator for the [SubsetAttribute](#) alternative.

**Parameters**

|               |            |
|---------------|------------|
| <i>subset</i> | The subset |
|---------------|------------|

Definition at line 32 of file SubsetVisitor.hpp.

References evaluate(), multiscale::verification::SubsetAttribute::subset, and timePoint.

**6.203.3.5 TimePoint multiscale::verification::SubsetVisitor::operator() ( const  
SubsetSpecificAttribute & *subset* ) const [inline]**

Overloading the "(") operator for the [SubsetSpecificAttribute](#) alternative.

**Parameters**

|               |                     |
|---------------|---------------------|
| <i>subset</i> | The specific subset |
|---------------|---------------------|

Definition at line 40 of file SubsetVisitor.hpp.

References setTimePointConsideredSpatialEntityType(), multiscale::verification::SubsetSpecificAttribute::subsetSpecificType, and timePoint.

**6.203.3.6 TimePoint multiscale::verification::SubsetVisitor::operator() ( const  
FilterSubsetAttribute & *subset* ) const [inline]**

Overloading the "(") operator for the [FilterSubsetAttribute](#) alternative.

**Parameters**

|               |                   |
|---------------|-------------------|
| <i>subset</i> | The filter subset |
|---------------|-------------------|

Definition at line 55 of file SubsetVisitor.hpp.

References multiscale::verification::FilterSubsetAttribute::constraint, filterTimePoint(), setTimePointConsideredSpatialEntityType(), multiscale::verification::FilterSubsetAttribute::subsetSpecific, multiscale::verification::SubsetSpecificAttribute::subsetSpecificType, and timePoint.

**6.203.3.7 TimePoint multiscale::verification::SubsetVisitor::operator() ( const  
SubsetSubsetOperationAttribute & *subset* ) const [inline]**

Overloading the "(") operator for the [SubsetSubsetOperationAttribute](#) alternative.

**Parameters**

|               |                                       |
|---------------|---------------------------------------|
| <i>subset</i> | The subset subset operation attribute |
|---------------|---------------------------------------|

Definition at line 75 of file SubsetVisitor.hpp.

References evaluate(), evaluateSubsetOperation(), multiscale::verification::SubsetSubsetOperationAttribute::firstSubset, multiscale::verification::SubsetSubsetOperationAttribute::secondSubset, multiscale::verification::SubsetSubsetOperationAttribute::subsetOperation, multiscale::verification::SubsetOperationAttribute::subsetOperationType, and timePoint.

**6.203.3.8 void multiscale::verification::SubsetVisitor::setTimePointConsidered-SpatialEntityType ( TimePoint & *timePoint*, const SubsetSpecificType & *subsetType* ) const [inline, private]**

Set the considered spatial entity type for the given timepoint using the specific subset type.

#### Parameters

|                   |                          |
|-------------------|--------------------------|
| <i>timePoint</i>  | The given timepoint      |
| <i>subsetType</i> | The specific subset type |

Definition at line 113 of file SubsetVisitor.hpp.

References multiscale::verification::TimePoint::setConsideredSpatialEntityType().

Referenced by operator()().

### 6.203.4 Member Data Documentation

**6.203.4.1 const MultiscaleArchitectureGraph& multiscale-::verification::SubsetVisitor::multiscaleArchitectureGraph [private]**

The considered multiscale architecture graph

Definition at line 21 of file SubsetVisitor.hpp.

Referenced by evaluate(), and filterTimePoint().

**6.203.4.2 TimePoint& multiscale::verification::SubsetVisitor::timePoint [private]**

The initial timepoint

Definition at line 19 of file SubsetVisitor.hpp.

Referenced by operator()().

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

- SubsetVisitor.hpp

## 6.204 multiscale::SubtractionOperation Class Reference

Functor representing a subtraction operation.

```
#include <Numeric.hpp>
```

### Public Member Functions

- template<typename Operand >  
Operand **operator()** (Operand operand1, Operand operand2) const  
*Subtract the two operands.*

#### 6.204.1 Detailed Description

Functor representing a subtraction operation.

Definition at line 591 of file Numeric.hpp.

#### 6.204.2 Member Function Documentation

##### 6.204.2.1 template<typename Operand > Operand multiscale::SubtractionOperation::operator() ( Operand *operand1*, Operand *operand2* ) const [inline]

Subtract the two operands.

###### Parameters

|                 |                    |
|-----------------|--------------------|
| <i>operand1</i> | The first operand  |
| <i>operand2</i> | The second operand |

Definition at line 601 of file Numeric.hpp.

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

- Numeric.hpp

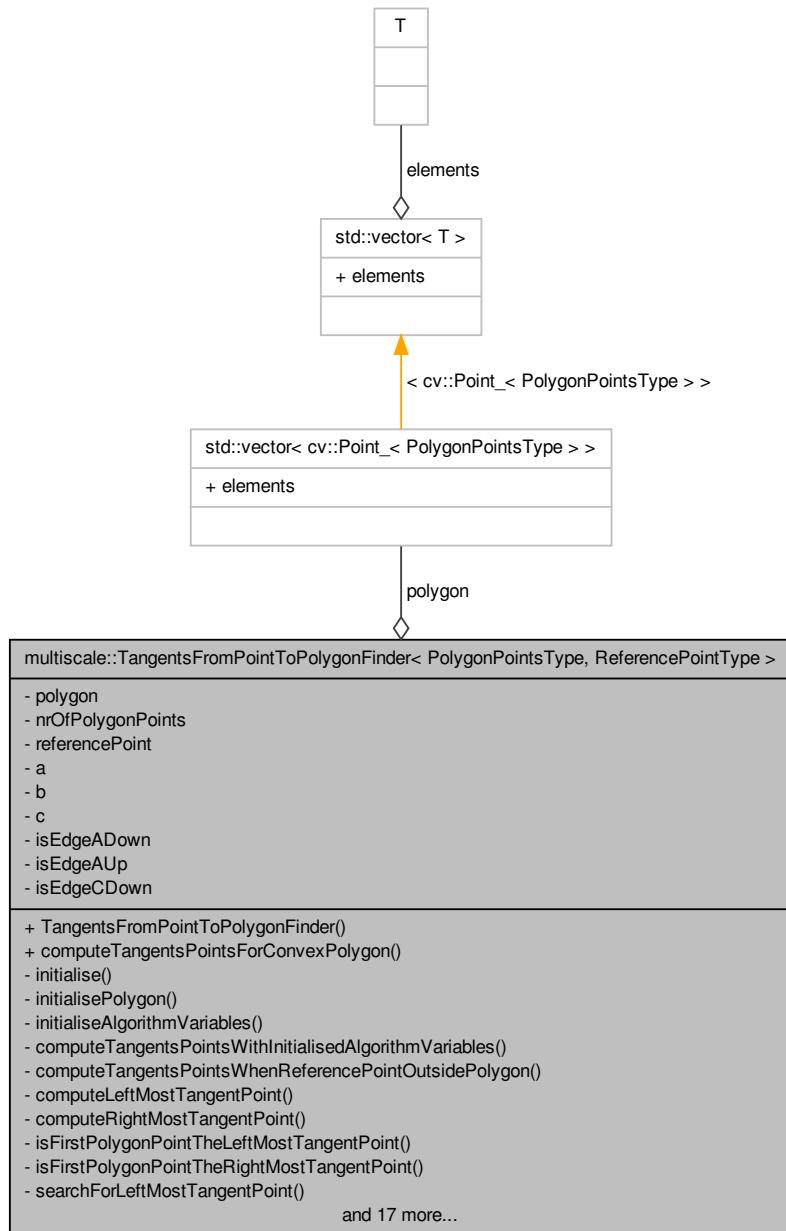
## 6.205 multiscale::TangentsFromPointToPolygonFinder< Polygon- PointsType, ReferencePointType > Class Template - Reference

Class for finding the tangents from a point to a polygon.

```
#include <TangentsFromPointToPolygonFinder.hpp>
```

Collaboration diagram for multiscale::TangentsFromPointToPolygonFinder< Polygon-

PointsType, ReferencePointType >:



### Public Member Functions

- `TangentsFromPointToPolygonFinder` (const std::vector< cv::Point\_< PolygonPointsType >> &`polygon`, const cv::Point\_< ReferencePointType > &`referencePoint`)
- void `computeTangentsPointsForConvexPolygon` (cv::Point\_< PolygonPointsType > &leftMostTangentPoint, cv::Point\_< PolygonPointsType > &rightMostTangentPoint)

*Compute the two polygon points through which the left, respectively right tangents pass.*

### Private Member Functions

- void `initialise` (const std::vector< cv::Point\_< PolygonPointsType >> &`polygon`)  
*Initialisation function.*
- void `initialisePolygon` (const std::vector< cv::Point\_< PolygonPointsType >> &`polygon`)  
*Initialisation of the polygon.*
- void `initialiseAlgorithmVariables` ()  
*Initialise the variables employed by the tangent points finding algorithm.*
- void `computeTangentsPointsWithInitialisedAlgorithmVariables` (cv::Point\_< - PolygonPointsType > &leftMostTangentPoint, cv::Point\_< PolygonPointsType > &rightMostTangentPoint)  
*Compute the two tangent points assuming the algorithm variables were initialised.*
- void `computeTangentsPointsWhenReferencePointOutsidePolygon` (cv::Point\_< - PolygonPointsType > &leftMostTangentPoint, cv::Point\_< PolygonPointsType > &rightMostTangentPoint)  
*Compute the two tangent points assuming the reference point lies outside the polygon.*
- cv::Point\_< PolygonPointsType > `computeLeftMostTangentPoint` ()  
*Compute the left-most polygon tangent point considering the given reference point.*
- cv::Point\_< PolygonPointsType > `computeRightMostTangentPoint` ()  
*Compute the right-most polygon tangent point considering the given reference point.*
- bool `isFirstPolygonPointTheLeftMostTangentPoint` ()  
*Check if the first polygon point is the left most tangent point.*
- bool `isFirstPolygonPointTheRightMostTangentPoint` ()  
*Check if the first polygon point is the right most tangent point.*
- cv::Point\_< PolygonPointsType > `searchForLeftMostTangentPoint` ()  
*Search for the left most tangent point.*
- cv::Point\_< PolygonPointsType > `searchForRightMostTangentPoint` ()  
*Search for the right most tangent point.*
- bool `isPointCLeftMostTangentPoint` ()  
*Check if the c-th polygon point is the left most tangent point.*
- bool `isPointCRightMostTangentPoint` ()  
*Check if the c-th polygon point is the right most tangent point.*

- void [updateLeftMostTangentPointSubChain \(\)](#)  
*Narrow the side chain in which the search for the left most tangent point continues.*
- void [updateRightMostTangentPointSubChain \(\)](#)  
*Narrow the side chain in which the search for the right most tangent point continues.*
- void [updateLeftMostTangentPointSubChainEdgeADown \(\)](#)  
*Narrow the side chain in which the search for the left most tangent point continues if edge A is down.*
- void [updateRightMostTangentPointSubChainEdgeADown \(\)](#)  
*Narrow the side chain in which the search for the right most tangent point continues if edge A is down.*
- void [updateLeftMostTangentPointSubChainEdgeAUp \(\)](#)  
*Narrow the side chain in which the search for the left most tangent point continues if edge A is up.*
- void [updateRightMostTangentPointSubChainEdgeAUp \(\)](#)  
*Narrow the side chain in which the search for the right most tangent point continues if edge A is up.*
- bool [isPointABelowPointC \(\)](#)  
*Check if point A is below point C.*
- bool [isPointAAbovePointC \(\)](#)  
*Check if point A is above point C.*
- void [updateCValue \(\)](#)  
*Set the value of index c as half of the sum of indices a and b.*
- void [updateEdgeCFlag \(\)](#)  
*Update the edge C related flag.*
- void [updateEdgeAUpFlag \(\)](#)  
*Update the edge A up flag.*
- void [updateEdgeADownFlag \(\)](#)  
*Update the edge A down flag.*
- void [setTangentPointsCoordinatesZero \(cv::Point\\_< PolygonPointsType > &leftMostTangentPoint, cv::Point\\_< PolygonPointsType > &rightMostTangentPoint\)](#)  
*Set the coordinates of the tangent points to zero.*
- std::size\_t [predecessor \(std::size\\_t currentIndex, std::size\\_t nrOfIndices\)](#)  
*Return the predecessor of the provided index.*
- std::size\_t [successor \(std::size\\_t currentIndex, std::size\\_t nrOfIndices\)](#)  
*Return the successor of the provided index.*

### Private Attributes

- std::vector< cv::Point\_< PolygonPointsType > > [polygon](#)
- std::size\_t [nrOfPolygonPoints](#)
- cv::Point\_< ReferencePointType > [referencePoint](#)
- std::size\_t [a](#)
- std::size\_t [b](#)
- std::size\_t [c](#)

- bool `isEdgeADown`
- bool `isEdgeAUp`
- bool `isEdgeCDown`

### 6.205.1 Detailed Description

```
template<typename PolygonPointsType, typename ReferencePointType> class multiscale::  
TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >
```

Class for finding the tangents from a point to a polygon.

Definition at line 15 of file TangentsFromPointToPolygonFinder.hpp.

### 6.205.2 Constructor & Destructor Documentation

```
6.205.2.1 template<typename PolygonPointsType, typename ReferencePointType>  
multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType,  
ReferencePointType >::TangentsFromPointToPolygonFinder ( const  
std::vector< cv::Point_< PolygonPointsType >> & polygon, const cv::Point_<  
ReferencePointType > & referencePoint ) [inline]
```

Definition at line 43 of file TangentsFromPointToPolygonFinder.hpp.

References `multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::initialise()`, and `multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::polygon`.

### 6.205.3 Member Function Documentation

```
6.205.3.1 template<typename PolygonPointsType, typename ReferencePointType> cv::Point_-  
<PolygonPointsType> multiscale::TangentsFromPointToPolygonFinder<  
PolygonPointsType, ReferencePointType >::computeLeftMostTangentPoint( )  
[inline, private]
```

Compute the left-most polygon tangent point considering the given reference point.

The tangent point is found using a binary search like procedure.

Definition at line 167 of file TangentsFromPointToPolygonFinder.hpp.

References `multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::initialiseAlgorithmVariables()`, `multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::isFirstPolygonPointTheLeftMostTangentPoint()`, `multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::polygon`, and `multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::searchForLeftMostTangentPoint()`.

Referenced by multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::computeTangentsPointsWhenReferencePointOutsidePolygon().

```
6.205.3.2 template<typename PolygonPointsType, typename ReferencePointType> cv::Point_<  
    <PolygonPointsType> multiscale::TangentsFromPointToPolygonFinder<  
        PolygonPointsType, ReferencePointType >::computeRightMostTangentPoint ( ) [inline, private]
```

Compute the right-most polygon tangent point considering the given reference point.

The tangent point is found using a binary search like procedure.

Definition at line 181 of file TangentsFromPointToPolygonFinder.hpp.

References multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::initialiseAlgorithmVariables(), multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::isFirstPolygonPointTheRightMostTangentPoint(), multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::polygon, and multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::searchForRightMostTangentPoint().

Referenced by multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::computeTangentsPointsWhenReferencePointOutsidePolygon().

```
6.205.3.3 template<typename PolygonPointsType, typename ReferencePointType> void  
multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType,  
ReferencePointType >::computeTangentsPointsForConvexPolygon  
( cv::Point_< PolygonPointsType > & leftMostTangentPoint, cv::Point_<  
    PolygonPointsType > & rightMostTangentPoint ) [inline]
```

Compute the two polygon points through which the left, respectively right tangents pass.

Each of the tangent lines passes through one or more polygon points and the reference point.

If the reference point lies inside the convex hull of the provided polygon then the left and right most tangent points are set to (0, 0).

If the number of points in the convex hull of the provided polygon is equal to: + 2, then the left and right most tangent points are the two polygon points; + 1, then the left and right most tangent points are identical with the polygon point; + 0, then the left and right most tangent points are set to (0, 0).

The employed algorithms are based on the information presented at: [http://geomalgorithms.com/a15-\\_tangents.html](http://geomalgorithms.com/a15-_tangents.html) (Accessed on: 16.12.-2014) However one problem with the algorithms presented at the above website is that they enter an infinite loop whenever a tangent to the polygon does not exist i.e. the tangent contains one of the sides of the polygon rather than a point.

**Parameters**

|                                |                                                          |
|--------------------------------|----------------------------------------------------------|
| <i>leftMost-Tangent-Point</i>  | The polygon point through which the left tangent passes  |
| <i>rightMost-Tangent-Point</i> | The polygon point through which the right tangent passes |

Definition at line 70 of file TangentsFromPointToPolygonFinder.hpp.

References `multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::computeTangentsPointsWithInitialisedAlgorithmVariables()`, and `multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::initialiseAlgorithmVariables()`.

Referenced by `multiscale::Geometry2D::tangentsFromPointToPolygon()`.

```
6.205.3.4 template<typename PolygonPointsType, typename ReferencePointType> void
    multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType,
    ReferencePointType >::computeTangentsPointsWhenReferencePoint-
    OutsidePolygon ( cv::Point_< PolygonPointsType > & leftMostTangentPoint,
    cv::Point_< PolygonPointsType > & rightMostTangentPoint ) [inline,
    private]
```

Compute the two tangent points assuming the reference point lies outside the polygon.

**Parameters**

|                                |                                                          |
|--------------------------------|----------------------------------------------------------|
| <i>leftMost-Tangent-Point</i>  | The polygon point through which the left tangent passes  |
| <i>rightMost-Tangent-Point</i> | The polygon point through which the right tangent passes |

Definition at line 145 of file TangentsFromPointToPolygonFinder.hpp.

References `multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::computeLeftMostTangentPoint()`, `multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::computeRightMostTangentPoint()`, `multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::nrOfPolygonPoints`, `multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::polygon`, and `multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::setTangentPointsCoordinatesZero()`.

Referenced by `multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::computeTangentsPointsWithInitialisedAlgorithmVariables()`.

---

```
6.205.3.5 template<typename PolygonPointsType, typename ReferencePointType> void
multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType,
ReferencePointType >::computeTangentsPointsWithInitialisedAlgorithm-
Variables ( cv::Point_< PolygonPointsType > & leftMostTangentPoint, cv::Point_<
PolygonPointsType > & rightMostTangentPoint ) [inline, private]
```

Compute the two tangent points assuming the algorithm variables were initialised.

**Parameters**

|                                                      |                                                          |
|------------------------------------------------------|----------------------------------------------------------|
| <i>leftMost-</i><br><i>Tangent-</i><br><i>Point</i>  | The polygon point through which the left tangent passes  |
| <i>rightMost-</i><br><i>Tangent-</i><br><i>Point</i> | The polygon point through which the right tangent passes |

Definition at line 125 of file TangentsFromPointToPolygonFinder.hpp.

References multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::computeTangentsPointsWhenReferencePointOutsidePolygon(), multiscale::Geometry2D::isPointInsidePolygon(), multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::polygon, multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::referencePoint, and multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::setTangentPointsCoordinatesZero().

Referenced by multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::computeTangentsPointsForConvexPolygon().

---

```
6.205.3.6 template<typename PolygonPointsType, typename ReferencePointType> void
multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType,
ReferencePointType >::initialise ( const std::vector< cv::Point_<
PolygonPointsType >> & polygon ) [inline, private]
```

Initialisation function.

**Parameters**

|                |                                                     |
|----------------|-----------------------------------------------------|
| <i>polygon</i> | The polygon for which the tangents will be computed |
|----------------|-----------------------------------------------------|

Definition at line 86 of file TangentsFromPointToPolygonFinder.hpp.

References multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::initialiseAlgorithmVariables(), multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::initialisePolygon(), and multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::polygon.

Referenced by multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::TangentsFromPointToPolygonFinder().

```
6.205.3.7 template<typename PolygonPointsType, typename ReferencePointType> void  
multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType,  
ReferencePointType >::initialiseAlgorithmVariables( ) [inline,  
private]
```

Initialise the variables employed by the tangent points finding algorithm.

Definition at line 109 of file TangentsFromPointToPolygonFinder.hpp.

References multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::a, multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::b, multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::c, multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::isEdgeADown, multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::isEdgeAUp, multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::isEdgeCDown, and multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::nrOfPolygonPoints.

Referenced by multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::computeLeftMostTangentPoint(), multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::computeRightMostTangentPoint(), multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::computeTangentsPointsForConvexPolygon(), and multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::initialise().

```
6.205.3.8 template<typename PolygonPointsType, typename ReferencePointType> void  
multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType,  
ReferencePointType >::initialisePolygon( const std::vector< cv::Point_<  
PolygonPointsType >> & polygon ) [inline, private]
```

Initialisation of the polygon.

If the given polygon is concave then its convex hull is considered.

**Parameters**

|                |                                                     |
|----------------|-----------------------------------------------------|
| <i>polygon</i> | The polygon for which the tangents will be computed |
|----------------|-----------------------------------------------------|

Definition at line 98 of file TangentsFromPointToPolygonFinder.hpp.

References multiscale::Geometry2D::computeConvexHull(), multiscale::Geometry2D::isConvexPolygon(), multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::nrOfPolygonPoints, and multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::polygon.

Referenced by multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::initialise().

---

```
6.205.3.9 template<typename PolygonPointsType, typename ReferencePointType> bool
multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType,
ReferencePointType >::isFirstPolygonPointTheLeftMostTangentPoint( )
[inline, private]
```

Check if the first polygon point is the left most tangent point.

Definition at line 193 of file TangentsFromPointToPolygonFinder.hpp.

References multiscale::Geometry2D::isToTheRightOfLine(), multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::nrOfPolygonPoints, multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::polygon, and multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::referencePoint.

Referenced by multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::computeLeftMostTangentPoint().

---

```
6.205.3.10 template<typename PolygonPointsType, typename ReferencePointType> bool
multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType,
ReferencePointType >::isFirstPolygonPointTheRightMostTangentPoint( )
[inline, private]
```

Check if the first polygon point is the right most tangent point.

Definition at line 209 of file TangentsFromPointToPolygonFinder.hpp.

References multiscale::Geometry2D::isToTheLeftOfLine(), multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::nrOfPolygonPoints, multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::polygon, and multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::referencePoint.

Referenced by multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::computeRightMostTangentPoint().

---

```
6.205.3.11 template<typename PolygonPointsType, typename ReferencePointType> bool
multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType,
ReferencePointType >::isPointAAbovePointC( ) [inline,
private]
```

Check if point A is above point C.

Definition at line 387 of file TangentsFromPointToPolygonFinder.hpp.

References multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::a, multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::c, multiscale::Geometry2D::isToTheLeftOfLine(), multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::polygon, and multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::referencePoint.

Referenced by multiscale::TangentsFromPointToPolygonFinder< PolygonPoints-

Type, ReferencePointType >::updateLeftMostTangentPointSubChainEdgeAUp(), and multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::updateRightMostTangentPointSubChainEdgeAUp().

**6.205.3.12 template<typename PolygonPointsType, typename ReferencePointType> bool  
multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType,  
ReferencePointType >::isPointABelowPointC( ) [inline, private]**

Check if point A is below point C.

Definition at line 375 of file TangentsFromPointToPolygonFinder.hpp.

References multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::a, multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::c, multiscale::Geometry2D::isToTheRightOfLine(), multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::polygon, and multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::referencePoint.

Referenced by multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::updateLeftMostTangentPointSubChainEdgeADown(), and multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::updateRightMostTangentPointSubChainEdgeADown().

**6.205.3.13 template<typename PolygonPointsType, typename ReferencePointType> bool  
multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType,  
ReferencePointType >::isPointCLeftMostTangentPoint( ) [inline,  
private]**

Check if the c-th polygon point is the left most tangent point.

Definition at line 261 of file TangentsFromPointToPolygonFinder.hpp.

References multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::c, multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::isEdgeCDown, multiscale::Geometry2D::isToTheRightOfLine(), multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::nrOfPolygonPoints, multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::polygon, multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::predecessor(), and multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::referencePoint.

Referenced by multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::searchForLeftMostTangentPoint().

---

```
6.205.3.14 template<typename PolygonPointsType, typename ReferencePointType> bool
multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType,
ReferencePointType >::isPointCRightMostTangentPoint( ) [inline,
private]
```

Check if the c-th polygon point is the right most tangent point.

Definition at line 275 of file TangentsFromPointToPolygonFinder.hpp.

References multiscale::Geometry2D::areCollinear(), multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::c, multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::isEdgeCDown, multiscale::Geometry2D::isToTheLeftOfLine(), multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::nrOfPolygonPoints, multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::polygon, multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::predecessor(), multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::referencePoint, and multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::successor().

Referenced by multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::searchForRightMostTangentPoint().

```
6.205.3.15 template<typename PolygonPointsType, typename ReferencePointType> std::size_t
multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType,
ReferencePointType >::predecessor( std::size_t currentIndex, std::size_t
nrOfIndices ) [inline, private]
```

Return the predecessor of the provided index.

#### Parameters

|                     |                             |
|---------------------|-----------------------------|
| <i>currentIndex</i> | The current index (0-based) |
| <i>nrOfIndices</i>  | The total number of indices |

Definition at line 450 of file TangentsFromPointToPolygonFinder.hpp.

Referenced by multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::isPointCLeftMostTangentPoint(), and multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::isPointCRightMostTangentPoint().

```
6.205.3.16 template<typename PolygonPointsType, typename ReferencePointType> cv::Point_<PolygonPointsType> multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::searchForLeftMostTangentPoint(
) [inline, private]
```

Search for the left most tangent point.

Definition at line 225 of file TangentsFromPointToPolygonFinder.hpp.

References multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::c, multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::isPointCLeftMostTangentPoint(), multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::polygon, multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::updateCValue(), multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::updateEdgeCFlag(), and multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::updateLeftMostTangentPointSubChain().

Referenced by multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::computeLeftMostTangentPoint().

**6.205.3.17 template<typename PolygonPointsType, typename ReferencePointType> cv::Point\_-<PolygonPointsType> multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::searchForRightMostTangentPoint( ) [inline, private]**

Search for the right most tangent point.

Definition at line 243 of file TangentsFromPointToPolygonFinder.hpp.

References multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::c, multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::isPointCRightMostTangentPoint(), multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::polygon, multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::updateCValue(), multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::updateEdgeCFlag(), and multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::updateRightMostTangentPointSubChain().

Referenced by multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::computeRightMostTangentPoint().

**6.205.3.18 template<typename PolygonPointsType, typename ReferencePointType> void multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::setTangentPointsCoordinatesZero( cv::Point\_-< PolygonPointsType > & leftMostTangentPoint, cv::Point\_-< PolygonPointsType > & rightMostTangentPoint ) [inline, private]**

Set the coordinates of the tangent points to zero.

Definition at line 435 of file TangentsFromPointToPolygonFinder.hpp.

Referenced by multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::computeTangentsPointsWhenReferencePointOutsidePolygon(), and multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::computeTangentsPointsWithInitialisedAlgorithmVariables().

---

6.205.3.19 template<typename PolygonPointsType, typename ReferencePointType> std::size\_t multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::successor ( std::size\_t currentIndex, std::size\_t nrOfIndices ) [inline, private]

Return the successor of the provided index.

#### Parameters

|                     |                             |
|---------------------|-----------------------------|
| <i>currentIndex</i> | The current index (0-based) |
| <i>nrOfIndices</i>  | The total number of indices |

Definition at line 464 of file TangentsFromPointToPolygonFinder.hpp.

Referenced by multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::isPointCRightMostTangentPoint(), multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::updateEdgeADownFlag(), multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::updateEdgeAUpFlag(), and multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::updateEdgeCFlag().

6.205.3.20 template<typename PolygonPointsType, typename ReferencePointType> void multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::updateCValue ( ) [inline, private]

Set the value of index c as half of the sum of indices a and b.

Definition at line 399 of file TangentsFromPointToPolygonFinder.hpp.

References multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::a, multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::b, and multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::c.

Referenced by multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::searchForLeftMostTangentPoint(), and multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::searchForRightMostTangentPoint().

6.205.3.21 template<typename PolygonPointsType, typename ReferencePointType> void multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::updateEdgeADownFlag ( ) [inline, private]

Update the edge A down flag.

Definition at line 425 of file TangentsFromPointToPolygonFinder.hpp.

References multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::a, multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::isEdgeADown, multiscale::Geometry2D::isToTheRightOfLine(), multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType,

ReferencePointType >::nrOfPolygonPoints, multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::polygon, multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::referencePoint, and multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::successor().

Referenced by multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::updateLeftMostTangentPointSubChain().

**6.205.3.22 template<typename PolygonPointsType, typename ReferencePointType> void  
multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType,  
ReferencePointType >::updateEdgeAUpFlag( ) [inline, private]**

Update the edge A up flag.

Definition at line 415 of file TangentsFromPointToPolygonFinder.hpp.

References multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::a, multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::isEdgeAUp, multiscale::Geometry2D::isToTheLeftOfLine(), multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::nrOfPolygonPoints, multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::polygon, multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::referencePoint, and multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::successor().

Referenced by multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::updateRightMostTangentPointSubChain().

**6.205.3.23 template<typename PolygonPointsType, typename ReferencePointType> void  
multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType,  
ReferencePointType >::updateEdgeCFlag( ) [inline, private]**

Update the edge C related flag.

Definition at line 405 of file TangentsFromPointToPolygonFinder.hpp.

References multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::c, multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::isEdgeCDown, multiscale::Geometry2D::isToTheRightOfLine(), multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::nrOfPolygonPoints, multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::polygon, multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::referencePoint, and multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::successor().

Referenced by multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::searchForLeftMostTangentPoint(), and multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::searchForRightMostTangentPoint().

---

```
6.205.3.24 template<typename PolygonPointsType, typename ReferencePointType> void
multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType,
ReferencePointType >::updateLeftMostTangentPointSubChain( )
[inline, private]
```

Narrow the side chain in which the search for the left most tangent point continues.

Definition at line 294 of file TangentsFromPointToPolygonFinder.hpp.

References multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::isEdgeADown, multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::updateEdgeADownFlag(), multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::updateLeftMostTangentPointSubChainEdgeADown(), and multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::updateLeftMostTangentPointSubChainEdgeAUp().

Referenced by multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::searchForLeftMostTangentPoint().

---

```
6.205.3.25 template<typename PolygonPointsType, typename ReferencePointType>
void multiscale::TangentsFromPointToPolygonFinder<
PolygonPointsType, ReferencePointType >::updateLeftMost-
TangentPointSubChainEdgeADown( ) [inline,
private]
```

Narrow the side chain in which the search for the left most tangent point continues if edge A is down.

Definition at line 319 of file TangentsFromPointToPolygonFinder.hpp.

References multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::a, multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::b, multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::c, multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::isEdgeCDown, and multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::isPointABelowPointC().

Referenced by multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::updateLeftMostTangentPointSubChain().

---

```
6.205.3.26 template<typename PolygonPointsType, typename ReferencePointType> void
multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType,
ReferencePointType >::updateLeftMostTangentPointSubChainEdgeAUp(
) [inline, private]
```

Narrow the side chain in which the search for the left most tangent point continues if edge A is up.

Definition at line 347 of file TangentsFromPointToPolygonFinder.hpp.

References multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::a, multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::b, multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::c, multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::isEdgeCDown, and multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::isPointAAbovePointC().

Referenced by multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::updateLeftMostTangentPointSubChain().

**6.205.3.27 template<typename PolygonPointsType, typename ReferencePointType> void  
multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType,  
ReferencePointType >::updateRightMostTangentPointSubChain( )  
[inline, private]**

Narrow the side chain in which the search for the right most tangent point continues.

Definition at line 307 of file TangentsFromPointToPolygonFinder.hpp.

References multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::isEdgeAUp, multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::updateEdgeAUpFlag(), multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::updateRightMostTangentPointSubChainEdgeADown(), and multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::updateRightMostTangentPointSubChainEdgeAUp().

Referenced by multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::searchForRightMostTangentPoint().

**6.205.3.28 template<typename PolygonPointsType, typename ReferencePointType>  
void multiscale::TangentsFromPointToPolygonFinder<  
PolygonPointsType, ReferencePointType >::updateRightMost-  
TangentPointSubChainEdgeADown( ) [inline,  
private]**

Narrow the side chain in which the search for the right most tangent point continues if edge A is down.

Definition at line 333 of file TangentsFromPointToPolygonFinder.hpp.

References multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::a, multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::b, multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::c, multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::isEdgeCDown, and multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::isPointABelowPointC().

Referenced by multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::updateRightMostTangentPointSubChain().

---

6.205.3.29 template<typename PolygonPointsType, typename ReferencePointType> void multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::updateRightMostTangentPointSubChainEdgeAUp( ) [inline, private]

Narrow the side chain in which the search for the right most tangent point continues if edge A is up.

Definition at line 361 of file TangentsFromPointToPolygonFinder.hpp.

References multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::a, multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::b, multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::c, multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::isEdgeCDown, and multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::isPointAAbovePointC().

Referenced by multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::updateRightMostTangentPointSubChain().

#### 6.205.4 Member Data Documentation

6.205.4.1 template<typename PolygonPointsType, typename ReferencePointType> std::size\_t multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::a [private]

Index a (0-based) used in the algorithms

Definition at line 29 of file TangentsFromPointToPolygonFinder.hpp.

Referenced by multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::initialiseAlgorithmVariables(), multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::isPointAAbovePointC(), multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::isPointABelowPointC(), multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::updateCValue(), multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::updateEdgeADownFlag(), multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::updateEdgeAUpFlag(), multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::updateLeftMostTangentPointSubChainEdgeADown(), multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::updateLeftMostTangentPointSubChainEdgeAUp(), multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::updateRightMostTangentPointSubChainEdgeADown(), and multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::updateRightMostTangentPointSubChainEdgeAUp().

**6.205.4.2 template<typename PolygonPointsType, typename ReferencePointType> std::size\_t  
multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType,  
ReferencePointType >::b [private]**

Index b (0-based) used in the algorithms

Definition at line 30 of file TangentsFromPointToPolygonFinder.hpp.

Referenced by multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::initialiseAlgorithmVariables(), multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::updateCValue(), multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::updateLeftMostTangentPointSubChainEdgeADown(), multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::updateLeftMostTangentPointSubChainEdgeAUp(), multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::updateRightMostTangentPointSubChainEdgeADown(), and multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::updateRightMostTangentPointSubChainEdgeAUp().

**6.205.4.3 template<typename PolygonPointsType, typename ReferencePointType> std::size\_t  
multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType,  
ReferencePointType >::c [private]**

Index c (0-based) used in the algorithms

Definition at line 31 of file TangentsFromPointToPolygonFinder.hpp.

Referenced by multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::initialiseAlgorithmVariables(), multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::isPointAAbovePointC(), multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::isPointABelowPointC(), multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::isPointCLeftMostTangentPoint(), multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::isPointCRightMostTangentPoint(), multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::searchForLeftMostTangentPoint(), multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::searchForRightMostTangentPoint(), multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::updateCValue(), multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::updateEdgeCFlag(), multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::updateLeftMostTangentPointSubChainEdgeADown(), multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::updateLeftMostTangentPointSubChainEdgeAUp(), multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::updateRightMostTangentPointSubChainEdgeADown(), and multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::updateRightMostTangentPointSubChainEdgeAUp().

---

**6.205.4.4 template<typename PolygonPointsType, typename ReferencePointType> bool multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::isEdgeADown [private]**

Flag indicating if the edge A is pointing down

Definition at line 33 of file TangentsFromPointToPolygonFinder.hpp.

Referenced by multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::initialiseAlgorithmVariables(), multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::updateEdgeADownFlag(), and multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::updateLeftMostTangentPointSubChain().

**6.205.4.5 template<typename PolygonPointsType, typename ReferencePointType> bool multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::isEdgeAUp [private]**

Flag indicating if the edge A is pointing up

Definition at line 35 of file TangentsFromPointToPolygonFinder.hpp.

Referenced by multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::initialiseAlgorithmVariables(), multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::updateEdgeAUpFlag(), and multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::updateRightMostTangentPointSubChain().

**6.205.4.6 template<typename PolygonPointsType, typename ReferencePointType> bool multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::isEdgeCDown [private]**

Flag indicating if the edge C is pointing down

Definition at line 37 of file TangentsFromPointToPolygonFinder.hpp.

Referenced by multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::initialiseAlgorithmVariables(), multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::isPointCLeftMostTangentPoint(), multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::isPointCRightMostTangentPoint(), multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::updateEdgeCFlag(), multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::updateLeftMostTangentPointSubChainEdgeADown(), multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::updateLeftMostTangentPointSubChainEdgeAUp(), multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::updateRightMostTangentPointSubChainEdgeADown(), and multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::updateRightMostTangentPointSubChainEdgeAUp().

**6.205.4.7 template<typename PolygonPointsType, typename ReferencePointType> std::size\_t  
multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType,  
ReferencePointType >::nrOfPolygonPoints [private]**

The number of points defining the polygon

Definition at line 23 of file TangentsFromPointToPolygonFinder.hpp.

Referenced by multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::computeTangentsPointsWhenReferencePointOutsidePolygon(), multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::initialiseAlgorithmVariables(), multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::initialisePolygon(), multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::isFirstPolygonPointTheLeftMostTangentPoint(), multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::isFirstPolygonPointTheRightMostTangentPoint(), multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::isPointCLeftMostTangentPoint(), multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::isPointCRightMostTangentPoint(), multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::updateEdgeADownFlag(), multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::updateEdgeAUpFlag(), and multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::updateEdgeCFlag().

**6.205.4.8 template<typename PolygonPointsType, typename ReferencePointType>  
std::vector<cv::Point<PolygonPointsType>> multiscale::TangentsFrom-  
PointToPolygonFinder< PolygonPointsType, ReferencePointType >::polygon  
[private]**

The polygon for which the tangents will be computed

Definition at line 20 of file TangentsFromPointToPolygonFinder.hpp.

Referenced by multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::computeLeftMostTangentPoint(), multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::computeRightMostTangentPoint(), multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::computeTangentsPointsWhenReferencePointOutsidePolygon(), multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::computeTangentsPointsWithInitialisedAlgorithmVariables(), multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::initialise(), multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::initialisePolygon(), multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::isFirstPolygonPointTheLeftMostTangentPoint(), multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::isFirstPolygonPointTheRightMostTangentPoint(), multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::isPointAAbovePointC(), multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::isPointABelowPointC(), multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::

Type >::isPointCLeftMostTangentPoint(), multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::isPointCRightMostTangentPoint(), multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::searchForLeftMostTangentPoint(), multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::searchForRightMostTangentPoint(), multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::TangentsFromPointToPolygonFinder(), multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::updateEdgeADownFlag(), multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::updateEdgeAUpFlag(), and multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::updateEdgeCFlag().

#### 6.205.4.9 template<typename PolygonPointsType, typename ReferencePointType> cv::Point-<ReferencePointType> multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::referencePoint [private]

The reference point through which the tangents pass

Definition at line 26 of file TangentsFromPointToPolygonFinder.hpp.

Referenced by multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::computeTangentsPointsWithInitialisedAlgorithmVariables(), multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::isFirstPolygonPointTheLeftMostTangentPoint(), multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::isFirstPolygonPointTheRightMostTangentPoint(), multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::isPointAAbovePointC(), multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::isPointABelowPointC(), multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::isPointCLeftMostTangentPoint(), multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::isPointCRightMostTangentPoint(), multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::updateEdgeADownFlag(), multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::updateEdgeAUpFlag(), and multiscale::TangentsFromPointToPolygonFinder< PolygonPointsType, ReferencePointType >::updateEdgeCFlag().

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

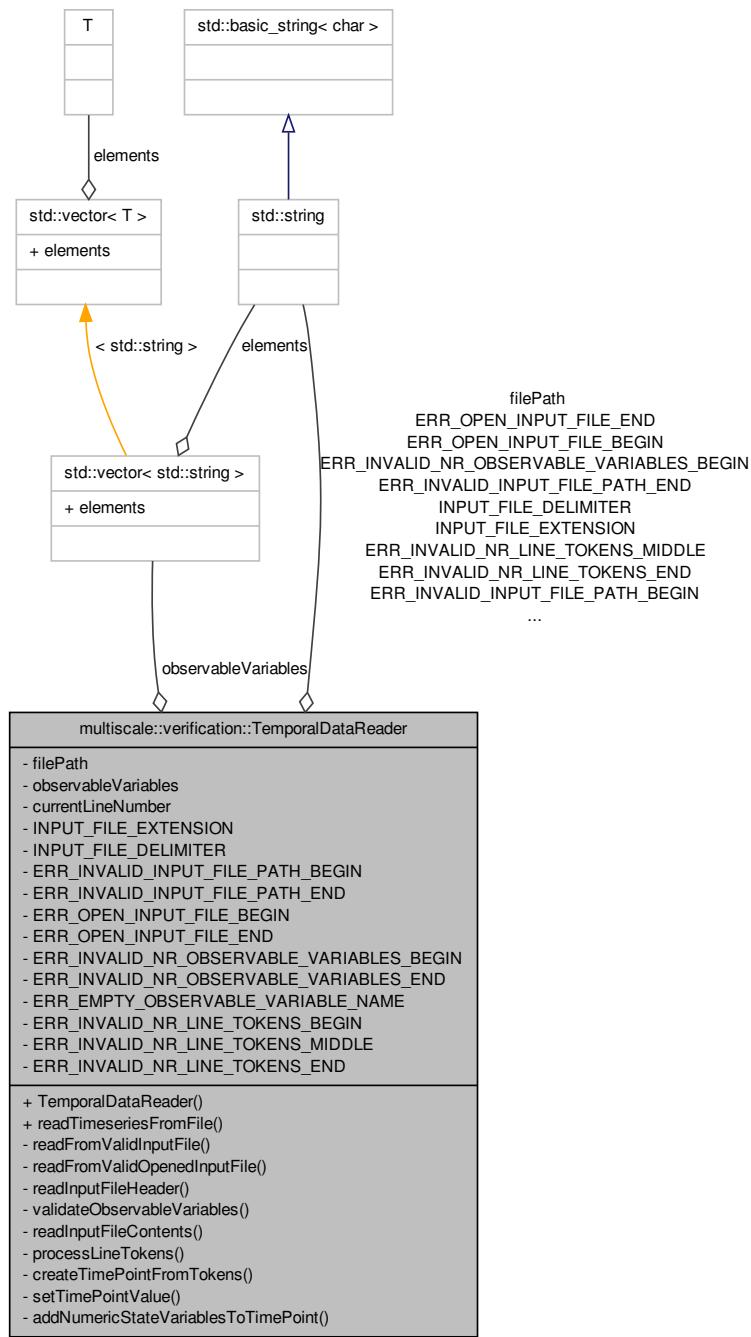
- TangentsFromPointToPolygonFinder.hpp

## 6.206 multiscale::verification::TemporalDataReader Class Reference

Class for reading (non-spatial) timeseries data from a .csv file.

```
#include <TemporalDataReader.hpp>
```

Collaboration diagram for multiscale::verification::TemporalDataReader:



## Public Member Functions

- `TemporalDataReader ()`
- `SpatialTemporalTrace readTimeseriesFromFile (const std::string &filePath)`

*Read the data from the input file and use it to construct a spatial temporal trace.*

## Private Member Functions

- `SpatialTemporalTrace readFromValidInputFile ()`  
*Read the data from the valid input file and construct a spatial temporal trace.*
- `void readFromValidOpenedInputFile (std::ifstream &fin, SpatialTemporalTrace &trace)`  
*Read the data from the valid opened input file and construct a spatial temporal trace.*
- `void readInputFileHeader (std::ifstream &fin, SpatialTemporalTrace &trace)`  
*Read the header row from the input file.*
- `void validateObservableVariables ()`  
*Validate the observable variables.*
- `void readInputFileContents (std::ifstream &fin, SpatialTemporalTrace &trace)`  
*Read the contents (excluding header row) from the input file.*
- `void processLineTokens (const std::vector< std::string > &lineTokens, SpatialTemporalTrace &trace)`  
*Check if the provided line tokens are valid and, if yes, add them to the trace.*
- `void createTimePointFromTokens (const std::vector< std::string > &lineTokens, SpatialTemporalTrace &trace)`  
*Create a new timepoint in the trace from the given tokens.*
- `void setTimePointValue (const std::vector< std::string > &lineTokens, TimePoint &timePoint)`  
*Set the value of the given timepoint considering the first token.*
- `void addNumericStateVariablesToTimePoint (const std::vector< std::string > &lineTokens, TimePoint &timePoint)`  
*Add the numeric state variable values to the timepoint.*

## Private Attributes

- `std::string filePath`
- `std::vector< std::string > observableVariables`
- `unsigned long currentLineNumber`

## Static Private Attributes

- `static const std::string INPUT_FILE_EXTENSION = ".csv"`
- `static const std::string INPUT_FILE_DELIMITER = ","`
- `static const std::string ERR_INVALID_INPUT_FILE_PATH_BEGIN = "The provided input file path ("`

- static const std::string `ERR_INVALID_INPUT_FILE_PATH_END` = ") does not point to a file with the required extension. Please change."
- static const std::string `ERR_OPEN_INPUT_FILE_BEGIN` = "The provided input file ("
- static const std::string `ERR_OPEN_INPUT_FILE_END` = ") could not be opened. Please make sure it is available and not currently used by another process."
- static const std::string `ERR_INVALID_NR_OBSERVABLE_VARIABLES_BEGIN` = "The number of observable variables ("
- static const std::string `ERR_INVALID_NR_OBSERVABLE_VARIABLES_END` = ") should be greater or equal to two. Please change."
- static const std::string `ERR_EMPTY_OBSERVABLE_VARIABLE_NAME` = "The name of one of the observable variables is empty when it should contain at least one character. Please change."
- static const std::string `ERR_INVALID_NR_LINE_TOKENS_BEGIN` = "The number of tokens on line "
- static const std::string `ERR_INVALID_NR_LINE_TOKENS_MIDDLE` = " of input file "
- static const std::string `ERR_INVALID_NR_LINE_TOKENS_END` = " is different from the number of observable variables in the header. Please change."

### 6.206.1 Detailed Description

Class for reading (non-spatial) timeseries data from a .csv file.

The format of the .csv input files is: Time, Observable 1, Observable 2, ..., Observable n T1, O11, O21, ..., On1 T2, O12, O22, ..., On2 ... Tm, O1m, O2m, ..., Onm where the first line contains the name of the observable variables (e.g. species) and the subsequent lines the values of these variables for a given timepoint (Ti, 1 <= i <= m)

Definition at line 26 of file TemporalDataReader.hpp.

### 6.206.2 Constructor & Destructor Documentation

#### 6.206.2.1 TemporalDataReader::TemporalDataReader( )

Definition at line 11 of file TemporalDataReader.cpp.

### 6.206.3 Member Function Documentation

#### 6.206.3.1 void TemporalDataReader::addNumericStateVariablesToTimePoint ( const std::vector< std::string > & *lineTokens*, TimePoint & *timePoint* ) [private]

Add the numeric state variable values to the timepoint.

**Parameters**

|                   |                        |
|-------------------|------------------------|
| <i>lineTokens</i> | The given line tokens  |
| <i>timePoint</i>  | The provided timepoint |

Definition at line 146 of file TemporalDataReader.cpp.

References multiscale::verification::TimePoint::addNumericStateVariable(), and observableVariables.

Referenced by createTimePointFromTokens().

**6.206.3.2 void TemporalDataReader::createTimePointFromTokens ( const std::vector< std::string > & *lineTokens*, SpatialTemporalTrace & *trace* ) [private]**

Create a new timepoint in the trace from the given tokens.

**Parameters**

|                   |                                                                       |
|-------------------|-----------------------------------------------------------------------|
| <i>lineTokens</i> | The given line tokens                                                 |
| <i>trace</i>      | The spatial temporal trace created using the data from the input file |

Definition at line 129 of file TemporalDataReader.cpp.

References addNumericStateVariablesToTimePoint(), multiscale::verification::SpatialTemporalTrace::addTimePoint(), and setTimePointValue().

Referenced by processLineTokens().

**6.206.3.3 void TemporalDataReader::processLineTokens ( const std::vector< std::string > & *lineTokens*, SpatialTemporalTrace & *trace* ) [private]**

Check if the provided line tokens are valid and, if yes, add them to the trace.

**Parameters**

|                   |                                                                       |
|-------------------|-----------------------------------------------------------------------|
| <i>lineTokens</i> | The given line tokens                                                 |
| <i>trace</i>      | The spatial temporal trace created using the data from the input file |

Definition at line 113 of file TemporalDataReader.cpp.

References createTimePointFromTokens(), currentLineNumber, ERR\_INVALID\_NR\_LINE\_TOKENS\_BEGIN, ERR\_INVALID\_NR\_LINE\_TOKENS\_END, ERR\_INVALID\_NR\_LINE\_TOKENS\_MIDDLE, filePath, and observableVariables.

Referenced by readInputFileContents().

**6.206.3.4 SpatialTemporalTrace TemporalDataReader::readFromValidInputFile ( ) [private]**

Read the data from the valid input file and construct a spatial temporal trace.

Definition at line 32 of file TemporalDataReader.cpp.

References filePath, and readFromValidOpenedInputFile().

Referenced by readTimeseriesFromFile().

**6.206.3.5 void TemporalDataReader::readFromValidOpenedInputFile ( std::ifstream & fin, SpatialTemporalTrace & trace ) [private]**

Read the data from the valid opened input file and construct a spatial temporal trace.

**Parameters**

|              |                                                                       |
|--------------|-----------------------------------------------------------------------|
| <i>fin</i>   | The input file stream opened for the given input file                 |
| <i>trace</i> | The spatial temporal trace created using the data from the input file |

Definition at line 44 of file TemporalDataReader.cpp.

References ERR\_OPEN\_INPUT\_FILE\_BEGIN, ERR\_OPEN\_INPUT\_FILE\_END, filePath, readInputFileContents(), and readInputFileHeader().

Referenced by readFromValidInputFile().

**6.206.3.6 void TemporalDataReader::readInputFileContents ( std::ifstream & fin, SpatialTemporalTrace & trace ) [private]**

Read the contents (excluding header row) from the input file.

**Parameters**

|              |                                                                       |
|--------------|-----------------------------------------------------------------------|
| <i>fin</i>   | The input file stream opened for the given input file                 |
| <i>trace</i> | The spatial temporal trace created using the data from the input file |

Definition at line 94 of file TemporalDataReader.cpp.

References currentLineNumber, INPUT\_FILE\_DELIMITER, processLineTokens(), and multiscale::StringManipulator::split().

Referenced by readFromValidOpenedInputFile().

**6.206.3.7 void TemporalDataReader::readInputFileHeader ( std::ifstream & fin, SpatialTemporalTrace & trace ) [private]**

Read the header row from the input file.

**Parameters**

|              |                                                                       |
|--------------|-----------------------------------------------------------------------|
| <i>fin</i>   | The input file stream opened for the given input file                 |
| <i>trace</i> | The spatial temporal trace created using the data from the input file |

Definition at line 58 of file TemporalDataReader.cpp.

References `currentLineNumber`, `INPUT_FILE_DELIMITER`, `observableVariables`, `multiscale::StringManipulator::split()`, and `validateObservableVariables()`.

Referenced by `readFromValidOpenedInputFile()`.

#### 6.206.3.8 SpatialTemporalTrace TemporalDataReader::readTimeseriesFromFile ( const std::string & *filePath* )

Read the data from the input file and use it to construct a spatial temporal trace.

**Parameters**

|                 |                     |
|-----------------|---------------------|
| <i>filePath</i> | The input file path |
|-----------------|---------------------|

Definition at line 13 of file TemporalDataReader.cpp.

References `currentLineNumber`, `ERR_INVALID_INPUT_FILE_PATH_BEGIN`, `ERR_INVALID_INPUT_FILE_PATH_END`, `filePath`, `INPUT_FILE_EXTENSION`, `multiscale::Filesystem::isValidFilePath()`, and `readFromValidInputFile()`.

#### 6.206.3.9 void TemporalDataReader::setTimePointValue ( const std::vector< std::string > & *lineTokens*, TimePoint & *timePoint* ) [private]

Set the value of the given timepoint considering the first token.

**Parameters**

|                   |                        |
|-------------------|------------------------|
| <i>lineTokens</i> | The given line tokens  |
| <i>timePoint</i>  | The provided timepoint |

Definition at line 139 of file TemporalDataReader.cpp.

References `multiscale::verification::TimePoint::setValue()`.

Referenced by `createTimePointFromTokens()`.

#### 6.206.3.10 void TemporalDataReader::validateObservableVariables ( ) [private]

Validate the observable variables.

The observable variables collection is valid if it contains two or more elements.

Definition at line 72 of file TemporalDataReader.cpp.

References `ERR_EMPTY_OBSERVABLE_VARIABLE_NAME`, `ERR_INVALID_NR_OBSERVABLE_VARIABLES_BEGIN`, `ERR_INVALID_NR_OBSERVABLE_VARIABLES_END`, and `observableVariables`.

Referenced by `readInputFileHeader()`.

#### 6.206.4 Member Data Documentation

**6.206.4.1 `unsigned long multiscale::verification::TemporalDataReader::currentLineNumber` [private]**

The current input file line number

Definition at line 34 of file `TemporalDataReader.hpp`.

Referenced by `processLineTokens()`, `readInputFileContents()`, `readInputFileHeader()`, and `readTimeseriesFromFile()`.

**6.206.4.2 `const std::string TemporalDataReader::ERR_EMPTY_OBSERVABLE_VARIABLE_NAME` = "The name of one of the observable variables is empty when it should contain at least one character. Please change." [static, private]**

Definition at line 124 of file `TemporalDataReader.hpp`.

Referenced by `validateObservableVariables()`.

**6.206.4.3 `const std::string TemporalDataReader::ERR_INVALID_INPUT_FILE_PATH_BEGIN` = "The provided input file path (" [static, private]**

Definition at line 115 of file `TemporalDataReader.hpp`.

Referenced by `readTimeseriesFromFile()`.

**6.206.4.4 `const std::string TemporalDataReader::ERR_INVALID_INPUT_FILE_PATH-END` = ") does not point to a file with the required extension. Please change." [static, private]**

Definition at line 116 of file `TemporalDataReader.hpp`.

Referenced by `readTimeseriesFromFile()`.

**6.206.4.5 `const std::string TemporalDataReader::ERR_INVALID_NR_LINE_TOKENS_BEGIN` = "The number of tokens on line " [static, private]**

Definition at line 126 of file `TemporalDataReader.hpp`.

Referenced by `processLineTokens()`.

6.206.4.6 `const std::string TemporalDataReader::ERR_INVALID_NR_LINE_TOKENS-END = " is different from the number of observable variables in the header. Please change." [static, private]`

Definition at line 128 of file TemporalDataReader.hpp.

Referenced by processLineTokens().

6.206.4.7 `const std::string TemporalDataReader::ERR_INVALID_NR_LINE_TOKENS_MIDDLE = " of input file " [static, private]`

Definition at line 127 of file TemporalDataReader.hpp.

Referenced by processLineTokens().

6.206.4.8 `const std::string TemporalDataReader::ERR_INVALID_NR_OBSERVABLE_VARIABLES_BEGIN = "The number of observable variables (" [static, private]`

Definition at line 121 of file TemporalDataReader.hpp.

Referenced by validateObservableVariables().

6.206.4.9 `const std::string TemporalDataReader::ERR_INVALID_NR_OBSERVABLE_VARIABLES_END = ") should be greater or equal to two. Please change." [static, private]`

Definition at line 122 of file TemporalDataReader.hpp.

Referenced by validateObservableVariables().

6.206.4.10 `const std::string TemporalDataReader::ERR_OPEN_INPUT_FILE_BEGIN = "The provided input file (" [static, private]`

Definition at line 118 of file TemporalDataReader.hpp.

Referenced by readFromValidOpenedInputFile().

6.206.4.11 `const std::string TemporalDataReader::ERR_OPEN_INPUT_FILE_END = " could not be opened. Please make sure it is available and not currently used by another process." [static, private]`

Definition at line 119 of file TemporalDataReader.hpp.

Referenced by readFromValidOpenedInputFile().

6.206.4.12 `std::string multiscale::verification::TemporalDataReader::filePath`  
[private]

The path to the input file

Definition at line 30 of file TemporalDataReader.hpp.

Referenced by processLineTokens(), readFromValidInputFile(), readFromValidOpenedInputFile(), and readTimeseriesFromFile().

6.206.4.13 `const std::string TemporalDataReader::INPUT_FILE_DELIMITER = ","`  
[static, private]

Definition at line 113 of file TemporalDataReader.hpp.

Referenced by readInputFileContents(), and readInputFileHeader().

6.206.4.14 `const std::string TemporalDataReader::INPUT_FILE_EXTENSION = ".csv"`  
[static, private]

Definition at line 111 of file TemporalDataReader.hpp.

Referenced by readTimeseriesFromFile().

6.206.4.15 `std::vector<std::string> multiscale::verification::TemporalDataReader::observableVariables` [private]

The names of the observable variables

Definition at line 32 of file TemporalDataReader.hpp.

Referenced by addNumericStateVariablesToTimePoint(), processLineTokens(), readInputFileHeader(), and validateObservableVariables().

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

- TemporalDataReader.hpp
- TemporalDataReader.cpp

## 6.207 multiscale::verification::TemporalNumericCollectionAttribute Class Reference

Class for representing a temporal numeric collection attribute.

```
#include <TemporalNumericCollectionAttribute.hpp>
```

### Public Attributes

- `TemporalNumericCollectionType temporalNumericCollection`

### 6.207.1 Detailed Description

Class for representing a temporal numeric collection attribute.

Definition at line 29 of file TemporalNumericCollectionAttribute.hpp.

### 6.207.2 Member Data Documentation

#### 6.207.2.1 **TemporalNumericCollectionType multiscale::verification::TemporalNumericCollectionAttribute::temporalNumericCollection**

The temporal numeric collection

Definition at line 33 of file TemporalNumericCollectionAttribute.hpp.

Referenced by multiscale::verification::NumericMeasureCollectionEvaluator::evaluateTemporalNumericCollection().

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

- TemporalNumericCollectionAttribute.hpp

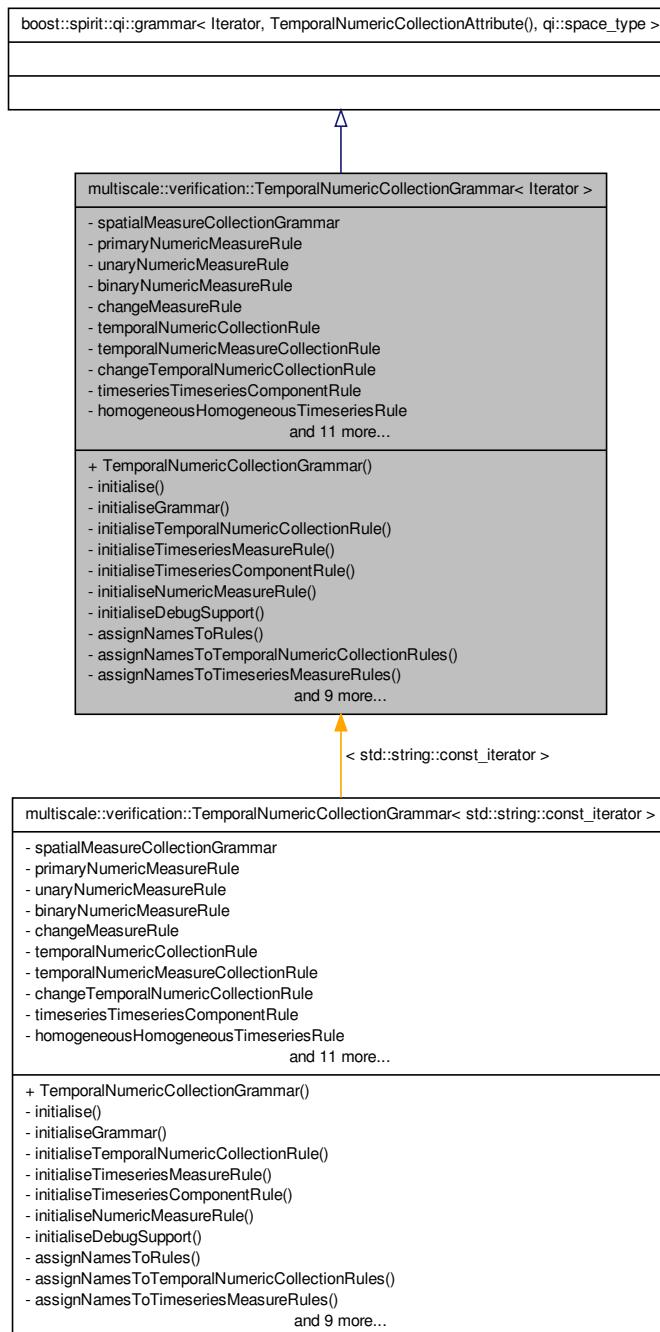
## 6.208 multiscale::verification::TemporalNumericCollectionGrammar< Iterator > Class Template Reference

The grammar for parsing temporal numeric collection statements.

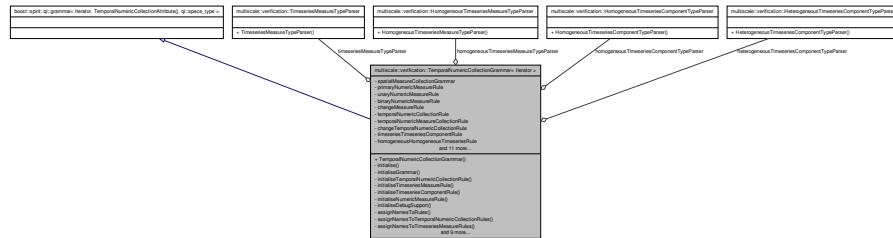
```
#include <TemporalNumericCollectionGrammar.hpp>
```

Inheritance diagram for multiscale::verification::TemporalNumericCollectionGrammar<

Iterator >:



Collaboration diagram for multiscale::verification::TemporalNumericCollection-Grammar< Iterator >:



# Public Member Functions

- TemporalNumericCollectionGrammar ()

## Private Member Functions

- void **initialise** ()  
*Initialisation function.*
  - void **initialiseGrammar** ()  
*Initialise the grammar.*
  - void **initialiseTemporalNumericCollectionRule** ()  
*Initialise the temporal numeric collection rule.*
  - void **initialiseTimeseriesMeasureRule** ()  
*Initialise the timeseries measure rule.*
  - void **initialiseTimeseriesComponentRule** ()  
*Initialise the timeseries component rule.*
  - void **initialiseNumericMeasureRule** ()  
*Initialise the numeric measure rule.*
  - void **initialiseDebugSupport** ()  
*Initialise debug support.*
  - void **assignNamesToRules** ()  
*Assign names to the rules.*
  - void **assignNamesToTemporalNumericCollectionRules** ()  
*Assign names to the temporal numeric collection rule.*
  - void **assignNamesToTimeseriesMeasureRules** ()  
*Assign names to the timeseries measure rule.*
  - void **assignNamesToTimeseriesComponentRules** ()  
*Assign names to the timeseries component rule.*
  - void **assignNamesToNumericMeasureRules** ()  
*Assign names to the numeric measure rules.*
  - void **initialiseRulesDebugging** ()

- Initialise the debugging of rules.*
- void [initialiseTemporalNumericCollectionRuleDebugging \(\)](#)  
*Initialise debugging for the temporal numeric collection rule.*
  - void [initialiseTimeseriesMeasureRuleDebugging \(\)](#)  
*Initialise debugging for the timeseries measure rule.*
  - void [initialiseTimeseriesComponentRuleDebugging \(\)](#)  
*Initialise debugging for the timeseries component rule.*
  - void [initialiseNumericMeasureRuleDebugging \(\)](#)  
*Initialise debugging for the numeric measure rule.*
  - void [initialiseErrorHandlingSupport \(\)](#)  
*Initialise the error handling routines.*
  - void [initialiseTemporalNumericCollectionErrorHandlingSupport \(\)](#)  
*Initialise the temporal numeric collection error handling support.*
  - void [initialiseNumericMeasureErrorHandlingSupport \(\)](#)  
*Initialise the numeric measure error handling support.*

## Private Attributes

- std::shared\_ptr < [SpatialMeasureCollectionGrammar](#) < Iterator > > [spatialMeasureCollectionGrammar](#)
- std::shared\_ptr < [PrimaryNumericMeasureGrammar](#) < Iterator > > [primaryNumericMeasureRule](#)
- [UnaryNumericMeasureGrammar](#) < Iterator > [unaryNumericMeasureRule](#)
- [BinaryNumericMeasureGrammar](#) < Iterator > [binaryNumericMeasureRule](#)
- [ChangeMeasureGrammar](#)< Iterator > [changeMeasureRule](#)
- qi::rule< Iterator, [TemporalNumericCollectionAttribute\(\)](#), qi::space\_type > [temporalNumericCollectionRule](#)
- qi::rule< Iterator, [TemporalNumericMeasureCollectionAttribute\(\)](#), qi::space\_type > [temporalNumericMeasureCollectionRule](#)
- qi::rule< Iterator, [ChangeTemporalNumericCollectionAttribute\(\)](#), qi::space\_type > [changeTemporalNumericCollectionRule](#)
- qi::rule< Iterator, [TimeseriesTimeseriesComponentAttribute\(\)](#), qi::space\_type > [timeseriesTimeseriesComponentRule](#)
- qi::rule< Iterator, [HomogeneousHomogeneousTimeseriesAttribute\(\)](#), qi::space\_type > [homogeneousHomogeneousTimeseriesRule](#)
- qi::rule< Iterator, [TimeseriesMeasureAttribute\(\)](#), qi::space\_type > [timeseriesMeasureRule](#)
- qi::rule< Iterator, [HomogeneousTimeseriesMeasureAttribute\(\)](#), qi::space\_type > [homogeneousTimeseriesMeasureRule](#)
- qi::rule< Iterator, [TimeseriesComponentAttribute\(\)](#), qi::space\_type > [timeseriesComponentRule](#)
- qi::rule< Iterator, [HeterogeneousTimeseriesComponentAttribute\(\)](#), qi::space\_type > [heterogeneousTimeseriesComponentRule](#)
- qi::rule< Iterator, [HomogeneousTimeseriesComponentAttribute\(\)](#), qi::space\_type > [homogeneousTimeseriesComponentRule](#)

- `qi::rule< Iterator, NumericMeasureAttribute(), qi::space_type > numeric-MeasureRule`
- `qi::rule< Iterator, UnaryNumericNumericAttribute(), qi::space_type > unary-NumericNumericRule`
- `qi::rule< Iterator, BinaryNumericNumericAttribute(), qi::space_type > binary-NumericNumericRule`
- `TimeseriesMeasureTypeParser timeseriesMeasureTypeParser`
- `HomogeneousTimeseriesMeasureTypeParser homogeneousTimeseries-MeasureTypeParser`
- `HeterogeneousTimeseriesComponentTypeParser heterogeneousTimeseries-ComponentTypeParser`
- `HomogeneousTimeseriesComponentTypeParser homogeneousTimeseries-ComponentTypeParser`

### 6.208.1 Detailed Description

```
template<typename Iterator> class multiscale::verification::TemporalNumericCollection-
Grammar< Iterator >
```

The grammar for parsing temporal numeric collection statements.

Definition at line 28 of file TemporalNumericCollectionGrammar.hpp.

### 6.208.2 Constructor & Destructor Documentation

```
6.208.2.1 template<typename Iterator > multiscale::verification::TemporalNumeric-
CollectionGrammar< Iterator >::TemporalNumericCollectionGrammar (
)
```

Definition at line 26 of file TemporalNumericCollectionGrammarDefinition.hpp.

References `multiscale::verification::TemporalNumericCollectionGrammar< Iterator >::initialise()`, `multiscale::verification::TemporalNumericCollectionGrammar< Iterator >::primaryNumericMeasureRule`, and `multiscale::verification::TemporalNumericCollectionGrammar< Iterator >::spatialMeasureCollectionGrammar`.

### 6.208.3 Member Function Documentation

```
6.208.3.1 template<typename Iterator > void multiscale::verification-
::TemporalNumericCollectionGrammar< Iterator
>::assignNamesToNumericMeasureRules( ) [private]
```

Assign names to the numeric measure rules.

Definition at line 201 of file TemporalNumericCollectionGrammarDefinition.hpp.

**6.208.3.2 template<typename Iterator > void multiscale::verification::Temporal-  
NumericCollectionGrammar< Iterator >::assignNamesToRules( )**  
[private]

Assign names to the rules.

Definition at line 167 of file TemporalNumericCollectionGrammarDefinition.hpp.

**6.208.3.3 template<typename Iterator > void multiscale::verification-  
::TemporalNumericCollectionGrammar< Iterator  
>::assignNamesToTemporalNumericCollectionRules( ) [private]**

Assign names to the temporal numeric collection rule.

Assign names to the temporal numeric collection rules.

Definition at line 176 of file TemporalNumericCollectionGrammarDefinition.hpp.

**6.208.3.4 template<typename Iterator > void multiscale::verification-  
::TemporalNumericCollectionGrammar< Iterator  
>::assignNamesToTimeseriesComponentRules( ) [private]**

Assign names to the timeseries component rule.

Assign names to the timeseries component rules.

Definition at line 193 of file TemporalNumericCollectionGrammarDefinition.hpp.

**6.208.3.5 template<typename Iterator > void multiscale::verification-  
::TemporalNumericCollectionGrammar< Iterator  
>::assignNamesToTimeseriesMeasureRules( ) [private]**

Assign names to the timeseries measure rule.

Assign names to the timeseries measure rules.

Definition at line 186 of file TemporalNumericCollectionGrammarDefinition.hpp.

**6.208.3.6 template<typename Iterator > void multiscale::verification::-  
TemporalNumericCollectionGrammar< Iterator >::initialise( )**  
[private]

Initialisation function.

Definition at line 42 of file TemporalNumericCollectionGrammarDefinition.hpp.

Referenced by multiscale::verification::TemporalNumericCollectionGrammar< Iterator  
>::TemporalNumericCollectionGrammar().

6.208.3.7 **template<typename Iterator > void multiscale::verification::TemporalNumericCollectionGrammar< Iterator >::initialiseDebugSupport( ) [private]**

Initialise debug support.

Definition at line 158 of file TemporalNumericCollectionGrammarDefinition.hpp.

6.208.3.8 **template<typename Iterator > void multiscale::verification::TemporalNumericCollectionGrammar< Iterator >::initialiseErrorHandlingSupport( ) [private]**

Initialise the error handling routines.

Definition at line 252 of file TemporalNumericCollectionGrammarDefinition.hpp.

6.208.3.9 **template<typename Iterator > void multiscale::verification::TemporalNumericCollectionGrammar< Iterator >::initialiseGrammar( ) [private]**

Initialise the grammar.

Definition at line 50 of file TemporalNumericCollectionGrammarDefinition.hpp.

6.208.3.10 **template<typename Iterator > void multiscale::verification::TemporalNumericCollectionGrammar< Iterator >::initialiseNumericMeasureErrorHandlingSupport( ) [private]**

Initialise the numeric measure error handling support.

Definition at line 280 of file TemporalNumericCollectionGrammarDefinition.hpp.

References multiscale::verification::handleUnexpectedTokenError.

6.208.3.11 **template<typename Iterator > void multiscale::verification::TemporalNumericCollectionGrammar< Iterator >::initialiseNumericMeasureRule( ) [private]**

Initialise the numeric measure rule.

Definition at line 131 of file TemporalNumericCollectionGrammarDefinition.hpp.

6.208.3.12 **template<typename Iterator > void multiscale::verification::TemporalNumericCollectionGrammar< Iterator >::initialiseNumericMeasureRuleDebugging( ) [private]**

Initialise debugging for the numeric measure rule.

Definition at line 244 of file TemporalNumericCollectionGrammarDefinition.hpp.

**6.208.3.13 template<typename Iterator > void multiscale::verification::Temporal-  
NumericCollectionGrammar< Iterator >::initialiseRulesDebugging( )**  
[private]

Initialise the debugging of rules.

Definition at line 210 of file TemporalNumericCollectionGrammarDefinition.hpp.

**6.208.3.14 template<typename Iterator > void multiscale::verification-  
::TemporalNumericCollectionGrammar< Iterator  
>::initialiseTemporalNumericCollectionErrorHandlingSupport( )**  
[private]

Initialise the temporal numeric collection error handling support.

Definition at line 259 of file TemporalNumericCollectionGrammarDefinition.hpp.

References multiscale::verification::handleUnexpectedTokenError.

**6.208.3.15 template<typename Iterator > void multiscale::verification-  
::TemporalNumericCollectionGrammar< Iterator  
>::initialiseTemporalNumericCollectionRule( ) [private]**

Initialise the temporal numeric collection rule.

Definition at line 59 of file TemporalNumericCollectionGrammarDefinition.hpp.

**6.208.3.16 template<typename Iterator > void multiscale::verification-  
::TemporalNumericCollectionGrammar< Iterator  
>::initialiseTemporalNumericCollectionRuleDebugging( )**  
[private]

Initialise debugging for the temporal numeric collection rule.

Definition at line 219 of file TemporalNumericCollectionGrammarDefinition.hpp.

**6.208.3.17 template<typename Iterator > void multiscale::verification-  
::TemporalNumericCollectionGrammar< Iterator  
>::initialiseTimeseriesComponentRule( ) [private]**

Initialise the timeseries component rule.

Definition at line 117 of file TemporalNumericCollectionGrammarDefinition.hpp.

**6.208.3.18 template<typename Iterator > void multiscale::verification-  
::TemporalNumericCollectionGrammar< Iterator  
>::initialiseTimeseriesComponentRuleDebugging( ) [private]**

Initialise debugging for the timeseries component rule.

Definition at line 236 of file TemporalNumericCollectionGrammarDefinition.hpp.

```
6.208.3.19 template<typename Iterator> void multiscale::verification-
::TemporalNumericCollectionGrammar< Iterator
>::initialiseTimeseriesMeasureRule( ) [private]
```

Initialise the timeseries measure rule.

Definition at line 107 of file TemporalNumericCollectionGrammarDefinition.hpp.

```
6.208.3.20 template<typename Iterator> void multiscale::verification-
::TemporalNumericCollectionGrammar< Iterator
>::initialiseTimeseriesMeasureRuleDebugging( ) [private]
```

Initialise debugging for the timeseries measure rule.

Initialise debugging for the timeseries measures rule.

Definition at line 229 of file TemporalNumericCollectionGrammarDefinition.hpp.

#### 6.208.4 Member Data Documentation

```
6.208.4.1 template<typename Iterator> BinaryNumericMeasureGrammar<Iterator>
multiscale::verification::TemporalNumericCollectionGrammar< Iterator
>::binaryNumericMeasureRule [private]
```

The grammar for parsing binary numeric measures

Definition at line 47 of file TemporalNumericCollectionGrammar.hpp.

```
6.208.4.2 template<typename Iterator> qi::rule<Iterator,
BinaryNumericNumericAttribute(), qi::space_type>
multiscale::verification::TemporalNumericCollectionGrammar< Iterator
>::binaryNumericNumericRule [private]
```

The rule for parsing a binary numeric numeric attribute

Definition at line 93 of file TemporalNumericCollectionGrammar.hpp.

```
6.208.4.3 template<typename Iterator> ChangeMeasureGrammar<Iterator>
multiscale::verification::TemporalNumericCollectionGrammar< Iterator
>::changeMeasureRule [private]
```

The grammar for parsing change measures

Definition at line 51 of file TemporalNumericCollectionGrammar.hpp.

```
6.208.4.4 template<typename Iterator> qi::rule<Iterator, Change-
TemporalNumericCollectionAttribute(), qi::space_type>
multiscale::verification::TemporalNumericCollectionGrammar< Iterator
>::changeTemporalNumericCollectionRule [private]
```

The rule for parsing a change temporal numeric collections attribute

Definition at line 62 of file TemporalNumericCollectionGrammar.hpp.

```
6.208.4.5 template<typename Iterator> qi::rule<Iterator, Heterogeneous-
TimeseriesComponentAttribute(), qi::space_type>
multiscale::verification::TemporalNumericCollectionGrammar< Iterator
>::heterogeneousTimeseriesComponentRule [private]
```

The rule for parsing a heterogeneous timeseries component

Definition at line 81 of file TemporalNumericCollectionGrammar.hpp.

```
6.208.4.6 template<typename Iterator> HeterogeneousTimeseriesComponentType-
Parser multiscale::verification::TemporalNumericCollectionGrammar<
Iterator >::heterogeneousTimeseriesComponentTypeParser
[private]
```

The heterogeneous timeseries component type parser

Definition at line 105 of file TemporalNumericCollectionGrammar.hpp.

```
6.208.4.7 template<typename Iterator> qi::rule<Iterator, Homogeneous-
HomogeneousTimeseriesAttribute(), qi::space_type>
multiscale::verification::TemporalNumericCollectionGrammar< Iterator
>::homogeneousHomogeneousTimeseriesRule [private]
```

The rule for parsing a homogeneous homogeneous timeseries measure attribute

Definition at line 68 of file TemporalNumericCollectionGrammar.hpp.

```
6.208.4.8 template<typename Iterator> qi::rule<Iterator, Homogeneous-
TimeseriesComponentAttribute(), qi::space_type>
multiscale::verification::TemporalNumericCollectionGrammar< Iterator
>::homogeneousTimeseriesComponentRule [private]
```

The rule for parsing a homogeneous timeseries component

Definition at line 84 of file TemporalNumericCollectionGrammar.hpp.

```
6.208.4.9 template<typename Iterator> HomogeneousTimeseriesComponentType-
Parser multiscale::verification::TemporalNumericCollectionGrammar<
Iterator >::homogeneousTimeseriesComponentTypeParser
[private]
```

The homogeneous timeseries component type parser

Definition at line 108 of file TemporalNumericCollectionGrammar.hpp.

```
6.208.4.10 template<typename Iterator> qi::rule<Iterator, Homogeneous-
TimeseriesMeasureAttribute(), qi::space_type>
multiscale::verification::TemporalNumericCollectionGrammar< Iterator
>::homogeneousTimeseriesMeasureRule [private]
```

The rule for parsing a homogeneous timeseries measure

Definition at line 74 of file TemporalNumericCollectionGrammar.hpp.

```
6.208.4.11 template<typename Iterator> HomogeneousTimeseriesMeasureType-
Parser multiscale::verification::TemporalNumericCollectionGrammar<
Iterator >::homogeneousTimeseriesMeasureTypeParser [private]
```

The homogeneous timeseries measure type parser

Definition at line 101 of file TemporalNumericCollectionGrammar.hpp.

```
6.208.4.12 template<typename Iterator> qi::rule<Iterator, NumericMeasureAttribute(),
qi::space_type> multiscale::verification::TemporalNumeric-
CollectionGrammar< Iterator >::numericMeasureRule
[private]
```

The rule for parsing a numeric measure

Definition at line 88 of file TemporalNumericCollectionGrammar.hpp.

```
6.208.4.13 template<typename Iterator> std::shared_ptr<-
PrimaryNumericMeasureGrammar<Iterator> >
multiscale::verification::TemporalNumericCollectionGrammar< Iterator
>::primaryNumericMeasureRule [private]
```

The grammar for parsing primary numeric measures

Definition at line 41 of file TemporalNumericCollectionGrammar.hpp.

Referenced by multiscale::verification::TemporalNumericCollectionGrammar< Iterator
>::TemporalNumericCollectionGrammar().

```
6.208.4.14 template<typename Iterator> std::shared_ptr<-
    SpatialMeasureCollectionGrammar<Iterator> >
    multiscale::verification::TemporalNumericCollectionGrammar< Iterator
    >::spatialMeasureCollectionGrammar [private]
```

The grammar for parsing spatial measure collection which will be passed by reference to the primary numeric measure grammar

Definition at line 36 of file TemporalNumericCollectionGrammar.hpp.

Referenced by multiscale::verification::TemporalNumericCollectionGrammar< Iterator >::TemporalNumericCollectionGrammar().

```
6.208.4.15 template<typename Iterator> qi::rule<Iterator, Temporal-
    NumericCollectionAttribute(), qi::space_type>
    multiscale::verification::TemporalNumericCollectionGrammar< Iterator
    >::temporalNumericCollectionRule [private]
```

The rule for parsing temporal numeric collections

Definition at line 56 of file TemporalNumericCollectionGrammar.hpp.

```
6.208.4.16 template<typename Iterator> qi::rule<Iterator, Temporal-
    NumericMeasureCollectionAttribute(), qi::space_type>
    multiscale::verification::TemporalNumericCollectionGrammar< Iterator
    >::temporalNumericMeasureCollectionRule [private]
```

The rule for parsing temporal numeric measure collections

Definition at line 59 of file TemporalNumericCollectionGrammar.hpp.

```
6.208.4.17 template<typename Iterator> qi::rule<Iterator,
    TimeseriesComponentAttribute(), qi::space_type>
    multiscale::verification::TemporalNumericCollectionGrammar< Iterator
    >::timeseriesComponentRule [private]
```

The rule for parsing a timeseries component

Definition at line 78 of file TemporalNumericCollectionGrammar.hpp.

```
6.208.4.18 template<typename Iterator> qi::rule<Iterator, TimeseriesMeasureAttribute(),
    qi::space_type> multiscale::verification::TemporalNumeric-
    CollectionGrammar< Iterator >::timeseriesMeasureRule
    [private]
```

The rule for parsing a timeseries measure

Definition at line 72 of file TemporalNumericCollectionGrammar.hpp.

---

```
6.208.4.19 template<typename Iterator> TimeseriesMeasureTypeParser
multiscale::verification::TemporalNumericCollectionGrammar< Iterator
>::timeseriesMeasureTypeParser [private]
```

The timeseries measure type parser

Definition at line 99 of file TemporalNumericCollectionGrammar.hpp.

```
6.208.4.20 template<typename Iterator> qi::rule<Iterator, Timeseries-
TimeseriesComponentAttribute(), qi::space_type>
multiscale::verification::TemporalNumericCollectionGrammar< Iterator
>::timeseriesTimeseriesComponentRule [private]
```

The rule for parsing a timeseries timeseries component attribute

Definition at line 65 of file TemporalNumericCollectionGrammar.hpp.

```
6.208.4.21 template<typename Iterator> UnaryNumericMeasureGrammar<Iterator>
multiscale::verification::TemporalNumericCollectionGrammar< Iterator
>::unaryNumericMeasureRule [private]
```

The grammar for parsing unary numeric measures

Definition at line 44 of file TemporalNumericCollectionGrammar.hpp.

```
6.208.4.22 template<typename Iterator> qi::rule<Iterator,
UnaryNumericNumericAttribute(), qi::space_type>
multiscale::verification::TemporalNumericCollectionGrammar< Iterator
>::unaryNumericNumericRule [private]
```

The rule for parsing a unary numeric numeric attribute

Definition at line 90 of file TemporalNumericCollectionGrammar.hpp.

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

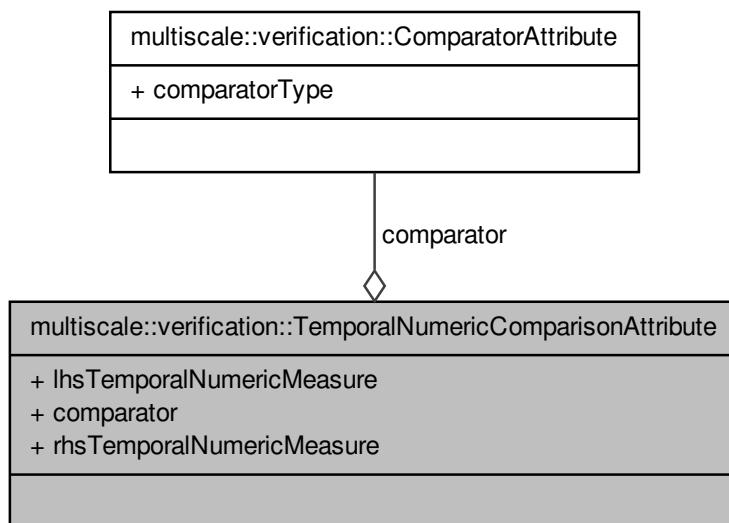
- TemporalNumericCollectionGrammar.hpp
- TemporalNumericCollectionGrammarDefinition.hpp

## 6.209 multiscale::verification::TemporalNumericComparison-Attribute Class Reference

Class for representing a temporal numeric comparison attribute.

```
#include <TemporalNumericComparisonAttribute.hpp>
```

Collaboration diagram for multiscale::verification::TemporalNumericComparisonAttribute:



## Public Attributes

- [TemporalNumericMeasureType lhsTemporalNumericMeasure](#)
- [ComparatorAttribute comparator](#)
- [TemporalNumericMeasureType rhsTemporalNumericMeasure](#)

### 6.209.1 Detailed Description

Class for representing a temporal numeric comparison attribute.

Definition at line 15 of file TemporalNumericComparisonAttribute.hpp.

### 6.209.2 Member Data Documentation

#### 6.209.2.1 ComparatorAttribute multiscale::verification::TemporalNumericComparisonAttribute::comparator

The comparator

Definition at line 22 of file TemporalNumericComparisonAttribute.hpp.

Referenced by multiscale::verification::LogicPropertyVisitor::evaluateTemporalNumericComparison().

#### 6.209.2.2 TemporalNumericMeasureType multiscale::verification::TemporalNumericComparisonAttribute::lhsTemporalNumericMeasure

The temporal numeric measure preceding the comparator

Definition at line 20 of file TemporalNumericComparisonAttribute.hpp.

Referenced by multiscale::verification::LogicPropertyVisitor::evaluateTemporalNumericComparison().

#### 6.209.2.3 TemporalNumericMeasureType multiscale::verification::TemporalNumericComparisonAttribute::rhsTemporalNumericMeasure

The temporal numeric measure succeeding the comparator

Definition at line 24 of file TemporalNumericComparisonAttribute.hpp.

Referenced by multiscale::verification::LogicPropertyVisitor::evaluateTemporalNumericComparison().

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

- TemporalNumericComparisonAttribute.hpp

### 6.210 multiscale::verification::TemporalNumericMeasureAttribute - Class Reference

Class for representing a temporal numeric measure attribute.

```
#include <TemporalNumericMeasureAttribute.hpp>
```

#### Public Attributes

- [TemporalNumericMeasureType temporalNumericMeasure](#)

#### 6.210.1 Detailed Description

Class for representing a temporal numeric measure attribute.

Definition at line 32 of file TemporalNumericMeasureAttribute.hpp.

#### 6.210.2 Member Data Documentation

**6.210.2.1 TemporalNumericMeasureType multiscale::verification::Temporal-  
NumericMeasureAttribute::temporalNumericMeasure**

The temporal numeric measure

Definition at line 36 of file TemporalNumericMeasureAttribute.hpp.

Referenced by multiscale::verification::TemporalNumericVisitor::operator()().

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

- TemporalNumericMeasureAttribute.hpp

**6.211 multiscale::verification::TemporalNumericMeasureCollection-  
Attribute Class Reference**

Class for representing temporal numeric measure collection attributes.

```
#include <TemporalNumericMeasureCollectionAttribute.hpp>
```

**Public Attributes**

- unsigned long [startTimepoint](#)
- unsigned long [endTimepoint](#)
- [NumericMeasureType numericMeasure](#)

**6.211.1 Detailed Description**

Class for representing temporal numeric measure collection attributes.

Definition at line 14 of file TemporalNumericMeasureCollectionAttribute.hpp.

**6.211.2 Member Data Documentation****6.211.2.1 unsigned long multiscale::verification::TemporalNumericMeasure-  
CollectionAttribute::endTimepoint**

The considered end timepoint

Definition at line 19 of file TemporalNumericMeasureCollectionAttribute.hpp.

Referenced by multiscale::verification::NumericMeasureCollectionVisitor::operator()().

**6.211.2.2 NumericMeasureType multiscale::verification::TemporalNumeric-  
MeasureCollectionAttribute::numericMeasure**

The numeric measure

Definition at line 21 of file TemporalNumericMeasureCollectionAttribute.hpp.

Referenced by multiscale::verification::NumericMeasureCollectionVisitor::operator()().

#### 6.211.2.3 **unsigned long multiscale::verification::TemporalNumericMeasureCollectionAttribute::startTimepoint**

The considered start timepoint

Definition at line 18 of file TemporalNumericMeasureCollectionAttribute.hpp.

Referenced by multiscale::verification::NumericMeasureCollectionVisitor::operator()().

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

- TemporalNumericMeasureCollectionAttribute.hpp

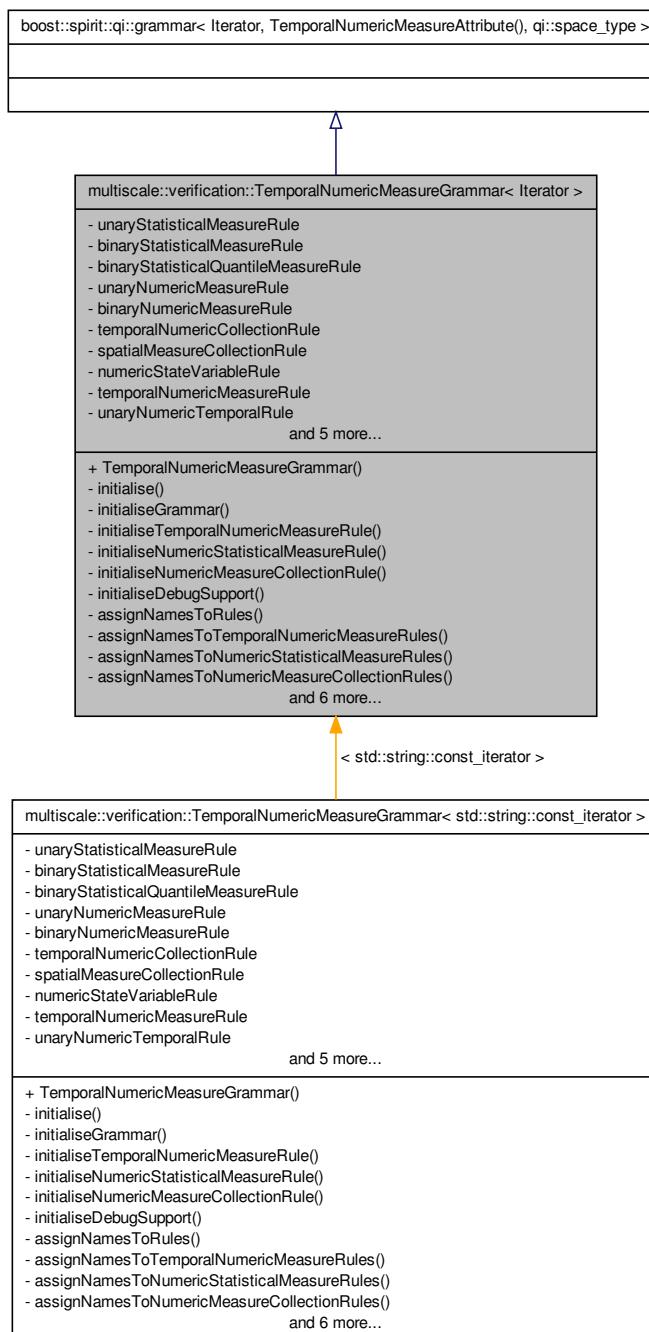
### 6.212 **multiscale::verification::TemporalNumericMeasureGrammar< Iterator >** Class Template Reference

The grammar for parsing temporal numeric measure statements.

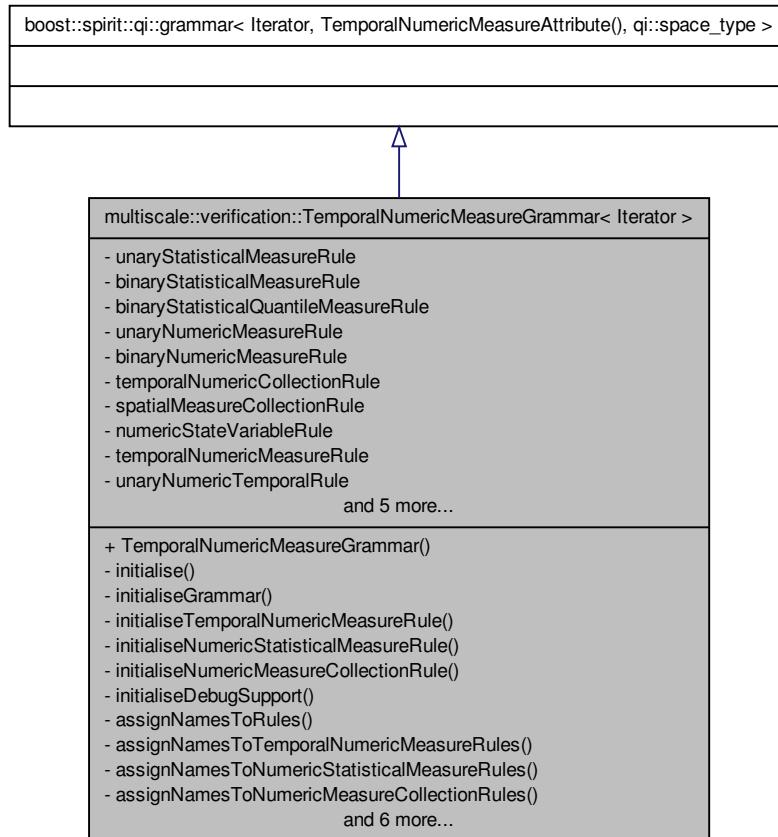
```
#include <TemporalNumericMeasureGrammar.hpp>
```

Inheritance diagram for multiscale::verification::TemporalNumericMeasureGrammar< -

Iterator >:



Collaboration diagram for multiscale::verification::TemporalNumericMeasureGrammar< Iterator >:



## Public Member Functions

- [TemporalNumericMeasureGrammar \(\)](#)

## Private Member Functions

- void [initialise \(\)](#)  
*Initialisation function.*
- void [initialiseGrammar \(\)](#)  
*Initialise the grammar.*
- void [initialiseTemporalNumericMeasureRule \(\)](#)

- *Initialise the temporal numeric measure rule.*  
• void [initialiseNumericStatisticalMeasureRule \(\)](#)  
*Initialise the numeric statistical measure rules.*
- void [initialiseNumericMeasureCollectionRule \(\)](#)  
*Initialise the numeric measure collection rule.*
- void [initialiseDebugSupport \(\)](#)  
*Initialise debug support.*
- void [assignNamesToRules \(\)](#)  
*Assign names to the rules.*
- void [assignNamesToTemporalNumericMeasureRules \(\)](#)  
*Assign names to the temporal numeric measure rules.*
- void [assignNamesToNumericStatisticalMeasureRules \(\)](#)  
*Assign names to the numeric statistical measure rules.*
- void [assignNamesToNumericMeasureCollectionRules \(\)](#)  
*Assign names to the numeric measure collection rules.*
- void [initialiseRulesDebugging \(\)](#)  
*Initialise the debugging of rules.*
- void [initialiseTemporalNumericMeasureRuleDebugging \(\)](#)  
*Initialise debugging for the temporal numeric measure rule.*
- void [initialiseNumericStatisticalMeasureRuleDebugging \(\)](#)  
*Initialise debugging for the numeric statistical measure rule.*
- void [initialiseNumericMeasureCollectionRuleDebugging \(\)](#)  
*Initialise debugging for the numeric measure collection rule.*
- void [initialiseErrorHandlingSupport \(\)](#)  
*Initialise the error handling routines.*
- void [initialiseTemporalNumericMeasureErrorHandlingSupport \(\)](#)  
*Initialise the temporal numeric measure error handling support.*
- void [initialiseNumericStatisticalMeasureErrorHandlingSupport \(\)](#)  
*Initialise the numeric statistical measure error handling support.*

## Private Attributes

- UnaryStatisticalMeasureGrammar < Iterator > [unaryStatisticalMeasureRule](#)
- BinaryStatisticalMeasureGrammar < Iterator > [binaryStatisticalMeasureRule](#)
- BinaryStatisticalQuantileMeasureGrammar < Iterator > [binaryStatisticalQuantileMeasureRule](#)
- UnaryNumericMeasureGrammar < Iterator > [unaryNumericMeasureRule](#)
- BinaryNumericMeasureGrammar < Iterator > [binaryNumericMeasureRule](#)
- TemporalNumericCollectionGrammar < Iterator > [temporalNumericCollectionRule](#)
- SpatialMeasureCollectionGrammar < Iterator > [spatialMeasureCollectionRule](#)
- NumericStateVariableGrammar < Iterator > [numericStateVariableRule](#)
- qi::rule< Iterator, TemporalNumericMeasureAttribute(), qi::space\_type > [temporalNumericMeasureRule](#)

- `qi::rule< Iterator, UnaryNumericTemporalAttribute(), qi::space_type > unary-NumericTemporalRule`
- `qi::rule< Iterator, BinaryNumericTemporalAttribute(), qi::space_type > binary-NumericTemporalRule`
- `qi::rule< Iterator, NumericStatisticalMeasureAttribute(), qi::space_type > numericStatisticalMeasureRule`
- `qi::rule< Iterator, UnaryStatisticalNumericAttribute(), qi::space_type > unary-StatisticalNumericRule`
- `qi::rule< Iterator, BinaryStatisticalNumericAttribute(), qi::space_type > binary-StatisticalNumericRule`
- `qi::rule< Iterator, BinaryStatisticalQuantileNumericAttribute(), qi::space_type > binaryStatisticalQuantileNumericRule`
- `qi::rule< Iterator, NumericMeasureCollectionAttribute(), qi::space_type > numericMeasureCollectionRule`

### 6.212.1 Detailed Description

```
template<typename Iterator>class multiscale::verification::TemporalNumericMeasureGrammar<Iterator>
```

The grammar for parsing temporal numeric measure statements.

Definition at line 32 of file TemporalNumericMeasureGrammar.hpp.

### 6.212.2 Constructor & Destructor Documentation

```
6.212.2.1 template<typename Iterator> multiscale::verification::TemporalNumericMeasureGrammar<Iterator>::TemporalNumericMeasureGrammar( )
```

Definition at line 24 of file TemporalNumericMeasureGrammarDefinition.hpp.

References `multiscale::verification::TemporalNumericMeasureGrammar< Iterator >::initialise()`.

### 6.212.3 Member Function Documentation

```
6.212.3.1 template<typename Iterator> void multiscale::verification::TemporalNumericMeasureGrammar< Iterator >::assignNamesToNumericMeasureCollectionRules( ) [private]
```

Assign names to the numeric measure collection rules.

Definition at line 155 of file TemporalNumericMeasureGrammarDefinition.hpp.

**6.212.3.2 template<typename Iterator > void multiscale::verification-  
::TemporalNumericMeasureGrammar< Iterator  
>::assignNamesToNumericStatisticalMeasureRules( ) [private]**

Assign names to the numeric statistical measure rules.

Definition at line 146 of file TemporalNumericMeasureGrammarDefinition.hpp.

**6.212.3.3 template<typename Iterator > void multiscale::verification::Temporal-  
NumericMeasureGrammar< Iterator >::assignNamesToRules( )  
[private]**

Assign names to the rules.

Definition at line 130 of file TemporalNumericMeasureGrammarDefinition.hpp.

**6.212.3.4 template<typename Iterator > void multiscale::verification-  
::TemporalNumericMeasureGrammar< Iterator  
>::assignNamesToTemporalNumericMeasureRules( ) [private]**

Assign names to the temporal numeric measure rules.

Definition at line 138 of file TemporalNumericMeasureGrammarDefinition.hpp.

**6.212.3.5 template<typename Iterator > void multiscale::verification::-  
TemporalNumericMeasureGrammar< Iterator >::initialise( )  
[private]**

Initialisation function.

Definition at line 31 of file TemporalNumericMeasureGrammarDefinition.hpp.

Referenced by multiscale::verification::TemporalNumericMeasureGrammar< Iterator  
>::TemporalNumericMeasureGrammar().

**6.212.3.6 template<typename Iterator > void multiscale::verification::Temporal-  
NumericMeasureGrammar< Iterator >::initialiseDebugSupport( )  
[private]**

Initialise debug support.

Definition at line 121 of file TemporalNumericMeasureGrammarDefinition.hpp.

**6.212.3.7 template<typename Iterator > void multiscale::verification::Temporal-  
NumericMeasureGrammar< Iterator >::initialiseErrorHandlingSupport( )  
[private]**

Initialise the error handling routines.

Definition at line 192 of file TemporalNumericMeasureGrammarDefinition.hpp.

```
6.212.3.8 template<typename Iterator> void multiscale::verification::Temporal-
    NumericMeasureGrammar< Iterator >::initialiseGrammar( )
    [private]
```

Initialise the grammar.

Definition at line 39 of file TemporalNumericMeasureGrammarDefinition.hpp.

```
6.212.3.9 template<typename Iterator> void multiscale::verification-
    ::TemporalNumericMeasureGrammar< Iterator
    >::initialiseNumericMeasureCollectionRule( ) [private]
```

Initialise the numeric measure collection rule.

Definition at line 113 of file TemporalNumericMeasureGrammarDefinition.hpp.

```
6.212.3.10 template<typename Iterator> void multiscale::verification-
    ::TemporalNumericMeasureGrammar< Iterator
    >::initialiseNumericMeasureCollectionRuleDebugging( )
    [private]
```

Initialise debugging for the numeric measure collection rule.

Definition at line 186 of file TemporalNumericMeasureGrammarDefinition.hpp.

```
6.212.3.11 template<typename Iterator> void multiscale::verification-
    ::TemporalNumericMeasureGrammar< Iterator
    >::initialiseNumericStatisticalMeasureErrorHandlingSupport( )
    [private]
```

Initialise the numeric statistical measure error handling support.

Definition at line 216 of file TemporalNumericMeasureGrammarDefinition.hpp.

References multiscale::verification::handleUnexpectedTokenError.

```
6.212.3.12 template<typename Iterator> void multiscale::verification-
    ::TemporalNumericMeasureGrammar< Iterator
    >::initialiseNumericStatisticalMeasureRule( ) [private]
```

Initialise the numeric statistical measure rules.

Definition at line 76 of file TemporalNumericMeasureGrammarDefinition.hpp.

```
6.212.3.13 template<typename Iterator > void multiscale::verification-
    ::TemporalNumericMeasureGrammar< Iterator
    >::initialiseNumericStatisticalMeasureRuleDebugging( )
    [private]
```

Initialise debugging for the numeric statistical measure rule.

Definition at line 177 of file TemporalNumericMeasureGrammarDefinition.hpp.

```
6.212.3.14 template<typename Iterator > void multiscale::verification::Temporal-
    NumericMeasureGrammar< Iterator >::initialiseRulesDebugging( )
    [private]
```

Initialise the debugging of rules.

Definition at line 161 of file TemporalNumericMeasureGrammarDefinition.hpp.

```
6.212.3.15 template<typename Iterator > void multiscale::verification-
    ::TemporalNumericMeasureGrammar< Iterator
    >::initialiseTemporalNumericMeasureErrorHandlingSupport( )
    [private]
```

Initialise the temporal numeric measure error handling support.

Definition at line 199 of file TemporalNumericMeasureGrammarDefinition.hpp.

References multiscale::verification::handleUnexpectedTokenError.

```
6.212.3.16 template<typename Iterator > void multiscale::verification-
    ::TemporalNumericMeasureGrammar< Iterator
    >::initialiseTemporalNumericMeasureRule( ) [private]
```

Initialise the temporal numeric measure rule.

Definition at line 47 of file TemporalNumericMeasureGrammarDefinition.hpp.

```
6.212.3.17 template<typename Iterator > void multiscale::verification-
    ::TemporalNumericMeasureGrammar< Iterator
    >::initialiseTemporalNumericMeasureRuleDebugging( )
    [private]
```

Initialise debugging for the temporal numeric measure rule.

Definition at line 169 of file TemporalNumericMeasureGrammarDefinition.hpp.

## **6.212.4 Member Data Documentation**

---

```
6.212.4.1 template<typename Iterator> BinaryNumericMeasureGrammar<Iterator>
multiscale::verification::TemporalNumericMeasureGrammar< Iterator
>::binaryNumericMeasureRule [private]
```

The grammar for parsing a binary numeric measure

Definition at line 53 of file TemporalNumericMeasureGrammar.hpp.

```
6.212.4.2 template<typename Iterator> qi::rule<Iterator, Binary-
NumericTemporalAttribute(), qi::space_type>
multiscale::verification::TemporalNumericMeasureGrammar< Iterator
>::binaryNumericTemporalRule [private]
```

The rule for parsing a binary numeric temporal attribute

Definition at line 76 of file TemporalNumericMeasureGrammar.hpp.

```
6.212.4.3 template<typename Iterator> BinaryStatisticalMeasureGrammar<Iterator>
multiscale::verification::TemporalNumericMeasureGrammar< Iterator
>::binaryStatisticalMeasureRule [private]
```

The grammar for parsing a binary statistical measure

Definition at line 43 of file TemporalNumericMeasureGrammar.hpp.

```
6.212.4.4 template<typename Iterator> qi::rule<Iterator, Binary-
StatisticalNumericAttribute(), qi::space_type>
multiscale::verification::TemporalNumericMeasureGrammar< Iterator
>::binaryStatisticalNumericRule [private]
```

The rule for parsing a binary statistical numeric attribute

Definition at line 86 of file TemporalNumericMeasureGrammar.hpp.

```
6.212.4.5 template<typename Iterator> BinaryStatisticalQuantileMeasureGrammar<-
Iterator> multiscale::verification::TemporalNumericMeasureGrammar<-
Iterator >::binaryStatisticalQuantileMeasureRule [private]
```

The grammar for parsing a binary statistical quantile measure

Definition at line 46 of file TemporalNumericMeasureGrammar.hpp.

```
6.212.4.6 template<typename Iterator> qi::rule<Iterator, Binary-
StatisticalQuantileNumericAttribute(), qi::space_type>
multiscale::verification::TemporalNumericMeasureGrammar< Iterator
>::binaryStatisticalQuantileNumericRule [private]
```

The rule for parsing a binary statistical quantile numeric attribute

Definition at line 89 of file TemporalNumericMeasureGrammar.hpp.

**6.212.4.7 template<typename Iterator> qi::rule<Iterator, NumericMeasureCollectionAttribute(), qi::space\_type>  
multiscale::verification::TemporalNumericMeasureGrammar< Iterator  
>::numericMeasureCollectionRule [private]**

The rule for parsing numeric measure collections

Definition at line 93 of file TemporalNumericMeasureGrammar.hpp.

**6.212.4.8 template<typename Iterator> NumericStateVariableGrammar<Iterator>  
multiscale::verification::TemporalNumericMeasureGrammar< Iterator  
>::numericStateVariableRule [private]**

The grammar for parsing a numeric state variable

Definition at line 64 of file TemporalNumericMeasureGrammar.hpp.

**6.212.4.9 template<typename Iterator> qi::rule<Iterator, NumericStatisticalMeasureAttribute(), qi::space\_type>  
multiscale::verification::TemporalNumericMeasureGrammar< Iterator  
>::numericStatisticalMeasureRule [private]**

The rule for parsing a numeric statistical measure

Definition at line 80 of file TemporalNumericMeasureGrammar.hpp.

**6.212.4.10 template<typename Iterator> SpatialMeasureCollectionGrammar<Iterator>  
multiscale::verification::TemporalNumericMeasureGrammar< Iterator  
>::spatialMeasureCollectionRule [private]**

The grammar for parsing a spatial measure collection

Definition at line 60 of file TemporalNumericMeasureGrammar.hpp.

**6.212.4.11 template<typename Iterator> TemporalNumericCollectionGrammar<-  
Iterator> multiscale::verification::TemporalNumericMeasureGrammar<  
Iterator >::temporalNumericCollectionRule [private]**

The grammar for parsing a temporal numeric collection

Definition at line 57 of file TemporalNumericMeasureGrammar.hpp.

```
6.212.4.12 template<typename Iterator> qi::rule<Iterator,
TemporalNumericMeasureAttribute(), qi::space_type>
multiscale::verification::TemporalNumericMeasureGrammar< Iterator
>::temporalNumericMeasureRule [private]
```

The rule for parsing a temporal numeric measure

Definition at line 70 of file TemporalNumericMeasureGrammar.hpp.

```
6.212.4.13 template<typename Iterator> UnaryNumericMeasureGrammar<Iterator>
multiscale::verification::TemporalNumericMeasureGrammar< Iterator
>::unaryNumericMeasureRule [private]
```

The grammar for parsing a unary numeric measure

Definition at line 50 of file TemporalNumericMeasureGrammar.hpp.

```
6.212.4.14 template<typename Iterator> qi::rule<Iterator,
UnaryNumericTemporalAttribute(), qi::space_type>
multiscale::verification::TemporalNumericMeasureGrammar< Iterator
>::unaryNumericTemporalRule [private]
```

The rule for parsing a unary numeric temporal attribute

Definition at line 73 of file TemporalNumericMeasureGrammar.hpp.

```
6.212.4.15 template<typename Iterator> UnaryStatisticalMeasureGrammar<Iterator>
multiscale::verification::TemporalNumericMeasureGrammar< Iterator
>::unaryStatisticalMeasureRule [private]
```

The grammar for parsing a unary statistical measure

Definition at line 40 of file TemporalNumericMeasureGrammar.hpp.

```
6.212.4.16 template<typename Iterator> qi::rule<Iterator,
UnaryStatisticalNumericAttribute(), qi::space_type>
multiscale::verification::TemporalNumericMeasureGrammar< Iterator
>::unaryStatisticalNumericRule [private]
```

The rule for parsing a unary statistical numeric attribute

Definition at line 83 of file TemporalNumericMeasureGrammar.hpp.

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

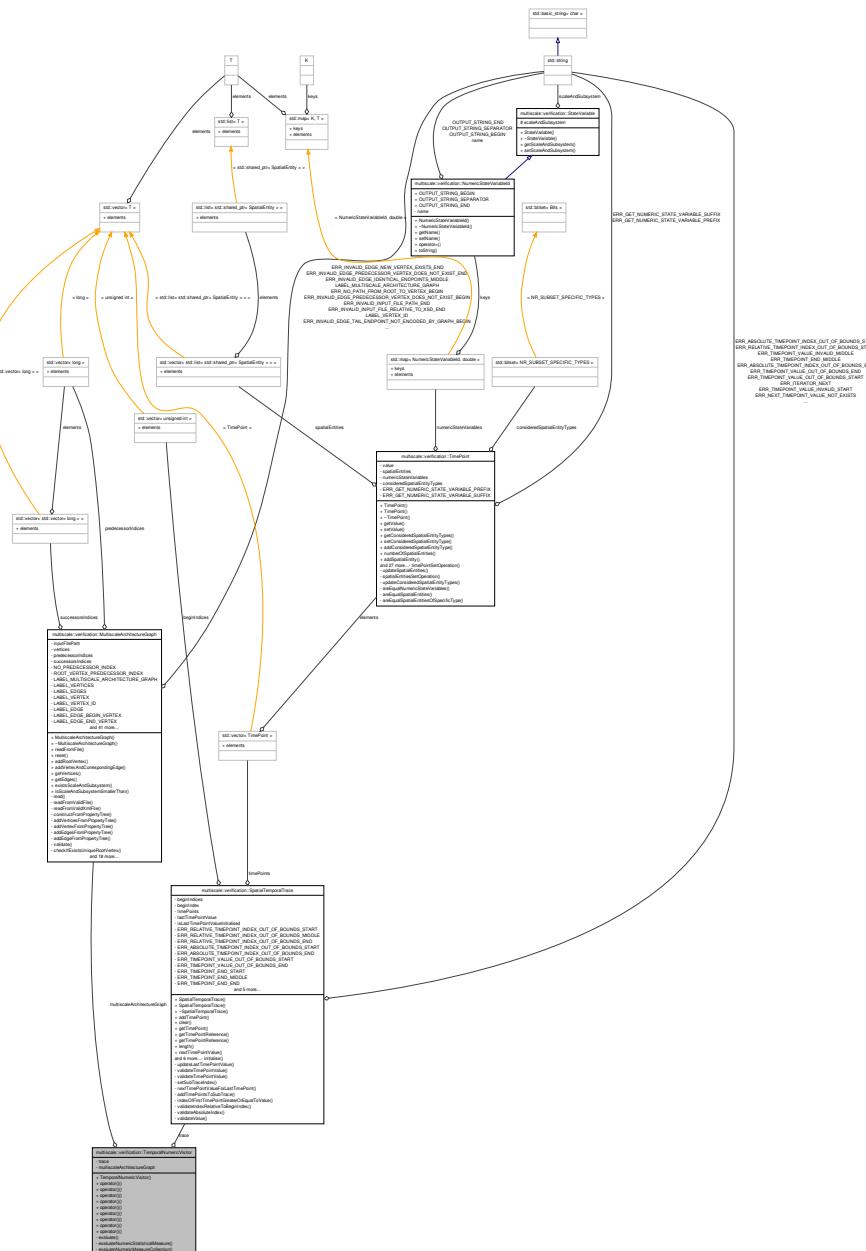
- TemporalNumericMeasureGrammar.hpp
- TemporalNumericMeasureGrammarDefinition.hpp

**6.213 multiscale::verification::TemporalNumericVisitor Class - Reference**

Class for evaluating temporal numeric measures.

```
#include <TemporalNumericVisitor.hpp>
```

## Collaboration diagram for multiscale::verification::TemporalNumericVisitor:



## Public Member Functions

- `TemporalNumericVisitor` (`SpatialTemporalTrace &trace`, `const MultiscaleArchitectureGraph &multiscaleArchitectureGraph`)

- double `operator()` (const `TemporalNumericMeasureAttribute` &temporalNumericMeasure) const
 

*Overloading the "()" operator for the `TemporalNumericMeasureAttribute` alternative.*
- double `operator()` (double realNumber) const
 

*Overloading the "()" operator for the real number alternative.*
- double `operator()` (const `NumericStateVariableAttribute` &numericStateVariable) const
 

*Overloading the "()" operator for the `NumericStateVariableAttribute` alternative.*
- double `operator()` (const `NumericStatisticalMeasureAttribute` &numericStatisticalMeasure) const
 

*Overloading the "()" operator for the `NumericStatisticalMeasureAttribute` alternative.*
- double `operator()` (const `UnaryNumericTemporalAttribute` &unaryNumericTemporalMeasure) const
 

*Overloading the "()" operator for the `UnaryNumericTemporalAttribute` alternative.*
- double `operator()` (const `BinaryNumericTemporalAttribute` &binaryNumericTemporalMeasure) const
 

*Overloading the "()" operator for the `BinaryNumericTemporalAttribute` alternative.*
- double `operator()` (const `UnaryStatisticalNumericAttribute` &unaryStatisticalNumericAttribute) const
 

*Overloading the "()" operator for the `UnaryStatisticalNumericAttribute` alternative.*
- double `operator()` (const `BinaryStatisticalNumericAttribute` &binaryStatisticalNumericAttribute) const
 

*Overloading the "()" operator for the `BinaryStatisticalNumericAttribute` alternative.*
- double `operator()` (const `BinaryStatisticalQuantileNumericAttribute` &binaryStatisticalQuantileNumericAttribute) const
 

*Overloading the "()" operator for the `BinaryStatisticalQuantileNumericAttribute` alternative.*

## Private Member Functions

- double `evaluate` (const `TemporalNumericMeasureType` &temporalNumericMeasure) const
 

*Evaluate the given temporal numeric measure considering the trace field.*
- double `evaluateNumericStatisticalMeasure` (const `NumericStatisticalMeasureType` &numericStatisticalMeasure) const
 

*Evaluate the given numeric statistical measure considering the trace field.*
- std::vector< double > `evaluateNumericMeasureCollection` (const `NumericMeasureCollectionAttribute` &numericMeasureCollection) const
 

*Evaluate the given numeric measure collection considering the trace field.*

## Private Attributes

- `SpatialTemporalTrace` & `trace`
- const `MultiscaleArchitectureGraph` & `multiscaleArchitectureGraph`

### 6.213.1 Detailed Description

Class for evaluating temporal numeric measures.

Definition at line 19 of file TemporalNumericVisitor.hpp.

### 6.213.2 Constructor & Destructor Documentation

**6.213.2.1 multiscale::verification::TemporalNumericVisitor::TemporalNumericVisitor ( SpatialTemporalTrace & trace, const MultiscaleArchitectureGraph & multiscaleArchitectureGraph ) [inline]**

Definition at line 30 of file TemporalNumericVisitor.hpp.

Referenced by evaluate(), and evaluateNumericStatisticalMeasure().

### 6.213.3 Member Function Documentation

**6.213.3.1 double multiscale::verification::TemporalNumericVisitor::evaluate ( const TemporalNumericMeasureType & temporalNumericMeasure ) const [inline, private]**

Evaluate the given temporal numeric measure considering the trace field.

#### Parameters

|                                 |                                    |
|---------------------------------|------------------------------------|
| <i>temporal-Numeric-Measure</i> | The given temporal numeric measure |
|---------------------------------|------------------------------------|

Definition at line 198 of file TemporalNumericVisitor.hpp.

References multiscaleArchitectureGraph, TemporalNumericVisitor(), and trace.

Referenced by operator()().

**6.213.3.2 std::vector< double > multiscale::verification::TemporalNumericVisitor::evaluateNumericMeasureCollection ( const NumericMeasureCollectionAttribute & numericMeasureCollection ) const [inline, private]**

Evaluate the given numeric measure collection considering the trace field.

#### Parameters

|                                   |                                      |
|-----------------------------------|--------------------------------------|
| <i>numeric-Measure-Collection</i> | The given numeric measure collection |
|-----------------------------------|--------------------------------------|

Definition at line 255 of file TemporalNumericVisitor.hpp.

References multiscaleArchitectureGraph, multiscale::verification::NumericMeasureCollectionAttribute::numericMeasureCollection, and trace.

Referenced by operator()().

**6.213.3.3 double multiscale::verification::TemporalNumericVisitor::evaluate-  
NumericStatisticalMeasure ( const NumericStatisticalMeasureType &  
numericStatisticalMeasure ) const [inline, private]**

Evaluate the given numeric statistical measure considering the trace field.

**Parameters**

|                                              |                                       |
|----------------------------------------------|---------------------------------------|
| <i>numeric-<br/>Statistical-<br/>Measure</i> | The given numeric statistical measure |
|----------------------------------------------|---------------------------------------|

Definition at line 215 of file TemporalNumericVisitor.hpp.

References multiscaleArchitectureGraph, TemporalNumericVisitor(), and trace.

Referenced by operator()().

**6.213.3.4 double multiscale::verification::TemporalNumericVisitor::operator() ( const  
TemporalNumericMeasureAttribute & temporalNumericMeasure ) const  
[inline]**

Overloading the "(") operator for the [TemporalNumericMeasureAttribute](#) alternative.

**Parameters**

|                                           |                              |
|-------------------------------------------|------------------------------|
| <i>temporal-<br/>Numeric-<br/>Measure</i> | The temporal numeric measure |
|-------------------------------------------|------------------------------|

Definition at line 39 of file TemporalNumericVisitor.hpp.

References evaluate(), and multiscale::verification::TemporalNumericMeasureAttribute-  
::temporalNumericMeasure.

**6.213.3.5 double multiscale::verification::TemporalNumericVisitor::operator() ( double  
realNumber ) const [inline]**

Overloading the "(") operator for the real number alternative.

**Parameters**

|                   |                 |
|-------------------|-----------------|
| <i>realNumber</i> | The real number |
|-------------------|-----------------|

Definition at line 48 of file TemporalNumericVisitor.hpp.

**6.213.3.6** double multiscale::verification::TemporalNumericVisitor::operator() ( const NumericStateVariableAttribute & *numericStateVariable* ) const [inline]

Overloading the "(" operator for the [NumericStateVariableAttribute](#) alternative.

**Parameters**

|                                         |                            |
|-----------------------------------------|----------------------------|
| <i>numeric-<br/>State-<br/>Variable</i> | The numeric state variable |
|-----------------------------------------|----------------------------|

Definition at line 57 of file TemporalNumericVisitor.hpp.

References multiscale::verification::NumericStateVariableEvaluator::evaluate(), multiscale::verification::SpatialTemporalTrace::getTimePointReference(), multiscale-ArchitectureGraph, and trace.

**6.213.3.7** double multiscale::verification::TemporalNumericVisitor::operator() ( const NumericStatisticalMeasureAttribute & *numericStatisticalMeasure* ) const [inline]

Overloading the "(" operator for the [NumericStatisticalMeasureAttribute](#) alternative.

**Parameters**

|                                              |                                           |
|----------------------------------------------|-------------------------------------------|
| <i>numeric-<br/>Statistical-<br/>Measure</i> | The numeric statistical measure attribute |
|----------------------------------------------|-------------------------------------------|

Definition at line 76 of file TemporalNumericVisitor.hpp.

References evaluateNumericStatisticalMeasure(), and multiscale::verification::NumericStatisticalMeasureAttribute::numericStatisticalMeasure.

**6.213.3.8** double multiscale::verification::TemporalNumericVisitor::operator() ( const UnaryNumericTemporalAttribute & *unaryNumericTemporalMeasure* ) const [inline]

Overloading the "(" operator for the [UnaryNumericTemporalAttribute](#) alternative.

**Parameters**

|                                                      |                                    |
|------------------------------------------------------|------------------------------------|
| <i>unary-<br/>Numeric-<br/>Temporal-<br/>Measure</i> | The unary numeric temporal measure |
|------------------------------------------------------|------------------------------------|

Definition at line 87 of file TemporalNumericVisitor.hpp.

References multiscale::verification::NumericEvaluator::evaluate(), evaluate(), multiscale::verification::UnaryNumericTemporalAttribute::temporalNumericMeasure, multiscale::verification::UnaryNumericTemporalAttribute::unaryNumericMeasure, and multiscale::verification::UnaryNumericMeasureAttribute::unaryNumericMeasureType.

```
6.213.3.9 double multiscale::verification::TemporalNumericVisitor::operator() ( const
    BinaryNumericTemporalAttribute & binaryNumericTemporalMeasure ) const
    [inline]
```

Overloading the "(") operator for the [BinaryNumericTemporalAttribute](#) alternative.

#### Parameters

|                                        |                                     |
|----------------------------------------|-------------------------------------|
| <i>binary-Numeric-Temporal-Measure</i> | The binary numeric temporal measure |
|----------------------------------------|-------------------------------------|

Definition at line 106 of file TemporalNumericVisitor.hpp.

References multiscale::verification::BinaryNumericTemporalAttribute::binaryNumericMeasure, multiscale::verification::BinaryNumericMeasureAttribute::binaryNumericMeasureType, multiscale::verification::NumericEvaluator::evaluate(), evaluate(), multiscale::verification::BinaryNumericTemporalAttribute::firstTemporalNumericMeasure, and multiscale::verification::BinaryNumericTemporalAttribute::secondTemporalNumericMeasure.

```
6.213.3.10 double multiscale::verification::TemporalNumericVisitor::operator() ( const
    UnaryStatisticalNumericAttribute & unaryStatisticalNumericAttribute ) const
    [inline]
```

Overloading the "(") operator for the [UnaryStatisticalNumericAttribute](#) alternative.

#### Parameters

|                                            |                                         |
|--------------------------------------------|-----------------------------------------|
| <i>unary-Statistical-Numeric-Attribute</i> | The unary statistical numeric attribute |
|--------------------------------------------|-----------------------------------------|

Definition at line 129 of file TemporalNumericVisitor.hpp.

References multiscale::verification::NumericEvaluator::evaluate(), evaluateNumericMeasureCollection(), multiscale::verification::UnaryStatisticalNumericAttribute::numericMeasureCollection, multiscale::verification::UnaryStatisticalNumericAttribute::unaryStatisticalMeasure, and multiscale::verification::UnaryStatisticalMeasureAttribute::unaryStatisticalMeasureType.

---

**6.213.3.11 double multiscale::verification::TemporalNumericVisitor::operator() ( const  
BinaryStatisticalNumericAttribute & *binaryStatisticalNumericAttribute* ) const  
[inline]**

Overloading the "(") operator for the [BinaryStatisticalNumericAttribute](#) alternative.

**Parameters**

|                                                            |                                          |
|------------------------------------------------------------|------------------------------------------|
| <i>binary-<br/>Statistical-<br/>Numeric-<br/>Attribute</i> | The binary statistical numeric attribute |
|------------------------------------------------------------|------------------------------------------|

Definition at line 148 of file TemporalNumericVisitor.hpp.

References multiscale::verification::BinaryStatisticalNumericAttribute::binaryStatisticalMeasure, multiscale::verification::BinaryStatisticalMeasureAttribute::binaryStatisticalMeasureType, multiscale::verification::NumericEvaluator::evaluate(), evaluateNumericMeasureCollection(), multiscale::verification::BinaryStatisticalNumericAttribute::firstNumericMeasureCollection, and multiscale::verification::BinaryStatisticalNumericAttribute::secondNumericMeasureCollection.

**6.213.3.12 double multiscale::verification::TemporalNumericVisitor::operator()  
( const BinaryStatisticalQuantileNumericAttribute &  
*binaryStatisticalQuantileNumericAttribute* ) const [inline]**

Overloading the "(") operator for the [BinaryStatisticalQuantileNumericAttribute](#) alternative.

**Parameters**

|                                                                          |                                                   |
|--------------------------------------------------------------------------|---------------------------------------------------|
| <i>binary-<br/>Statistical-<br/>Quantile-<br/>Numeric-<br/>Attribute</i> | The binary statistical quantile numeric attribute |
|--------------------------------------------------------------------------|---------------------------------------------------|

Definition at line 173 of file TemporalNumericVisitor.hpp.

References multiscale::verification::BinaryStatisticalQuantileNumericAttribute::binaryStatisticalQuantileMeasure, multiscale::verification::NumericEvaluator::evaluate(), evaluateNumericMeasureCollection(), multiscale::verification::BinaryStatisticalQuantileNumericAttribute::numericMeasureCollection, and multiscale::verification::BinaryStatisticalQuantileNumericAttribute::parameter.

---

#### 6.213.4 Member Data Documentation

6.213.4.1 `const MultiscaleArchitectureGraph& multiscale::verification-  
::TemporalNumericVisitor::multiscaleArchitectureGraph  
[private]`

The considered multiscale architecture graph

Definition at line 26 of file TemporalNumericVisitor.hpp.

Referenced by `evaluate()`, `evaluateNumericMeasureCollection()`, `evaluateNumericStatisticalMeasure()`, and `operator()()`.

6.213.4.2 `SpatialTemporalTrace& multiscale::verification::TemporalNumeric-  
Visitor::trace [private]`

The considered spatial temporal trace

Definition at line 24 of file TemporalNumericVisitor.hpp.

Referenced by `evaluate()`, `evaluateNumericMeasureCollection()`, `evaluateNumericStatisticalMeasure()`, and `operator()()`.

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

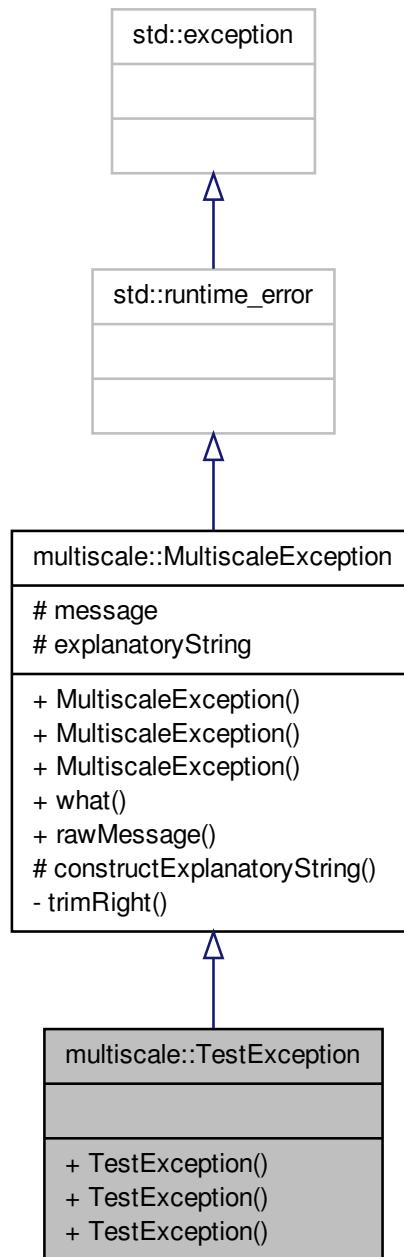
- TemporalNumericVisitor.hpp

## 6.214 multiscale::TestException Class Reference

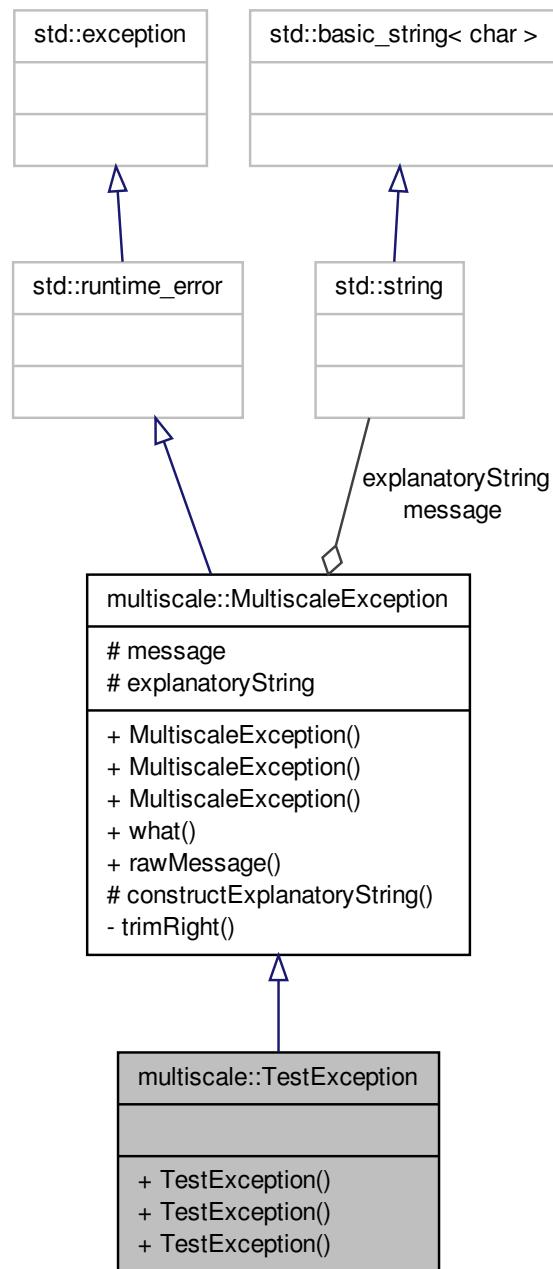
Class for representing testing exceptions.

```
#include <TestException.hpp>
```

Inheritance diagram for multiscale::TestException:



Collaboration diagram for multiscale::TestException:



## Public Member Functions

- [TestException \(\)](#)
- [TestException \(const std::string &file, int line, const std::string &msg\)](#)
- [TestException \(const std::string &file, int line, const char \\*msg\)](#)

### 6.214.1 Detailed Description

Class for representing testing exceptions.

Definition at line 12 of file `TestException.hpp`.

### 6.214.2 Constructor & Destructor Documentation

#### 6.214.2.1 multiscale::TestException::TestException( ) [inline]

Definition at line 16 of file `TestException.hpp`.

#### 6.214.2.2 multiscale::TestException::TestException( const std::string & file, int line, const std::string & msg ) [inline, explicit]

Definition at line 18 of file `TestException.hpp`.

#### 6.214.2.3 multiscale::TestException::TestException( const std::string & file, int line, const char \* msg ) [inline, explicit]

Definition at line 23 of file `TestException.hpp`.

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

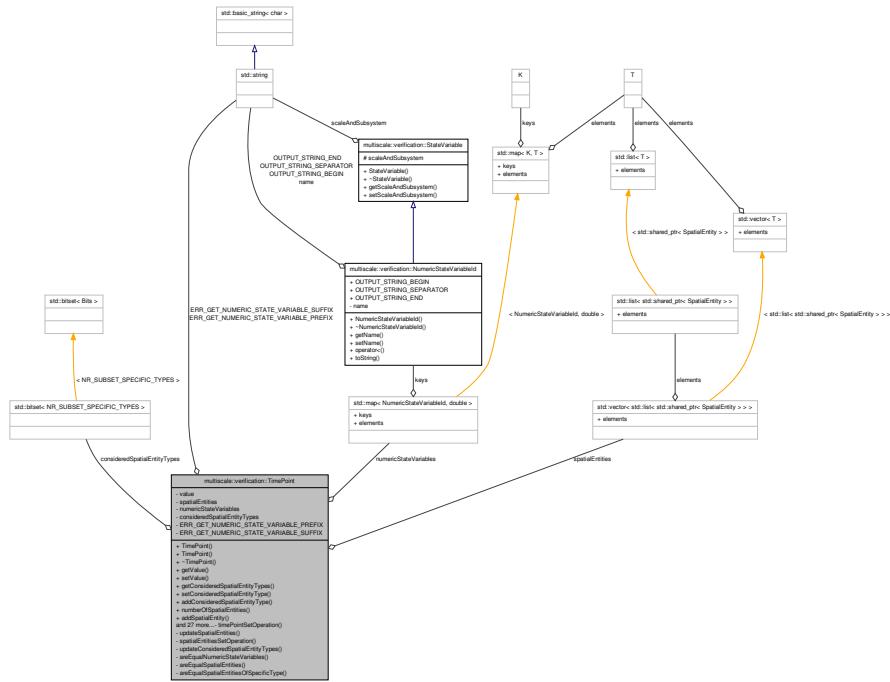
- `TestException.hpp`

## 6.215 multiscale::verification::TimePoint Class Reference

Class for representing a timepoint.

```
#include <TimePoint.hpp>
```

## Collaboration diagram for multiscale::verification::TimePoint:



## Public Member Functions

- `TimePoint` (`unsigned long value=std::numeric_limits< unsigned long >::max()`)
  - `TimePoint` (`const TimePoint &timePoint`)
  - `~TimePoint ()`
  - `unsigned long getValue () const`

*Get the value of the timepoint.*
  - `void setValue (unsigned long value)`

*Set the value of the timepoint.*
  - `std::bitset < NR_SUBSET_SPECIFIC_TYPES > getConsideredSpatialEntity-Types ()`

*Get the considered spatial entity type.*
  - `void setConsideredSpatialEntityType (const SubsetSpecificType &considered-SpatialEntityType)`

*Set the considered spatial entity type to the given type.*
  - `void addConsideredSpatialEntityType (const SubsetSpecificType &considered-SpatialEntityType)`

*Add the given type to the collection of considered spatial entity types.*
  - `double numberOfSpatialEntities () const`

*Get the number of considered spatial entities.*

- void `addSpatialEntity` (const std::shared\_ptr< `SpatialEntity` > &spatialEntity, const `SubsetSpecificType` &spatialEntityType)
 

*Add a spatial entity of the given type to the list of spatial entities.*
- void `addSpatialEntityAndType` (const std::shared\_ptr< `SpatialEntity` > &spatialEntity, const `SubsetSpecificType` &spatialEntityType)
 

*Add a spatial entity and its corresponding type to the timepoint.*
- void `addNumericStateVariable` (const `NumericStateVariableId` &id, double value)
 

*Add a numeric state variable to the map.*
- bool `containsNumericStateVariable` (const `NumericStateVariableId` &id)
 

*Check if the numeric state variable with the given id exists.*
- bool `containsNumericStateVariable` (const `NumericStateVariableId` &id) const
 

*Check if the numeric state variable with the given id exists.*
- bool `containsSpatialEntity` (const std::shared\_ptr< `SpatialEntity` > &spatialEntity, const `SubsetSpecificType` &spatialEntityType)
 

*Check if an identical valued spatial entity is already contained by the timepoint.*
- bool `containsSpatialEntity` (const std::shared\_ptr< `SpatialEntity` > &spatialEntity, const std::size\_t &spatialEntityTypeIndex)
 

*Check if an identical valued spatial entity is already contained by the timepoint.*
- bool `containsSpatialEntity` (const std::shared\_ptr< `SpatialEntity` > &spatialEntity, const `SubsetSpecificType` &spatialEntityType) const
 

*Check if an identical valued spatial entity is already contained by the timepoint.*
- bool `containsSpatialEntity` (const std::shared\_ptr< `SpatialEntity` > &spatialEntity, const std::size\_t &spatialEntityTypeIndex) const
 

*Check if an identical valued spatial entity is already contained by the timepoint.*
- std::list< std::shared\_ptr < `SpatialEntity` > ::iterator `getSpatialEntitiesBeginIterator` (const `SubsetSpecificType` &spatialEntityType)
 

*Get the begin iterator for the spatial entities of the given type.*
- std::list< std::shared\_ptr < `SpatialEntity` > ::iterator `getSpatialEntitiesBeginIterator` (const std::size\_t &spatialEntityTypeIndex)
 

*Get the begin iterator for the spatial entities corresponding to the given type index.*
- std::list< std::shared\_ptr < `SpatialEntity` > ::const\_iterator `getSpatialEntitiesBeginIterator` (const `SubsetSpecificType` &spatialEntityType) const
 

*Get the begin iterator for the spatial entities of the given type.*
- std::list< std::shared\_ptr < `SpatialEntity` > ::const\_iterator `getSpatialEntitiesBeginIterator` (const std::size\_t &spatialEntityTypeIndex) const
 

*Get the begin iterator for the spatial entities corresponding to the given type index.*
- std::list< std::shared\_ptr < `SpatialEntity` > ::iterator `getSpatialEntitiesEndIterator` (const `SubsetSpecificType` &spatialEntityType)
 

*Get the end iterator for the spatial entities of the given type.*
- std::list< std::shared\_ptr < `SpatialEntity` > ::iterator `getSpatialEntitiesEndIterator` (const std::size\_t &spatialEntityTypeIndex)
 

*Get the end iterator for the spatial entities corresponding to the given type index.*
- std::list< std::shared\_ptr < `SpatialEntity` > ::const\_iterator `getSpatialEntitiesEndIterator` (const `SubsetSpecificType` &spatialEntityType) const
 

*Get the end iterator for the spatial entities of the given type.*

- `Get the end iterator for the spatial entities of the given type.`  
`std::list< std::shared_ptr < SpatialEntity > >::const_iterator getSpatialEntities-EndIterator (const std::size_t &spatialEntityTypeIndex) const`
- `Get the end iterator for the spatial entities corresponding to the given type index.`  
`std::vector< std::shared_ptr < SpatialEntity > > getConsideredSpatialEntities () const`
- `Get the collection of considered spatial entities.`  
`double getNumericStateVariableValue (const NumericStateVariableId &id) const`
- `Get the value of the numeric state variable with the given id.`  
`std::map < NumericStateVariableId, double >::iterator getNumericState-VariablesBeginIterator ()`
- `Get the begin iterator for the collection of numeric state variables.`  
`std::map < NumericStateVariableId, double >::const_iterator getNumericState-VariablesBeginIterator () const`
- `Get the begin iterator for the collection of numeric state variables.`  
`std::map < NumericStateVariableId, double >::iterator getNumericState-VariablesEndIterator ()`
- `Get the end iterator for the collection of numeric state variables.`  
`std::map < NumericStateVariableId, double >::const_iterator getNumericState-VariablesEndIterator () const`
- `Get the end iterator for the collection of numeric state variables.`  
`void timePointDifference (const TimePoint &timePoint)`
- `Compute the difference of this timepoint and the given timepoint (spatial entities only)`  
`void timePointIntersection (const TimePoint &timePoint)`
- `Compute the intersection of this timepoint and the given timepoint (spatial entities only)`  
`void timePointUnion (const TimePoint &timePoint)`
- `Compute the union of this timepoint and the given timepoint (spatial entities only)`  
`std::list< std::shared_ptr < SpatialEntity > >::iterator removeSpatialEntity (std::list< std::shared_ptr < SpatialEntity > >::iterator &position, const Subset-SpecificType &spatialEntityType)`
- `Remove the spatial entity of the given type from the given position.`  
`bool operator==(const TimePoint &rhsTimepoint)`
- `Check if two timepoints (this instance and the provided one) are equal.`  
`bool operator!=(const TimePoint &rhsTimepoint)`
- `Check if two timepoints (this instance and the provided one) are different (i.e. not equal)`

## Private Member Functions

- `void timePointSetOperation (const TimePoint &timePoint, const SubsetOperation-Type &setOperationType)`  
`Compute the given set operation of this timepoint and the given timepoint considering the given set operation type.`

- void `updateSpatialEntities` (const `TimePoint` &timePoint, const `SubsetOperationType` &setOperationType)
 

*Apply the set operation to the collection of spatial entities from this and the given timepoint.*
- std::list< std::shared\_ptr < `SpatialEntity` > > `spatialEntitiesSetOperation` (const `TimePoint` &timePoint, const `SubsetOperationType` &setOperationType, const - `SubsetSpecificType` &spatialEntityTypeIndex)
 

*Compute the given set operation on the set of spatial entities of the given type from this and the provided timepoint.*
- void `updateConsideredSpatialEntityTypes` (const std::bitset< `NR_SUBSET_SPECIFIC_TYPES` > &consideredSpatialEntityTypes, const `SubsetOperationType` &setOperationType)
 

*Update the considered spatial entity type of this timepoint considering the given set-OperationType and consideredSpatialEntityTypes.*
- bool `areEqualNumericStateVariables` (const `TimePoint` &rhsTimepoint)
 

*Check if this and the provided timepoint contain the same numeric state variables.*
- bool `areEqualSpatialEntities` (const `TimePoint` &rhsTimepoint)
 

*Check if this and the provided timepoint contain the same spatial entities.*
- bool `areEqualSpatialEntitiesOfSpecificType` (const `TimePoint` &rhsTimepoint, const std::size\_t &spatialEntityTypeIndex)
 

*Check if this and the provided timepoint contain the same spatial entities of the given type.*

## Private Attributes

- unsigned long `value`
- std::vector< std::list < std::shared\_ptr < `SpatialEntity` > > > `spatialEntities`
- std::map < `NumericStateVariableId`, double > `numericStateVariables`
- std::bitset < `NR_SUBSET_SPECIFIC_TYPES` > `consideredSpatialEntityTypes`

## Static Private Attributes

- static const std::string `ERR_GET_NUMERIC_STATE_VARIABLE_PREFIX` = "- The numeric state variable with the given id "
- static const std::string `ERR_GET_NUMERIC_STATE_VARIABLE_SUFFIX` = " does not exist."

### 6.215.1 Detailed Description

Class for representing a timepoint.

Definition at line 24 of file `TimePoint.hpp`.

### 6.215.2 Constructor & Destructor Documentation

6.215.2.1 `TimePoint::TimePoint ( unsigned long value = std::numeric_limits<unsigned long>::max () )`

Definition at line 13 of file TimePoint.cpp.

6.215.2.2 `TimePoint::TimePoint ( const TimePoint & timePoint )`

Definition at line 20 of file TimePoint.cpp.

6.215.2.3 `TimePoint::~TimePoint ( )`

Definition at line 25 of file TimePoint.cpp.

### 6.215.3 Member Function Documentation

6.215.3.1 `void TimePoint::addConsideredSpatialEntityType ( const SubsetSpecificType & consideredSpatialEntityType )`

Add the given type to the collection of considered spatial entity types.

The flag corresponding to the given spatial entity type is set to 1 and all other flags will keep their previous value.

#### Parameters

|                                                                           |                                    |
|---------------------------------------------------------------------------|------------------------------------|
| <code><i>considered-</i><br/><i>Spatial-</i><br/><i>EntityType</i></code> | The considered spatial entity type |
|---------------------------------------------------------------------------|------------------------------------|

Definition at line 51 of file TimePoint.cpp.

References `multiscale::verification::subsetspecific::computeSubsetSpecificTypeIndex()`, `consideredSpatialEntityTypes`, and `multiscale::verification::subsetspecific::validateSubsetSpecificType()`.

Referenced by `addSpatialEntityAndType()`.

6.215.3.2 `void TimePoint::addNumericStateVariable ( const NumericStateVariableId & id, double value )`

Add a numeric state variable to the map.

If a numeric state variable with the same id exists then the value of the existing numeric state variable will be replaced by the provided new value.

**Parameters**

|              |                                                   |
|--------------|---------------------------------------------------|
| <i>id</i>    | The id (name, type) of the numeric state variable |
| <i>value</i> | The value of the numeric state variable           |

Definition at line 88 of file TimePoint.cpp.

References numericStateVariables, and value.

Referenced by multiscale::verification::TemporalDataReader::addNumericStateVariablesToTimePoint(), multiscale::verification::SpatialTemporalDataReader::addNumericStateVariableToTimePoint(), and multiscale::verification::MSTMLSubfilesMerger::addNumericStateVariableToTimepoint().

#### 6.215.3.3 void TimePoint::addSpatialEntity ( const std::shared\_ptr< SpatialEntity > & *spatialEntity*, const SubsetSpecificType & *spatialEntityType* )

Add a spatial entity of the given type to the list of spatial entities.

This method does not update the considered spatial entity types. If the considered spatial entity types should be updated then use the [addSpatialEntityAndType\(\)](#) method instead.

**Parameters**

|                          |                                |
|--------------------------|--------------------------------|
| <i>spatialEntity</i>     | The spatial entity             |
| <i>spatialEntityType</i> | The type of the spatial entity |

Definition at line 71 of file TimePoint.cpp.

References multiscale::verification::subsetspecific::computeSubsetSpecificTypeIndex(), spatialEntities, and multiscale::verification::subsetspecific::validateSubsetSpecificType().

Referenced by addSpatialEntityAndType().

#### 6.215.3.4 void TimePoint::addSpatialEntityAndType ( const std::shared\_ptr< SpatialEntity > & *spatialEntity*, const SubsetSpecificType & *spatialEntityType* )

Add a spatial entity and its corresponding type to the timepoint.

The spatial entity is added to the timepoint and the corresponding spatial entity type flag is set to true in the collection of considered spatial entity types.

**Parameters**

|                          |                                |
|--------------------------|--------------------------------|
| <i>spatialEntity</i>     | The spatial entity             |
| <i>spatialEntityType</i> | The type of the spatial entity |

Definition at line 81 of file TimePoint.cpp.

References addConsideredSpatialEntityType(), and addSpatialEntity().

Referenced by multiscale::verification::SpatialTemporalDataReader::addSpatialEntityToTimePoint(), multiscale::verification::MSTMLSubfilesMerger::addSpatialEntityToTimepoint(), and multiscaletest::ModelCheckerTest::InitialiseSpatioTemporalTraceWithAreaValues().

#### 6.215.3.5 bool TimePoint::areEqualNumericStateVariables ( const TimePoint & *rhsTimepoint* ) [private]

Check if this and the provided timepoint contain the same numeric state variables

##### Parameters

|                                 |                                                                                   |
|---------------------------------|-----------------------------------------------------------------------------------|
| <i>rhs-</i><br><i>Timepoint</i> | The timepoint against which this timepoint's numeric state variables are compared |
|---------------------------------|-----------------------------------------------------------------------------------|

Definition at line 396 of file TimePoint.cpp.

References containsNumericStateVariable(), and numericStateVariables.

Referenced by operator==().

#### 6.215.3.6 bool TimePoint::areEqualSpatialEntities ( const TimePoint & *rhsTimepoint* ) [private]

Check if this and the provided timepoint contain the same spatial entities.

##### Parameters

|                                 |                                                                                   |
|---------------------------------|-----------------------------------------------------------------------------------|
| <i>rhs-</i><br><i>Timepoint</i> | The timepoint against which this timepoint's numeric state variables are compared |
|---------------------------------|-----------------------------------------------------------------------------------|

Definition at line 417 of file TimePoint.cpp.

References areEqualSpatialEntitiesOfSpecificType(), and multiscale::verification::NR\_-SUBSET\_SPECIFIC\_TYPES.

Referenced by operator==().

#### 6.215.3.7 bool TimePoint::areEqualSpatialEntitiesOfSpecificType ( const TimePoint & *rhsTimepoint*, const std::size\_t & *spatialEntityTypeIndex* ) [private]

Check if this and the provided timepoint contain the same spatial entities of the given type.

##### Parameters

|                                 |                                                                                   |
|---------------------------------|-----------------------------------------------------------------------------------|
| <i>rhs-</i><br><i>Timepoint</i> | The timepoint against which this timepoint's numeric state variables are compared |
| <i>spatialEntityTypeIndex</i>   | The index of the considered spatial entity type                                   |

Definition at line 431 of file TimePoint.cpp.

References containsSpatialEntity(), and spatialEntities.

Referenced by areEqualSpatialEntities().

**6.215.3.8 bool TimePoint::containsNumericStateVariable ( const NumericStateVariableId & id )**

Check if the numeric state variable with the given id exists.

#### Parameters

|           |                                      |
|-----------|--------------------------------------|
| <i>id</i> | The id of the numeric state variable |
|-----------|--------------------------------------|

Definition at line 93 of file TimePoint.cpp.

References numericStateVariables.

Referenced by multiscale::verification::MSTMLSubfilesMerger::addNumericStateVariableToTimepoint(), and areEqualNumericStateVariables().

**6.215.3.9 bool TimePoint::containsNumericStateVariable ( const NumericStateVariableId & id ) const**

Check if the numeric state variable with the given id exists.

#### Parameters

|           |                                      |
|-----------|--------------------------------------|
| <i>id</i> | The id of the numeric state variable |
|-----------|--------------------------------------|

Definition at line 98 of file TimePoint.cpp.

References numericStateVariables.

**6.215.3.10 bool TimePoint::containsSpatialEntity ( const std::shared\_ptr< SpatialEntity > & spatialEntity, const SubsetSpecificType & spatialEntityType )**

Check if an identical valued spatial entity is already contained by the timepoint.

#### Parameters

|                          |                                |
|--------------------------|--------------------------------|
| <i>spatialEntity</i>     | The considered spatial entity  |
| <i>spatialEntityType</i> | The type of the spatial entity |

Definition at line 103 of file TimePoint.cpp.

References multiscale::verification::subsetspecific::computeSubsetSpecificTypeIndex().

Referenced by multiscale::verification::MSTMLSubfilesMerger::addSpatialEntityToTimepoint(), areEqualSpatialEntitiesOfSpecificType(), and containsSpatialEntity().

**6.215.3.11 bool TimePoint::containsSpatialEntity ( const std::shared\_ptr< SpatialEntity > & spatialEntity, const std::size\_t & spatialEntityTypeIndex )**

Check if an identical valued spatial entity is already contained by the timepoint.

**Parameters**

|                               |                                      |
|-------------------------------|--------------------------------------|
| <i>spatialEntity</i>          | The considered spatial entity        |
| <i>spatialEntityTypeIndex</i> | The index of the spatial entity type |

Definition at line 115 of file TimePoint.cpp.

References spatialEntities.

**6.215.3.12 bool TimePoint::containsSpatialEntity ( const std::shared\_ptr< SpatialEntity > & spatialEntity, const SubsetSpecificType & spatialEntityType ) const**

Check if an identical valued spatial entity is already contained by the timepoint.

**Parameters**

|                          |                                |
|--------------------------|--------------------------------|
| <i>spatialEntity</i>     | The considered spatial entity  |
| <i>spatialEntityType</i> | The type of the spatial entity |

Definition at line 134 of file TimePoint.cpp.

References multiscale::verification::subsetsspecific::computeSubsetSpecificTypeIndex(), and containsSpatialEntity().

**6.215.3.13 bool TimePoint::containsSpatialEntity ( const std::shared\_ptr< SpatialEntity > & spatialEntity, const std::size\_t & spatialEntityTypeIndex ) const**

Check if an identical valued spatial entity is already contained by the timepoint.

**Parameters**

|                               |                                      |
|-------------------------------|--------------------------------------|
| <i>spatialEntity</i>          | The considered spatial entity        |
| <i>spatialEntityTypeIndex</i> | The index of the spatial entity type |

Definition at line 146 of file TimePoint.cpp.

References spatialEntities.

---

**6.215.3.14 std::vector< std::shared\_ptr< SpatialEntity > >  
TimePoint::getConsideredSpatialEntities( ) const**

Get the collection of considered spatial entities.

Definition at line 237 of file TimePoint.cpp.

References consideredSpatialEntityTypes, multiscale::verification::NR\_SUBSET\_SPECIFIC\_TYPES, and spatialEntities.

Referenced by multiscale::verification::TimePointEvaluator::getSpatialMeasureValues().

**6.215.3.15 std::bitset< NR\_SUBSET\_SPECIFIC\_TYPES >  
TimePoint::getConsideredSpatialEntityTypes( )**

Get the considered spatial entity type.

Definition at line 38 of file TimePoint.cpp.

References consideredSpatialEntityTypes.

Referenced by multiscale::verification::ConstraintVisitor::evaluateScaleAndSubsystemConstraint(), and multiscale::verification::ConstraintVisitor::evaluateSpatialMeasureConstraint().

**6.215.3.16 std::map< NumericStateVariableId, double >::iterator  
TimePoint::getNumericStateVariablesBeginIterator( )**

Get the begin iterator for the collection of numeric state variables.

Definition at line 263 of file TimePoint.cpp.

References numericStateVariables.

Referenced by multiscale::verification::MSTMLSubfilesMerger::addNumericStateVariablesToResultingTraceTimepoint(), and multiscale::verification::SpatialTemporalDataWriter::addNumericStateVariablesToTimepointTree().

**6.215.3.17 std::map< NumericStateVariableId, double >::const\_iterator  
TimePoint::getNumericStateVariablesBeginIterator( ) const**

Get the begin iterator for the collection of numeric state variables.

Definition at line 268 of file TimePoint.cpp.

References numericStateVariables.

**6.215.3.18 std::map< NumericStateVariableId, double >::iterator  
TimePoint::getNumericStateVariablesEndIterator( )**

Get the end iterator for the collection of numeric state variables.

Definition at line 273 of file TimePoint.cpp.

References numericStateVariables.

Referenced by multiscale::verification::MSTMLSubfilesMerger::addNumericStateVariablesToResultingTraceTimepoint(), and multiscale::verification::SpatialTemporalDataWriter::addNumericStateVariablesToTimepointTree().

**6.215.3.19 std::map< NumericStateVariableId, double >::const\_iterator  
TimePoint::getNumericStateVariablesEndIterator( ) const**

Get the end iterator for the collection of numeric state variables.

Definition at line 278 of file TimePoint.cpp.

References numericStateVariables.

**6.215.3.20 double TimePoint::getNumericStateVariableValue( const  
NumericStateVariableId & id ) const**

Get the value of the numeric state variable with the given id.

Get the value of the numeric state variable with the given id if it exists and throw an exception otherwise

#### Parameters

|           |                                      |
|-----------|--------------------------------------|
| <i>id</i> | The id of the numeric state variable |
|-----------|--------------------------------------|

Definition at line 251 of file TimePoint.cpp.

References ERR\_GET\_NUMERIC\_STATE\_VARIABLE\_PREFIX, ERR\_GET\_NUMERIC\_STATE\_VARIABLE\_SUFFIX, and numericStateVariables.

Referenced by multiscale::verification::NumericStateVariableEvaluator::evaluate().

**6.215.3.21 std::list< std::shared\_ptr< SpatialEntity > >::iterator  
TimePoint::getSpatialEntitiesBeginIterator( const SubsetSpecificType  
& spatialEntityType )**

Get the begin iterator for the spatial entities of the given type.

Return the spatial entities begin iterator if the considered spatial entity type is of the given type. Otherwise return the spatial entities end iterator.

#### Parameters

|                          |                                  |
|--------------------------|----------------------------------|
| <i>spatialEntityType</i> | The type of the spatial entities |
|--------------------------|----------------------------------|

Definition at line 165 of file TimePoint.cpp.

References      multiscale::verification::subsetsspecific::computeSubsetSpecificTypeIndex(), and multiscale::verification::subsetsspecific::validateSubsetSpecificType().

Referenced by multiscale::verification::SpatialTemporalDataWriter::addSpatialEntitiesOfSpecificTypeToTimepointTree(), multiscale::verification::MSTMLSubfilesMerger::addSpatialEntitiesToResultingTraceTimepoint(), multiscale::verification::ConstraintVisitor::filterSpatialEntitiesWrtScaleAndSubsystemConsideringEqualComparator(), multiscale::verification::ConstraintVisitor::filterSpatialEntitiesWrtScaleAndSubsystemConsideringNonEqualComparator(), multiscale::verification::ConstraintVisitor::filterSpatialEntitiesWrtSpatialMeasure(), getSpatialEntitiesBeginIterator(), multiscaletest::ModelCheckerTest::InitialiseSpatioTemporalTraceWithAreaValues(), and spatialEntitiesSetOperation().

**6.215.3.22 std::list< std::shared\_ptr< SpatialEntity > >::iterator  
TimePoint::getSpatialEntitiesBeginIterator ( const std::size\_t &  
spatialEntityTypeIndex )**

Get the begin iterator for the spatial entities corresponding to the given type index.

Return the spatial entities begin iterator if the considered spatial entity type is of the given type. Otherwise return the spatial entities end iterator.

#### Parameters

|                               |                                            |
|-------------------------------|--------------------------------------------|
| <i>spatialEntityTypeIndex</i> | The index of the the spatial entities type |
|-------------------------------|--------------------------------------------|

Definition at line 174 of file TimePoint.cpp.

References consideredSpatialEntityTypes, spatialEntities, and multiscale::verification::subsetsspecific::validateSubsetSpecificTypeIndex().

**6.215.3.23 std::list< std::shared\_ptr< SpatialEntity > >::const\_iterator  
TimePoint::getSpatialEntitiesBeginIterator ( const SubsetSpecificType  
& spatialEntityType ) const**

Get the begin iterator for the spatial entities of the given type.

Return the spatial entities begin iterator if the considered spatial entity type is of the given type. Otherwise return the spatial entities end iterator.

#### Parameters

|                          |                                  |
|--------------------------|----------------------------------|
| <i>spatialEntityType</i> | The type of the spatial entities |
|--------------------------|----------------------------------|

Definition at line 185 of file TimePoint.cpp.

References      multiscale::verification::subsetsspecific::computeSubsetSpecificTypeIndex(), getSpatialEntitiesBeginIterator(), and multiscale::verification::subsetsspecific::validateSubsetSpecificType().

```
6.215.3.24 std::list< std::shared_ptr< SpatialEntity > >::const_iterator
    TimePoint::getSpatialEntitiesBeginIterator ( const std::size_t &
        spatialEntityTypeIndex ) const
```

Get the begin iterator for the spatial entities corresponding to the given type index.

Return the spatial entities begin iterator if the considered spatial entity type is of the given type. Otherwise return the spatial entities end iterator.

#### Parameters

|                                           |                                            |
|-------------------------------------------|--------------------------------------------|
| <i>spatialEntity-</i><br><i>TypeIndex</i> | The index of the the spatial entities type |
|-------------------------------------------|--------------------------------------------|

Definition at line 194 of file TimePoint.cpp.

References consideredSpatialEntityTypes, spatialEntities, and multiscale::verification::subsetspecific::validateSubsetSpecificTypeIndex().

```
6.215.3.25 std::list< std::shared_ptr< SpatialEntity > >::iterator
    TimePoint::getSpatialEntitiesEndIterator ( const SubsetSpecificType &
        spatialEntityType )
```

Get the end iterator for the spatial entities of the given type.

#### Parameters

|                                      |                                  |
|--------------------------------------|----------------------------------|
| <i>spatialEntity-</i><br><i>Type</i> | The type of the spatial entities |
|--------------------------------------|----------------------------------|

Definition at line 205 of file TimePoint.cpp.

References multiscale::verification::subsetspecific::computeSubsetSpecificTypeIndex(), and multiscale::verification::subsetspecific::validateSubsetSpecificType().

Referenced by multiscale::verification::SpatialTemporalDataWriter::addSpatialEntitiesOfSpecificTypeToTimepointTree(), multiscale::verification::MSTMLSubfilesMerger::addSpatialEntitiesToResultingTraceTimepoint(), multiscale::verification::ConstraintVisitor::filterSpatialEntitiesWrtScaleAndSubsystemConsideringEqualComparator(), multiscale::verification::ConstraintVisitor::filterSpatialEntitiesWrtScaleAndSubsystemConsideringNonEqualComparator(), multiscale::verification::ConstraintVisitor::filterSpatialEntitiesWrtSpatialMeasure(), getSpatialEntitiesEndIterator(), and spatialEntitiesSetOperation().

```
6.215.3.26 std::list< std::shared_ptr< SpatialEntity > >::iterator
    TimePoint::getSpatialEntitiesEndIterator ( const std::size_t &
        spatialEntityTypeIndex )
```

Get the end iterator for the spatial entities corresponding to the given type index.

**Parameters**

|                       |                                        |
|-----------------------|----------------------------------------|
| <i>spatialEntity-</i> | The index of the spatial entities type |
| <i>TypeIndex</i>      |                                        |

Definition at line 214 of file TimePoint.cpp.

References spatialEntities, and multiscale::verification::subsetspecific::validateSubsetSpecificTypeIndex().

```
6.215.3.27 std::list< std::shared_ptr< SpatialEntity > >::const_iterator
TimePoint::getSpatialEntitiesEndIterator ( const SubsetSpecificType &
spatialEntityType ) const
```

Get the end iterator for the spatial entities of the given type.

**Parameters**

|                       |                                  |
|-----------------------|----------------------------------|
| <i>spatialEntity-</i> | The type of the spatial entities |
| <i>Type</i>           |                                  |

Definition at line 221 of file TimePoint.cpp.

References multiscale::verification::subsetspecific::computeSubsetSpecificTypeIndex(), getSpatialEntitiesEndIterator(), and multiscale::verification::subsetspecific::validateSubsetSpecificType().

```
6.215.3.28 std::list< std::shared_ptr< SpatialEntity > >::const_iterator
TimePoint::getSpatialEntitiesEndIterator ( const std::size_t &
spatialEntityTypeIndex ) const
```

Get the end iterator for the spatial entities corresponding to the given type index.

**Parameters**

|                       |                                        |
|-----------------------|----------------------------------------|
| <i>spatialEntity-</i> | The index of the spatial entities type |
| <i>TypeIndex</i>      |                                        |

Definition at line 230 of file TimePoint.cpp.

References spatialEntities, and multiscale::verification::subsetspecific::validateSubsetSpecificTypeIndex().

```
6.215.3.29 unsigned long TimePoint::getValue ( ) const
```

Get the value of the timepoint.

Definition at line 28 of file TimePoint.cpp.

References value.

Referenced by multiscale::verification::SpatialTemporalDataWriter::addValueAttributeToTimepointTree(), multiscale::verification::MSTMLSubfilesMerger::areMismatchingTimepointValues(), multiscale::verification::LogicPropertyVisitor::evaluateChangeLhsTemporalNumericMeasure(), multiscale::verification::SpatialTemporalTrace::updateLastTimePointValue(), and multiscale::verification::SpatialTemporalTrace::validateTimePointValue().

#### 6.215.3.30 double TimePoint::numberOfSpatialEntities ( ) const

Get the number of considered spatial entities.

Definition at line 58 of file TimePoint.cpp.

References consideredSpatialEntityTypes, multiscale::verification::NR\_SUBSET\_SPECIFIC\_TYPES, and spatialEntities.

#### 6.215.3.31 bool TimePoint::operator!= ( const TimePoint & rhsTimepoint )

Check if two timepoints (this instance and the provided one) are different (i.e. not equal)

##### Parameters

|                      |                                                                 |
|----------------------|-----------------------------------------------------------------|
| <i>rhs-Timepoint</i> | The provided timepoint against which this timepoint is compared |
|----------------------|-----------------------------------------------------------------|

Definition at line 320 of file TimePoint.cpp.

References operator==().

#### 6.215.3.32 bool TimePoint::operator== ( const TimePoint & rhsTimepoint )

Check if two timepoints (this instance and the provided one) are equal.

##### Parameters

|                      |                                                                 |
|----------------------|-----------------------------------------------------------------|
| <i>rhs-Timepoint</i> | The provided timepoint against which this timepoint is compared |
|----------------------|-----------------------------------------------------------------|

Definition at line 306 of file TimePoint.cpp.

References areEqualNumericStateVariables(), areEqualSpatialEntities(), and value.

Referenced by operator!=().

#### 6.215.3.33 std::list< std::shared\_ptr< SpatialEntity > >::iterator TimePoint::removeSpatialEntity ( std::list< std::shared\_ptr< SpatialEntity >>::iterator & position, const SubsetSpecificType & spatialEntityType )

Remove the spatial entity of the given type from the given position.

**Parameters**

|                           |                                                  |
|---------------------------|--------------------------------------------------|
| <i>position</i>           | The position of the spatial entity to be removed |
| <i>spatialEntity-Type</i> | The type of the spatial entity                   |

Definition at line 298 of file TimePoint.cpp.

References multiscale::verification::subsetsspecific::computeSubsetSpecificTypeIndex(), spatialEntities, and multiscale::verification::subsetsspecific::validateSubsetSpecificType().

Referenced by multiscale::verification::ConstraintVisitor::filterSpatialEntitiesWrtScaleAndSubsystemConsideringEqualComparator(), multiscale::verification::ConstraintVisitor::filterSpatialEntitiesWrtScaleAndSubsystemConsideringNonEqualComparator(), multiscale::verification::ConstraintVisitor::filterSpatialEntitiesWrtSpatialMeasure(), and multiscaletest::ModelCheckerTest::InitialiseSpatioTemporalTraceWithAreaValues().

#### 6.215.3.34 void TimePoint::setConsideredSpatialEntityType ( const SubsetSpecificType & *consideredSpatialEntityType* )

Set the considered spatial entity type to the given type.

The flag corresponding to the given spatial entity type is set to 1 and all others are set to 0.

**Parameters**

|                                      |                                             |
|--------------------------------------|---------------------------------------------|
| <i>considered-Spatial-EntityType</i> | The considered type of the spatial entities |
|--------------------------------------|---------------------------------------------|

Definition at line 43 of file TimePoint.cpp.

References multiscale::verification::subsetsspecific::computeSubsetSpecificTypeIndex(), consideredSpatialEntityTypes, and multiscale::verification::subsetsspecific::validateSubsetSpecificType().

Referenced by multiscaletest::ModelCheckerTest::InitialiseSpatioTemporalTraceWithAreaValues(), and multiscale::verification::SubsetVisitor::setTimePointConsideredSpatialEntityType().

#### 6.215.3.35 void TimePoint::setValue ( unsigned long *value* )

Set the value of the timepoint.

**Parameters**

|              |                            |
|--------------|----------------------------|
| <i>value</i> | The value of the timepoint |
|--------------|----------------------------|

Definition at line 33 of file TimePoint.cpp.

References value.

Referenced by multiscale::verification::TemporalDataReader::setTimePointValue(), multiscale::verification::SpatialTemporalDataReader::setTimePointValue(), multiscale::verification::SpatialTemporalTrace::updateLastTimePointValue(), and multiscale::verification::MSTMLSubfilesMerger::updateResultingTraceTimepointsValues().

**6.215.3.36 std::list< std::shared\_ptr< SpatialEntity > > TimePoint::spatialEntities-SetOperation ( const TimePoint & *timePoint*, const SubsetOperationType & *setOperationType*, const SubsetSpecificType & *spatialEntityTypeIndex* )  
[private]**

Compute the given set operation on the set of spatial entities of the given type from this and the provided timepoint.

#### Parameters

|                               |                                          |
|-------------------------------|------------------------------------------|
| <i>timePoint</i>              | The given timepoint                      |
| <i>set-Operation-Type</i>     | The considered set operation type        |
| <i>spatialEntityTypeIndex</i> | The considered spatial entity type index |

Definition at line 343 of file TimePoint.cpp.

References getSpatialEntitiesBeginIterator(), and getSpatialEntitiesEndIterator().

Referenced by updateSpatialEntities().

**6.215.3.37 void TimePoint::timePointDifference ( const TimePoint & *timePoint* )**

Compute the difference of this timepoint and the given timepoint (spatial entities only)

Compute the difference of this timepoint and the given timepoint by taking into account the value of consideredSpatialEntityType

Spatial entities belonging to the first and not to the second timepoint will be included in the resulting timepoint.

The consideredSpatialEntityType of the resulting timepoint will be the considered-SpatialEntityType of this timepoint.

#### Parameters

|                  |                     |
|------------------|---------------------|
| <i>timePoint</i> | The given timepoint |
|------------------|---------------------|

Definition at line 283 of file TimePoint.cpp.

References timePointSetOperation().

Referenced by multiscale::verification::SubsetVisitor::evaluateSubsetOperation(), and multiscale::verification::ConstraintVisitor::operator()().

### 6.215.3.38 void TimePoint::timePointIntersection ( const TimePoint & *timePoint* )

Compute the intersection of this timepoint and the given timepoint (spatial entities only)

Compute the intersection of this timepoint and the given timepoint by taking into account the value of consideredSpatialEntityType

Spatial entities belonging both to the first and the second timepoint will be included in the resulting timepoint.

The consideredSpatialEntityType of the resulting timepoint will be the intersection of the timepoints' consideredSpatialEntityTypes.

#### Parameters

|                  |                     |
|------------------|---------------------|
| <i>timePoint</i> | The given timepoint |
|------------------|---------------------|

Definition at line 288 of file TimePoint.cpp.

References timePointSetOperation().

Referenced by multiscale::verification::SubsetVisitor::evaluateSubsetOperation(), and multiscale::verification::ConstraintVisitor::operator()().

### 6.215.3.39 void TimePoint::timePointSetOperation ( const TimePoint & *timePoint*, const SubsetOperationType & *setOperationType* ) [private]

Compute the given set operation of this timepoint and the given timepoint considering the given set operation type.

#### Parameters

|                                     |                                   |
|-------------------------------------|-----------------------------------|
| <i>timePoint</i>                    | The given timepoint               |
| <i>set-<br/>Operation-<br/>Type</i> | The considered set operation type |

Definition at line 328 of file TimePoint.cpp.

References consideredSpatialEntityTypes, updateConsideredSpatialEntityTypes(), and updateSpatialEntities().

Referenced by timePointDifference(), timePointIntersection(), and timePointUnion().

### 6.215.3.40 void TimePoint::timePointUnion ( const TimePoint & *timePoint* )

Compute the union of this timepoint and the given timepoint (spatial entities only)

Compute the union of this timepoint and the given timepoint by taking into account the value of consideredSpatialEntityType.

Spatial entities belonging either to the first or the second timepoint will be included in the resulting timepoint.

The consideredSpatialEntityType of the resulting timepoint will be the union of the timepoints' consideredSpatialEntityTypes.

#### Parameters

|                  |                     |
|------------------|---------------------|
| <i>timePoint</i> | The given timepoint |
|------------------|---------------------|

Definition at line 293 of file TimePoint.cpp.

References timePointSetOperation().

Referenced by multiscale::verification::SubsetVisitor::evaluateSubsetOperation(), and multiscale::verification::ConstraintVisitor::operator()().

```
6.215.3.41 void TimePoint::updateConsideredSpatialEntityTypes ( const std::bitset<
    NR_SUBSET_SPECIFIC_TYPES > & consideredSpatialEntityTypes, const
    SubsetOperationType & setOperationType ) [private]
```

Update the considered spatial entity type of this timepoint considering the given setOperationType and consideredSpatialEntityTypes.

Definition at line 377 of file TimePoint.cpp.

References consideredSpatialEntityTypes.

Referenced by timePointSetOperation().

```
6.215.3.42 void TimePoint::updateSpatialEntities ( const TimePoint & timePoint, const
    SubsetOperationType & setOperationType ) [private]
```

Apply the set operation to the collection of spatial entities from this and the given timepoint.

#### Parameters

|                                     |                                   |
|-------------------------------------|-----------------------------------|
| <i>timePoint</i>                    | The given timepoint               |
| <i>set-<br/>Operation-<br/>Type</i> | The considered set operation type |

Definition at line 334 of file TimePoint.cpp.

References multiscale::verification::subsetspecific::computeSubsetSpecificType(), multiscale::verification::NR\_SUBSET\_SPECIFIC\_TYPES, spatialEntities, and spatialEntitiesSetOperation().

Referenced by timePointSetOperation().

## 6.215.4 Member Data Documentation

---

**6.215.4.1 std::bitset<NR\_SUBSET\_SPECIFIC\_TYPES> multiscale-  
::verification::TimePoint::consideredSpatialEntityTypes  
[private]**

The collection of bits recording the considered spatial entity types. The i-th bit corresponds to the i-th SubsetSpecificType enum value. If the bit is set true then the corresponding subset specific type is considered. Otherwise it is not.

Definition at line 49 of file TimePoint.hpp.

Referenced by addConsideredSpatialEntityType(), getConsideredSpatialEntities(), getConsideredSpatialEntityTypes(), getSpatialEntitiesBeginIterator(), numberOfSpatialEntities(), setConsideredSpatialEntityType(), timePointSetOperation(), and updateConsideredSpatialEntityTypes().

**6.215.4.2 const std::string TimePoint::ERR\_GET\_NUMERIC\_STATE\_VARIABLE\_  
E\_PREFIX = "The numeric state variable with the given id " [static,  
private]**

Definition at line 396 of file TimePoint.hpp.

Referenced by getNumericStateVariableValue().

**6.215.4.3 const std::string TimePoint::ERR\_GET\_NUMERIC\_STATE\_VARIABLE\_  
SUFFIX = " does not exist." [static,  
private]**

Definition at line 397 of file TimePoint.hpp.

Referenced by getNumericStateVariableValue().

**6.215.4.4 std::map<NumericStateVariableId, double> multiscale-  
::verification::TimePoint::numericStateVariables  
[private]**

The associative map for storing numeric state variables

Definition at line 46 of file TimePoint.hpp.

Referenced by addNumericStateVariable(), areEqualNumericStateVariables(), containsNumericStateVariable(), getNumericStateVariablesBeginIterator(), getNumericStateVariablesEndIterator(), and getNumericStateVariableValue().

**6.215.4.5 std::vector<std::list<std::shared\_ptr<SpatialEntity>>>  
multiscale::verification::TimePoint::spatialEntities [private]**

The meta-list of spatial entities smart pointers. The i-th spatial entities list in the meta-list corresponds to the i-th SubsetSpecificType enumeration value

Definition at line 40 of file TimePoint.hpp.

Referenced by addSpatialEntity(), areEqualSpatialEntitiesOfSpecificType(), containsSpatialEntity(), getConsideredSpatialEntities(), getSpatialEntitiesBeginIterator(), getSpatialEntitiesEndIterator(), numberofSpatialEntities(), removeSpatialEntity(), and updateSpatialEntities().

#### 6.215.4.6 unsigned long multiscale::verification::TimePoint::value [private]

The value of the timepoint within a simulation/experiment

Definition at line 28 of file TimePoint.hpp.

Referenced by addNumericStateVariable(), getValue(), operator==(), and setValue().

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

- TimePoint.hpp
- TimePoint.cpp

## 6.216 multiscale::verification::TimePointEvaluator Class Reference

Class used to evaluate timepoints.

```
#include <TimePointEvaluator.hpp>
```

### Static Public Member Functions

- static std::vector< double > [getSpatialMeasureValues](#) (const TimePoint &timePoint, const [SpatialMeasureType](#) &spatialMeasure)

*Return the spatial measure values for all considered spatial entities in the given time-point.*

### Static Private Member Functions

- static void [getSpatialMeasureValues](#) (const std::vector< std::shared\_ptr< - [SpatialEntity](#) >> &consideredSpatialEntities, std::vector< double > &spatialMeasureValues, const [SpatialMeasureType](#) &spatialMeasure)

*Return the spatial measure values for all considered spatial entities in the given time-point.*

### 6.216.1 Detailed Description

Class used to evaluate timepoints.

Definition at line 12 of file TimePointEvaluator.hpp.

### 6.216.2 Member Function Documentation

6.216.2.1 static std::vector<double> multiscale::verification::TimePointEvaluator-  
   ::getSpatialMeasureValues ( const TimePoint & *timePoint*, const  
   SpatialMeasureType & *spatialMeasure* ) [inline, static]

Return the spatial measure values for all considered spatial entities in the given time-point.

#### Parameters

|                             |                                |
|-----------------------------|--------------------------------|
| <i>timePoint</i>            | The considered timepoint       |
| <i>spatial-<br/>Measure</i> | The considered spatial measure |

Definition at line 21 of file TimePointEvaluator.hpp.

References multiscale::verification::TimePoint::getConsideredSpatialEntities().

Referenced by multiscale::verification::SpatialMeasureCollectionVisitor::operator()().

6.216.2.2 static void multiscale::verification::TimePointEvaluator::getSpatial-  
   MeasureValues ( const std::vector< std::shared\_ptr< SpatialEntity >>  
   & *consideredSpatialEntities*, std::vector< double > & *spatialMeasureValues*,  
   const SpatialMeasureType & *spatialMeasure* ) [inline, static,  
   private]

Return the spatial measure values for all considered spatial entities in the given time-point.

#### Parameters

|                                              |                                                        |
|----------------------------------------------|--------------------------------------------------------|
| <i>considered-<br/>Spatial-<br/>Entities</i> | The considered spatial entities                        |
| <i>spatial-<br/>Measure-<br/>Values</i>      | The collection of values for the given spatial measure |
| <i>spatial-<br/>Measure</i>                  | The considered spatial measure                         |

Definition at line 41 of file TimePointEvaluator.hpp.

References multiscale::verification::SpatialMeasureEvaluator::evaluate().

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

- TimePointEvaluator.hpp

## **6.217 multiscale::verification::TimeseriesComponentAttribute Class Reference**

### **6.217 multiscale::verification::TimeseriesComponentAttribute - Class Reference**

Class for representing a timeseries component attribute.

```
#include <TimeseriesComponentAttribute.hpp>
```

#### **Public Attributes**

- [TimeseriesComponentType timeseriesComponent](#)

#### **6.217.1 Detailed Description**

Class for representing a timeseries component attribute.

Definition at line 22 of file TimeseriesComponentAttribute.hpp.

#### **6.217.2 Member Data Documentation**

##### **6.217.2.1 TimeseriesComponentType multiscale::verification::Timeseries- ComponentAttribute::timeseriesComponent**

The timeseries component

Definition at line 26 of file TimeseriesComponentAttribute.hpp.

Referenced by multiscale::verification::NumericMeasureCollectionVisitor::evaluate- TimeseriesComponent().

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

- [TimeseriesComponentAttribute.hpp](#)

### **6.218 multiscale::verification::TimeseriesComponentEvaluator - Class Reference**

Class for evaluating timeseries components.

```
#include <TimeseriesComponentEvaluator.hpp>
```

#### **Classes**

- class [HomogeneousComponentEvaluator](#)
- class [UniformHomogeneousComponentEvaluator](#)

## Static Public Member Functions

- static std::vector< std::size\_t > **evaluate** (const **HomogeneousTimeseriesComponentAttribute** &homogeneousTimeseriesComponent, const std::vector< double > &values)
 

*Evaluate the homogeneous timeseries component considering the given collection of values.*
- template<typename Relation , template< typename > class HomogeneousTimeseriesComponentEvaluator>
 static std::vector< std::size\_t > **evaluateHomogeneousComponentIndices** (const std::vector< double > &values, const Relation &relation, const HomogeneousTimeseriesComponentEvaluator< Relation > &evaluator)
 

*Compute the set of (start, end) indices pointing to nonuniform homogeneous time-series components.*

### 6.218.1 Detailed Description

Class for evaluating timeseries components.

Definition at line 15 of file TimeseriesComponentEvaluator.hpp.

### 6.218.2 Member Function Documentation

6.218.2.1 static std::vector<std::size\_t> **multiscale::verification::TimeseriesComponentEvaluator::evaluate** ( const **HomogeneousTimeseriesComponentAttribute** & **homogeneousTimeseriesComponent**, const std::vector< double > & **values** )  
 [inline, static]

Evaluate the homogeneous timeseries component considering the given collection of values.

#### Parameters

|                                       |                                            |
|---------------------------------------|--------------------------------------------|
| <i>homogeneousTimeseriesComponent</i> | The given homogeneous timeseries component |
| <i>values</i>                         | The given collection of values             |

Definition at line 25 of file TimeseriesComponentEvaluator.hpp.

References **multiscale::ERR\_UNDEFINED\_ENUM\_VALUE**, **evaluateHomogeneousComponentIndices()**, and **multiscale::verification::HomogeneousTimeseriesComponentAttribute::homogeneousTimeseriesComponent**.

Referenced by **multiscale::verification::TimeseriesComponentVisitor::operator()()**, and **multiscale::verification::NumericMeasureCollectionVisitor::operator()()**.

## **6.219 multiscale::verification::TimeseriesComponentVisitor Class Reference**

---

```
6.218.2.2 template<typename Relation , template< typename > class  
HomogeneousTimeseriesComponentEvaluator> static std::vector<std::size_t>  
multiscale::verification::TimeseriesComponentEvaluator::evaluate-  
HomogeneousComponentIndices ( const std::vector< double > & values,  
const Relation & relation, const HomogeneousTimeseriesComponentEvaluator<  
Relation > & evaluator ) [inline, static]
```

Compute the set of (start, end) indices pointing to nonuniform homogeneous timeseries components.

The value of relation depending on the considered homogeneous component type is:

- (uniform) ascent: "<"
- (uniform) descent: ">"
- plateau: "="

### **Parameters**

|                  |                                |
|------------------|--------------------------------|
| <i>values</i>    | The collection of given values |
| <i>relation</i>  | The considered relation        |
| <i>evaluator</i> | The considered evaluator       |

Definition at line 87 of file TimeseriesComponentEvaluator.hpp.

Referenced by evaluate().

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

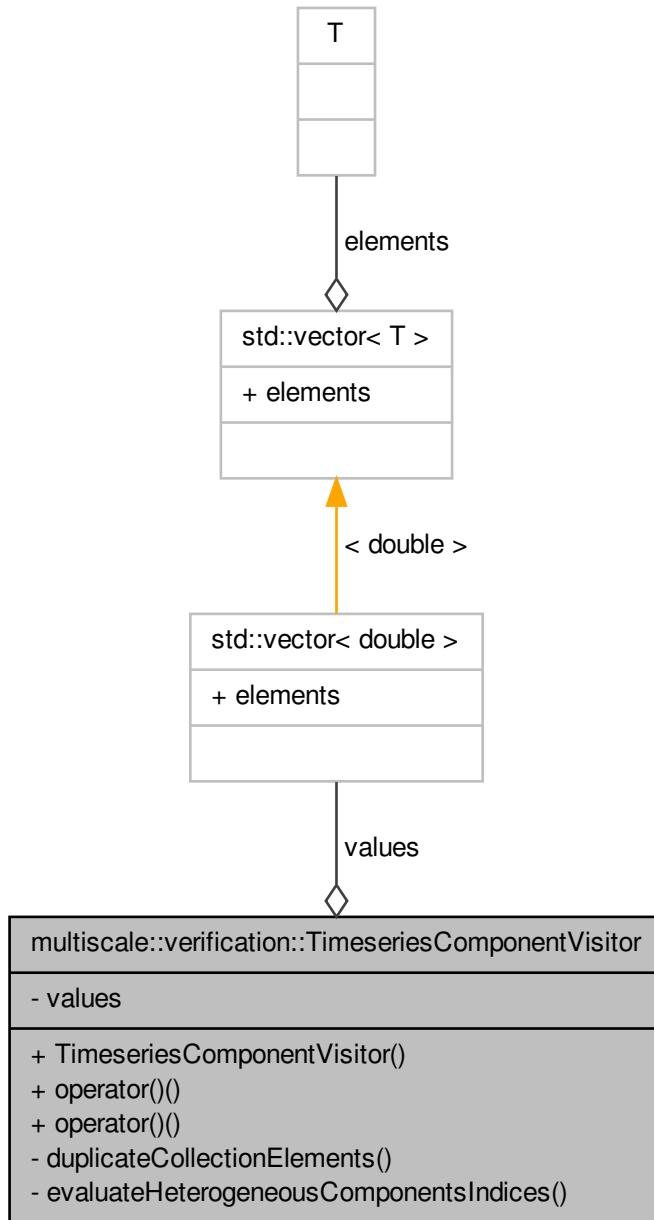
- TimeseriesComponentEvaluator.hpp

## **6.219 multiscale::verification::TimeseriesComponentVisitor Class Reference**

Class for evaluating timeseries components.

```
#include <TimeseriesComponentVisitor.hpp>
```

Collaboration diagram for multiscale::verification::TimeseriesComponentVisitor:



### Public Member Functions

- `TimeseriesComponentVisitor (const std::vector< double > &values)`
- `std::vector< std::size_t > operator() (const HeterogeneousTimeseries-ComponentAttribute &heterogeneousTimeseriesComponent) const`  
*Overloading the "(" operator for the `HeterogeneousTimeseriesComponentAttribute` alternative.*
- `std::vector< std::size_t > operator() (const HomogeneousTimeseries-ComponentAttribute &homogeneousTimeseriesComponent) const`  
*Overloading the "(" operator for the `HomogeneousTimeseriesComponentAttribute` alternative.*

### Private Member Functions

- `std::vector< std::size_t > duplicateCollectionElements (const std::vector< std::size_t > &collection) const`  
*Duplicate each element in the given indices collection.*
- `template<typename Relation> std::vector< std::size_t > evaluateHeterogeneousComponentsIndices (const std::vector< double > &values, const Relation &relation) const`  
*Compute the set of indices pointing to the heterogeneous components in the given values collection.*

### Private Attributes

- `const std::vector< double > & values`

### 6.219.1 Detailed Description

Class for evaluating timeseries components.

The output values of the visitor is a collection of indices pointing to the start and end positions of the timeseries components identified in the given collection of values. In case of heterogeneous timeseries components the start position = end position, while in the case of homogeneous timeseries components start position != end position.

Definition at line 24 of file TimeseriesComponentVisitor.hpp.

### 6.219.2 Constructor & Destructor Documentation

#### 6.219.2.1 multiscale::verification::TimeseriesComponentVisitor::TimeseriesComponentVisitor ( const std::vector< double > & values ) [inline]

Definition at line 32 of file TimeseriesComponentVisitor.hpp.

### 6.219.3 Member Function Documentation

6.219.3.1 `std::vector<std::size_t> multiscale::verification::TimeseriesComponentVisitor::duplicateCollectionElements ( const std::vector< std::size_t > & collection ) const [inline, private]`

Duplicate each element in the given indices collection.

Definition at line 77 of file TimeseriesComponentVisitor.hpp.

Referenced by operator()().

6.219.3.2 `template<typename Relation > std::vector<std::size_t> multiscale::verification::TimeseriesComponentVisitor::evaluateHeterogeneousComponentsIndices ( const std::vector< double > & values, const Relation & relation ) const [inline, private]`

Compute the set of indices pointing to the heterogeneous components in the given values collection.

A value located at index i in the collection,  $0 < i < (\text{values.size()} - 1)$ , is a heterogeneous component if and only if `relation(values[i - 1], values[i])` and `!relation(values[i], values[i + 1])` and `!equal_to(values[i], values[i + 1])`

#### Parameters

|                       |                                |
|-----------------------|--------------------------------|
| <code>values</code>   | The collection of given values |
| <code>relation</code> | The considered relation        |

Definition at line 105 of file TimeseriesComponentVisitor.hpp.

Referenced by operator()().

6.219.3.3 `std::vector<std::size_t> multiscale::verification::TimeseriesComponentVisitor::operator() ( const HeterogeneousTimeseriesComponentAttribute & heterogeneousTimeseriesComponent ) const [inline]`

Overloading the "(") operator for the [HeterogeneousTimeseriesComponentAttribute](#) alternative.

#### Parameters

|                                               |                                        |
|-----------------------------------------------|----------------------------------------|
| <code>heterogeneousTimeseriesComponent</code> | The heterogeneous timeseries component |
|-----------------------------------------------|----------------------------------------|

Definition at line 39 of file TimeseriesComponentVisitor.hpp.

References `duplicateCollectionElements()`, `multiscale::ERR_UNDEFINED_ENUM_VALUE`, `evaluateHeterogeneousComponentsIndices()`, `multiscale::verification::-`

## **6.220 multiscale::verification::TimeseriesMeasureAttribute Class Reference 1209**

---

HeterogeneousTimeseriesComponentAttribute::heterogeneousTimeseriesComponent, and values.

**6.219.3.4 std::vector<std::size\_t> multiscale::verification::TimeseriesComponentVisitor-  
::operator() ( const HomogeneousTimeseriesComponentAttribute &  
homogeneousTimeseriesComponent ) const [inline]**

Overloading the "(") operator for the [HomogeneousTimeseriesComponentAttribute](#) alternative.

### Parameters

|                                                   |                                      |
|---------------------------------------------------|--------------------------------------|
| <i>homogeneous-<br/>Timeseries-<br/>Component</i> | The homogeneous timeseries component |
|---------------------------------------------------|--------------------------------------|

Definition at line 66 of file TimeseriesComponentVisitor.hpp.

References [multiscale::verification::TimeseriesComponentEvaluator::evaluate\(\)](#), and values.

### **6.219.4 Member Data Documentation**

**6.219.4.1 const std::vector<double>& multiscale::verification::Timeseries-  
ComponentVisitor::values [private]**

The collection of considered values

Definition at line 28 of file TimeseriesComponentVisitor.hpp.

Referenced by [operator\(\)\(\)](#).

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

- [TimeseriesComponentVisitor.hpp](#)

## **6.220 multiscale::verification::TimeseriesMeasureAttribute Class - Reference**

Class for representing a timeseries measure attribute.

```
#include <TimeseriesMeasureAttribute.hpp>
```

### **Public Attributes**

- [TimeseriesMeasureType timeseriesMeasure](#)

### 6.220.1 Detailed Description

Class for representing a timeseries measure attribute.

Definition at line 28 of file TimeseriesMeasureAttribute.hpp.

### 6.220.2 Member Data Documentation

#### 6.220.2.1 **TimeseriesMeasureType multiscale::verification::TimeseriesMeasureAttribute::timeseriesMeasure**

The timeseries measure

Definition at line 32 of file TimeseriesMeasureAttribute.hpp.

Referenced by multiscale::verification::NumericMeasureCollectionVisitor::operator()().

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

- TimeseriesMeasureAttribute.hpp

## 6.221 multiscale::verification::TimeseriesMeasureTypeParser - Struct Reference

Symbol table and parser for the timeseries measure type.

```
#include <SymbolTables.hpp>
```

### Public Member Functions

- [TimeseriesMeasureTypeParser \(\)](#)

### 6.221.1 Detailed Description

Symbol table and parser for the timeseries measure type.

Definition at line 183 of file SymbolTables.hpp.

### 6.221.2 Constructor & Destructor Documentation

#### 6.221.2.1 **multiscale::verification::TimeseriesMeasureTypeParser::TimeseriesMeasureTypeParser( ) [inline]**

Definition at line 186 of file SymbolTables.hpp.

References multiscale::verification::EnteringValue.

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

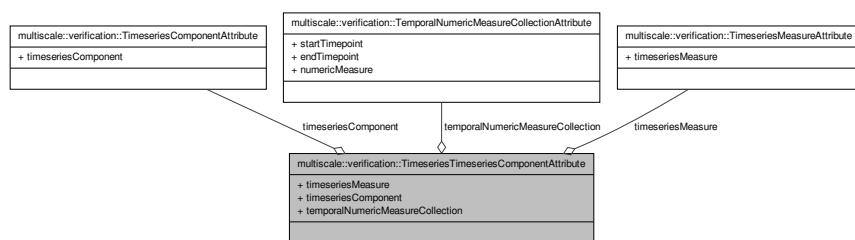
- SymbolTables.hpp

## 6.222 multiscale::verification::TimeseriesTimeseriesComponent-Attribute Class Reference

Class for representing a timeseries timeseries component attribute.

```
#include <TimeseriesTimeseriesComponentAttribute.hpp>
```

Collaboration diagram for multiscale::verification::TimeseriesTimeseriesComponent-Attribute:



### Public Attributes

- `TimeseriesMeasureAttribute timeseriesMeasure`
- `TimeseriesComponentAttribute timeseriesComponent`
- `TemporalNumericMeasureCollectionAttribute temporalNumericMeasureCollection`

#### 6.222.1 Detailed Description

Class for representing a timeseries timeseries component attribute.

Definition at line 17 of file `TimeseriesTimeseriesComponentAttribute.hpp`.

#### 6.222.2 Member Data Documentation

##### 6.222.2.1 TemporalNumericMeasureCollectionAttribute multiscale::verification::TimeseriesTimeseriesComponentAttribute- ::temporalNumericMeasureCollection

The temporal numeric collection

Definition at line 26 of file `TimeseriesTimeseriesComponentAttribute.hpp`.

Referenced by `multiscale::verification::NumericMeasureCollectionVisitor::operator()()`.

### 6.222.2.2 TimeseriesComponentAttribute multiscale::verification::Timeseries-TimeseriesComponentAttribute::timeseriesComponent

The timeseries component

Definition at line 24 of file TimeseriesTimeseriesComponentAttribute.hpp.

Referenced by multiscale::verification::NumericMeasureCollectionVisitor::operator()().

### 6.222.2.3 TimeseriesMeasureAttribute multiscale::verification::Timeseries-TimeseriesComponentAttribute::timeseriesMeasure

The timeseries measure

Definition at line 22 of file TimeseriesTimeseriesComponentAttribute.hpp.

Referenced by multiscale::verification::NumericMeasureCollectionVisitor::operator()().

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

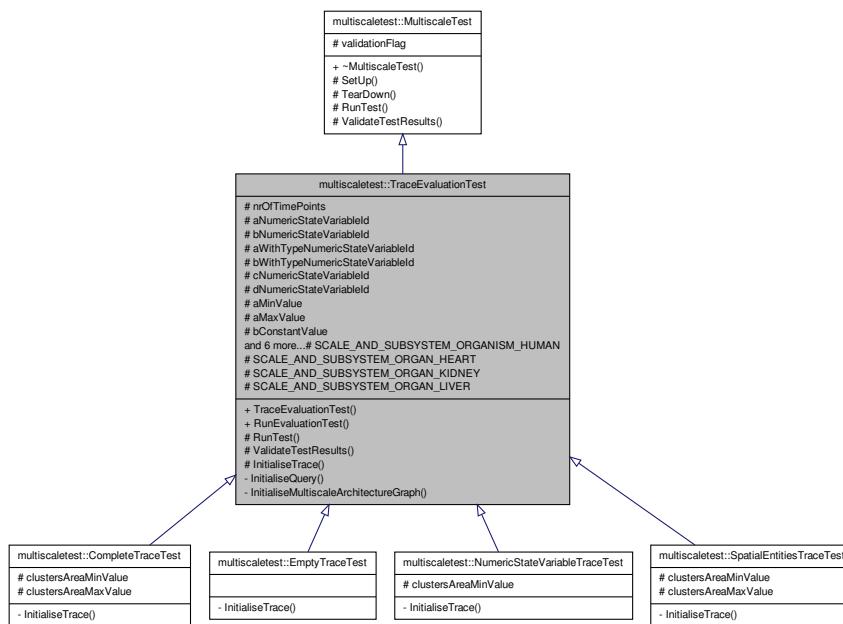
- TimeseriesTimeseriesComponentAttribute.hpp

## 6.223 multiscaletest::TraceEvaluationTest Class Reference

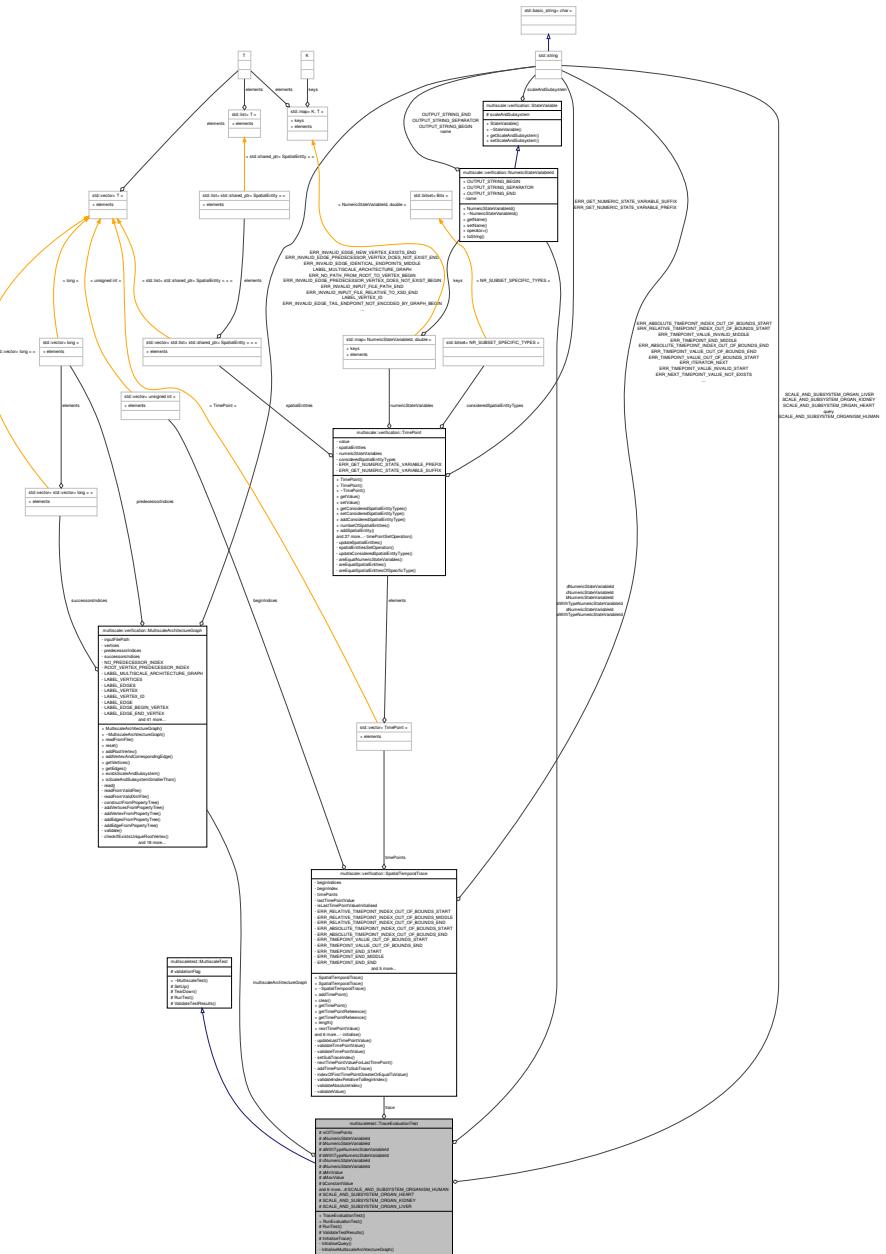
Class for testing evaluation of traces.

```
#include <TraceEvaluationTest.hpp>
```

Inheritance diagram for multiscaletest::TraceEvaluationTest:



Collaboration diagram for multiscaletest::TraceEvaluationTest:



## Public Member Functions

- `TraceEvaluationTest ()`
- `bool RunEvaluationTest (const std::string &query)`

*Run the test with the given string.*

### Protected Member Functions

- virtual void `RunTest ()` override  
*Run the test.*
- virtual void `ValidateTestResults ()` override  
*Validate the results of the test.*
- virtual void `InitialiseTrace ()=0`  
*Initialise the trace.*

### Protected Attributes

- `std::size_t nrOfTimePoints`
- `NumericStateVariableId aNumericStateVariableId`
- `NumericStateVariableId bNumericStateVariableId`
- `NumericStateVariableId aWithTypeNumericStateVariableId`
- `NumericStateVariableId bWithTypeNumericStateVariableId`
- `NumericStateVariableId cNumericStateVariableId`
- `NumericStateVariableId dNumericStateVariableId`
- `double aMinValue`
- `double aMaxValue`
- `double bConstantValue`
- `double cMinValue`
- `double cMaxValue`
- `double dConstantValue`
- `mv::SpatialTemporalTrace trace`
- `mv::MultiscaleArchitectureGraph multiscaleArchitectureGraph`
- `std::string query`
- `bool evaluationResult`

### Static Protected Attributes

- static const `std::string SCALE_AND_SUBSYSTEM_ORGANISM_HUMAN = "-Organism.Human"`
- static const `std::string SCALE_AND_SUBSYSTEM_ORGAN_HEART = "Organ.-Heart"`
- static const `std::string SCALE_AND_SUBSYSTEM_ORGAN_KIDNEY = "-Organ.Kidney"`
- static const `std::string SCALE_AND_SUBSYSTEM_ORGAN_LIVER = "Organ.-Liver"`

## Private Member Functions

- void [InitialiseQuery \(const std::string &query\)](#)  
*Initialise the query.*
- void [InitialiseMultiscaleArchitectureGraph \(\)](#)  
*Initialise the multiscale architecture graph.*

### 6.223.1 Detailed Description

Class for testing evaluation of traces.

Definition at line 23 of file TraceEvaluationTest.hpp.

### 6.223.2 Constructor & Destructor Documentation

#### 6.223.2.1 [TraceEvaluationTest::TraceEvaluationTest \( \)](#)

Definition at line 6 of file TraceEvaluationTest.cpp.

### 6.223.3 Member Function Documentation

#### 6.223.3.1 [void TraceEvaluationTest::InitialiseMultiscaleArchitectureGraph \( \)](#) [private]

Initialise the multiscale architecture graph.

Definition at line 49 of file TraceEvaluationTest.cpp.

References multiscale::verification::MultiscaleArchitectureGraph::addRootVertex(), multiscale::verification::MultiscaleArchitectureGraph::addVertexAndCorrespondingEdge(), multiscaleArchitectureGraph, multiscale::verification::MultiscaleArchitectureGraph::reset(), SCALE\_AND\_SUBSYSTEM\_ORGAN\_HEART, SCALE\_AND\_SUBSYSTEM\_ORGAN\_KIDNEY, and SCALE\_AND\_SUBSYSTEM\_ORGANISM\_HUMAN.

Referenced by RunEvaluationTest().

#### 6.223.3.2 [void TraceEvaluationTest::InitialiseQuery \( const std::string & query \)](#) [private]

Initialise the query.

##### Parameters

|              |                 |
|--------------|-----------------|
| <i>query</i> | The given query |
|--------------|-----------------|

Definition at line 45 of file TraceEvaluationTest.cpp.

References query.

Referenced by RunEvaluationTest().

**6.223.3.3 virtual void multiscaletest::TraceEvaluationTest::InitialiseTrace( )**  
[protected, pure virtual]

Initialise the trace.

Implemented in [multiscaletest::CompleteTraceTest](#), [multiscaletest::SpatialEntitiesTraceTest](#), [multiscaletest::NumericStateVariableTraceTest](#), and [multiscaletest::EmptyTraceTest](#).

Referenced by RunEvaluationTest().

**6.223.3.4 bool TraceEvaluationTest::RunEvaluationTest( const std::string & query )**

Run the test with the given string.

Parameters

|                    |                 |
|--------------------|-----------------|
| <code>query</code> | The given query |
|--------------------|-----------------|

Definition at line 21 of file TraceEvaluationTest.cpp.

References [evaluationResult](#), [InitialiseMultiscaleArchitectureGraph\(\)](#), [InitialiseQuery\(\)](#), [InitialiseTrace\(\)](#), [RunTest\(\)](#), and [ValidateTestResults\(\)](#).

**6.223.3.5 void TraceEvaluationTest::RunTest( )** [override, protected, virtual]

Run the test.

Implements [multiscaletest::MultiscaleTest](#).

Definition at line 32 of file TraceEvaluationTest.cpp.

References [multiscale::verification::AbstractSyntaxTree::evaluate\(\)](#), [evaluationResult](#), [multiscaleArchitectureGraph](#), [multiscale::verification::Parser::parse\(\)](#), [query](#), and [trace](#).

Referenced by RunEvaluationTest().

**6.223.3.6 void TraceEvaluationTest::ValidateTestResults( )** [override, protected, virtual]

Validate the results of the test.

Implements [multiscaletest::MultiscaleTest](#).

Definition at line 43 of file TraceEvaluationTest.cpp.

Referenced by RunEvaluationTest().

#### 6.223.4 Member Data Documentation

6.223.4.1 **double multiscaletest::TraceEvaluationTest::a.MaxValue** [protected]

The maximum value of numeric state variable "A"

Definition at line 43 of file TraceEvaluationTest.hpp.

6.223.4.2 **double multiscaletest::TraceEvaluationTest::a.MinValue** [protected]

The minimum value of numeric state variable "A"

Definition at line 42 of file TraceEvaluationTest.hpp.

6.223.4.3 **NumericStateVariableId multiscaletest::TraceEvaluationTest::a.Numeric-StateVariableId** [protected]

The id of the numeric state variable "A" (no type)

Definition at line 30 of file TraceEvaluationTest.hpp.

6.223.4.4 **NumericStateVariableId multiscaletest::Trace-EvaluationTest::aWithTypeNumericStateVariableId** [protected]

The id of the numeric state variable "A" (with type)

Definition at line 34 of file TraceEvaluationTest.hpp.

6.223.4.5 **double multiscaletest::TraceEvaluationTest::b.ConstantValue** [protected]

The constant value of numeric state variable "B"

Definition at line 44 of file TraceEvaluationTest.hpp.

6.223.4.6 **NumericStateVariableId multiscaletest::TraceEvaluationTest::b.NumericStateVariableId** [protected]

The id of the numeric state variable "B" (no type)

Definition at line 32 of file TraceEvaluationTest.hpp.

6.223.4.7 **NumericStateVariableId multiscaletest::Trace-EvaluationTest::bWithTypeNumericStateVariableId** [protected]

The id of the numeric state variable "B" (with type)

Definition at line 36 of file TraceEvaluationTest.hpp.

**6.223.4.8 double multiscaletest::TraceEvaluationTest::c.MaxValue [protected]**

The maximum value of numeric state variable "C"

Definition at line 46 of file TraceEvaluationTest.hpp.

**6.223.4.9 double multiscaletest::TraceEvaluationTest::c.MinValue [protected]**

The minimum value of numeric state variable "C"

Definition at line 45 of file TraceEvaluationTest.hpp.

**6.223.4.10 NumericStateVariableId multiscaletest::TraceEvaluationTest::c-NumericStateVariableId [protected]**

The id of the numeric state variable "C"

Definition at line 38 of file TraceEvaluationTest.hpp.

**6.223.4.11 double multiscaletest::TraceEvaluationTest::d.ConstantValue [protected]**

The constant value of numeric state variable "D"

Definition at line 47 of file TraceEvaluationTest.hpp.

**6.223.4.12 NumericStateVariableId multiscaletest::TraceEvaluationTest::d-NumericStateVariableId [protected]**

The id of the numeric state variable "D"

Definition at line 40 of file TraceEvaluationTest.hpp.

**6.223.4.13 bool multiscaletest::TraceEvaluationTest::evaluationResult [protected]**

The result of the evaluation

Definition at line 56 of file TraceEvaluationTest.hpp.

Referenced by RunEvaluationTest(), and RunTest().

6.223.4.14 **mv::MultiscaleArchitectureGraph multiscaletest::-TraceEvaluationTest::multiscaleArchitectureGraph** [protected]

The multiscale architecture graph

Definition at line 52 of file TraceEvaluationTest.hpp.

Referenced by InitialiseMultiscaleArchitectureGraph(), and RunTest().

6.223.4.15 **std::size\_t multiscaletest::TraceEvaluationTest::nrOfTimePoints** [protected]

The number of timepoints in the trace

Definition at line 27 of file TraceEvaluationTest.hpp.

6.223.4.16 **std::string multiscaletest::TraceEvaluationTest::query** [protected]

The query to be checked

Definition at line 54 of file TraceEvaluationTest.hpp.

Referenced by InitialiseQuery(), and RunTest().

6.223.4.17 **const std::string multiscaletest::TraceEvaluationTest::SCALE\_AN-ND\_SUBSYSTEM\_ORGAN\_HEART = "Organ.Heart"** [static, protected]

Definition at line 94 of file TraceEvaluationTest.hpp.

Referenced by InitialiseMultiscaleArchitectureGraph().

6.223.4.18 **const std::string multiscaletest::TraceEvaluationTest::SCALE\_AN-ND\_SUBSYSTEM\_ORGAN\_KIDNEY = "Organ.Kidney"** [static, protected]

Definition at line 95 of file TraceEvaluationTest.hpp.

Referenced by InitialiseMultiscaleArchitectureGraph().

6.223.4.19 **const std::string multiscaletest::TraceEvaluationTest::SCALE\_AN-ND\_SUBSYSTEM\_ORGAN\_LIVER = "Organ.Liver"** [static, protected]

Definition at line 96 of file TraceEvaluationTest.hpp.

## **6.224 multiscale::verification::UnaryNumericFilterAttribute Class Reference 1221**

---

**6.223.4.20 const std::string multiscaletest::TraceEvaluationTest::SCALE\_AND\_S-UBSYSTEM\_ORGANISM\_HUMAN = "Organism.Human" [static, protected]**

Definition at line 93 of file TraceEvaluationTest.hpp.

Referenced by InitialiseMultiscaleArchitectureGraph().

**6.223.4.21 mv::SpatialTemporalTrace multiscaletest::TraceEvaluationTest::trace [protected]**

The spatial temporal trace

Definition at line 50 of file TraceEvaluationTest.hpp.

Referenced by RunTest().

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

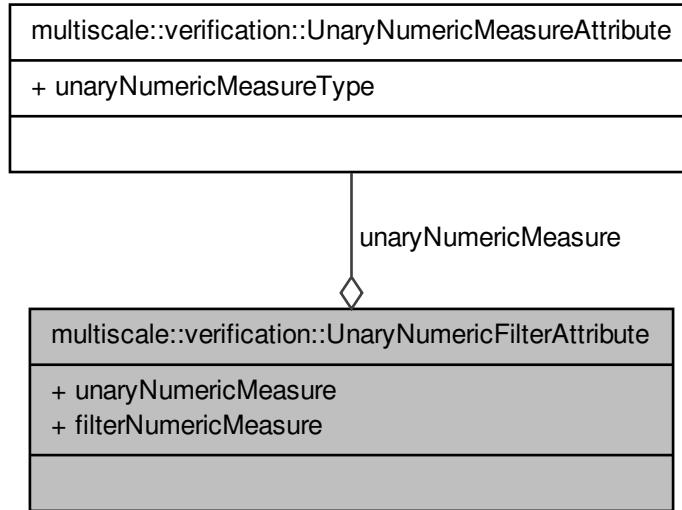
- TraceEvaluationTest.hpp
  
- TraceEvaluationTest.cpp

## **6.224 multiscale::verification::UnaryNumericFilterAttribute Class - Reference**

Class for representing a unary numeric filter attribute.

```
#include <UnaryNumericFilterAttribute.hpp>
```

Collaboration diagram for multiscale::verification::UnaryNumericFilterAttribute:



## Public Attributes

- [UnaryNumericMeasureAttribute unaryNumericMeasure](#)
- [FilterNumericMeasureAttributeType filterNumericMeasure](#)

### 6.224.1 Detailed Description

Class for representing a unary numeric filter attribute.

Definition at line 15 of file UnaryNumericFilterAttribute.hpp.

### 6.224.2 Member Data Documentation

#### 6.224.2.1 FilterNumericMeasureAttributeType multiscale::verification::UnaryNumericFilterAttribute::filterNumericMeasure

The considered filter numeric measure

Definition at line 20 of file UnaryNumericFilterAttribute.hpp.

Referenced by multiscale::verification::FilterNumericVisitor::operator()().

## **6.225 multiscale::verification::UnaryNumericMeasureAttribute Class Reference**

### **6.224.2.2 UnaryNumericMeasureAttribute multiscale::verification::Unary- NumericFilterAttribute::unaryNumericMeasure**

The unary numeric measure

Definition at line 19 of file UnaryNumericFilterAttribute.hpp.

Referenced by multiscale::verification::FilterNumericVisitor::operator()().

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

- UnaryNumericFilterAttribute.hpp

## **6.225 multiscale::verification::UnaryNumericMeasureAttribute - Class Reference**

Class for representing a unary numeric measure attribute.

```
#include <UnaryNumericMeasureAttribute.hpp>
```

### **Public Attributes**

- [UnaryNumericMeasureType unaryNumericMeasureType](#)

### **6.225.1 Detailed Description**

Class for representing a unary numeric measure attribute.

Definition at line 33 of file UnaryNumericMeasureAttribute.hpp.

### **6.225.2 Member Data Documentation**

#### **6.225.2.1 UnaryNumericMeasureType multiscale::verification::UnaryNumeric- MeasureAttribute::unaryNumericMeasureType**

The unary numeric measure type

Definition at line 37 of file UnaryNumericMeasureAttribute.hpp.

Referenced by multiscale::verification::SpatialMeasureCollectionVisitor::evaluate-  
UnaryNumericSpatialMeasureCollection(), multiscale::verification::FilterNumeric-  
Visitor::operator()(), multiscale::verification::TemporalNumericVisitor::operator()(), and  
multiscale::verification::NumericVisitor::operator()().

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

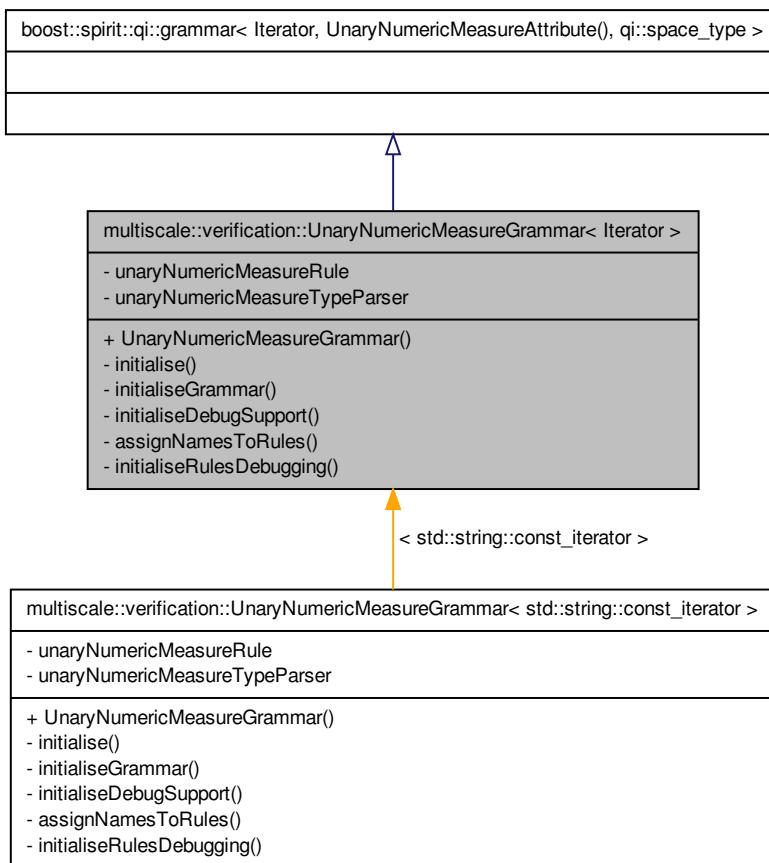
- UnaryNumericMeasureAttribute.hpp

## 6.226 multiscale::verification::UnaryNumericMeasureGrammar< - Iterator > Class Template Reference

The grammar for parsing unary numeric measure statements.

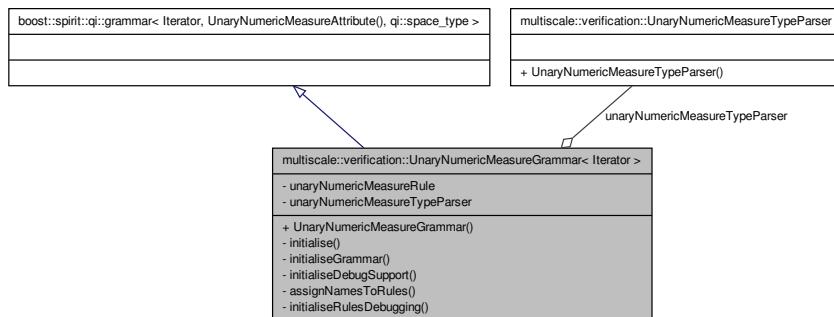
```
#include <UnaryNumericMeasureGrammar.hpp>
```

Inheritance diagram for multiscale::verification::UnaryNumericMeasureGrammar< - Iterator >:



Collaboration diagram for multiscale::verification::UnaryNumericMeasureGrammar< -

Iterator >:



## Public Member Functions

- [UnaryNumericMeasureGrammar \(\)](#)

## Private Member Functions

- void [initialise \(\)](#)  
*Initialisation function.*
- void [initialiseGrammar \(\)](#)  
*Initialise the grammar.*
- void [initialiseDebugSupport \(\)](#)  
*Initialise debug support.*
- void [assignNamesToRules \(\)](#)  
*Assign names to the rules.*
- void [initialiseRulesDebugging \(\)](#)  
*Initialise the debugging of rules.*

## Private Attributes

- [qi::rule< Iterator, UnaryNumericMeasureAttribute\(\), qi::space\\_type > unary-NumericMeasureRule](#)
- [UnaryNumericMeasureTypeParser unaryNumericMeasureTypeParser](#)

### 6.226.1 Detailed Description

```
template<typename Iterator>class multiscale::verification::UnaryNumericMeasureGrammar< Iterator >
```

The grammar for parsing unary numeric measure statements.

Definition at line 24 of file UnaryNumericMeasureGrammar.hpp.

### 6.226.2 Constructor & Destructor Documentation

6.226.2.1 `template<typename Iterator> multiscale::verification::UnaryNumericMeasureGrammar<Iterator>::UnaryNumericMeasureGrammar( )`

Definition at line 17 of file UnaryNumericMeasureGrammarDefinition.hpp.

References `multiscale::verification::UnaryNumericMeasureGrammar<Iterator>::initialise()`.

### 6.226.3 Member Function Documentation

6.226.3.1 `template<typename Iterator> void multiscale::verification::UnaryNumericMeasureGrammar<Iterator>::assignNamesToRules( ) [private]`

Assign names to the rules.

Definition at line 50 of file UnaryNumericMeasureGrammarDefinition.hpp.

6.226.3.2 `template<typename Iterator> void multiscale::verification::UnaryNumericMeasureGrammar<Iterator>::initialise( ) [private]`

Initialisation function.

Definition at line 27 of file UnaryNumericMeasureGrammarDefinition.hpp.

Referenced by `multiscale::verification::UnaryNumericMeasureGrammar<Iterator>::UnaryNumericMeasureGrammar()`.

6.226.3.3 `template<typename Iterator> void multiscale::verification::UnaryNumericMeasureGrammar<Iterator>::initialiseDebugSupport( ) [private]`

Initialise debug support.

Definition at line 41 of file UnaryNumericMeasureGrammarDefinition.hpp.

6.226.3.4 `template<typename Iterator> void multiscale::verification::UnaryNumericMeasureGrammar<Iterator>::initialiseGrammar( ) [private]`

Initialise the grammar.

Definition at line 34 of file UnaryNumericMeasureGrammarDefinition.hpp.

**6.226.3.5 template<typename Iterator > void multiscale::verification::UnaryNumericMeasureGrammar< Iterator >::initialiseRulesDebugging( ) [private]**

Initialise the debugging of rules.

Definition at line 56 of file UnaryNumericMeasureGrammarDefinition.hpp.

#### **6.226.4 Member Data Documentation**

**6.226.4.1 template<typename Iterator> qi::rule<Iterator, UnaryNumericMeasureAttribute(), qi::space\_type> multiscale::verification::UnaryNumericMeasureGrammar< Iterator >::unaryNumericMeasureRule [private]**

The rule for parsing a unary numeric measure

Definition at line 30 of file UnaryNumericMeasureGrammar.hpp.

**6.226.4.2 template<typename Iterator> UnaryNumericMeasureTypeParser multiscale::verification::UnaryNumericMeasureGrammar< Iterator >::unaryNumericMeasureTypeParser [private]**

The unary numeric measure type parser

Definition at line 35 of file UnaryNumericMeasureGrammar.hpp.

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

- UnaryNumericMeasureGrammar.hpp
- UnaryNumericMeasureGrammarDefinition.hpp

## **6.227 multiscale::verification::UnaryNumericMeasureTypeParser Struct Reference**

Symbol table and parser for the unary numeric measure type.

```
#include <SymbolTables.hpp>
```

### **Public Member Functions**

- [UnaryNumericMeasureTypeParser \(\)](#)

#### **6.227.1 Detailed Description**

Symbol table and parser for the unary numeric measure type.

Definition at line 196 of file SymbolTables.hpp.

## 6.227.2 Constructor & Destructor Documentation

### 6.227.2.1 multiscale::verification::UnaryNumericMeasureTypeParser::UnaryNumericMeasureTypeParser ( ) [inline]

Definition at line 199 of file SymbolTables.hpp.

References multiscale::verification::Ceil, multiscale::verification::Round, and multiscale::verification::Sqrt.

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

- SymbolTables.hpp

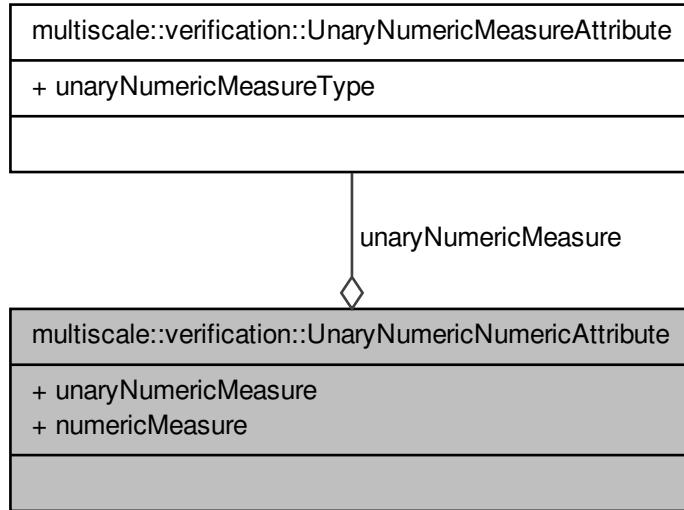
## 6.228 multiscale::verification::UnaryNumericNumericAttribute - Class Reference

Class for representing a unary numeric numeric measure attribute.

```
#include <UnaryNumericNumericAttribute.hpp>
```

## **6.228 multiscale::verification::UnaryNumericNumericAttribute Class Reference**

Collaboration diagram for multiscale::verification::UnaryNumericNumericAttribute:



### **Public Attributes**

- `UnaryNumericMeasureAttribute unaryNumericMeasure`
- `NumericMeasureType numericMeasure`

#### **6.228.1 Detailed Description**

Class for representing a unary numeric numeric measure attribute.

Definition at line 15 of file `UnaryNumericNumericAttribute.hpp`.

#### **6.228.2 Member Data Documentation**

##### **6.228.2.1 NumericMeasureType multiscale::verification::UnaryNumericNumericAttribute::numericMeasure**

The considered numeric measure

Definition at line 20 of file `UnaryNumericNumericAttribute.hpp`.

Referenced by `multiscale::verification::NumericVisitor::operator()()`.

### 6.228.2.2 UnaryNumericMeasureAttribute multiscale::verification::UnaryNumericNumericAttribute::unaryNumericMeasure

The unary numeric measure

Definition at line 19 of file UnaryNumericNumericAttribute.hpp.

Referenced by multiscale::verification::NumericVisitor::operator()().

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

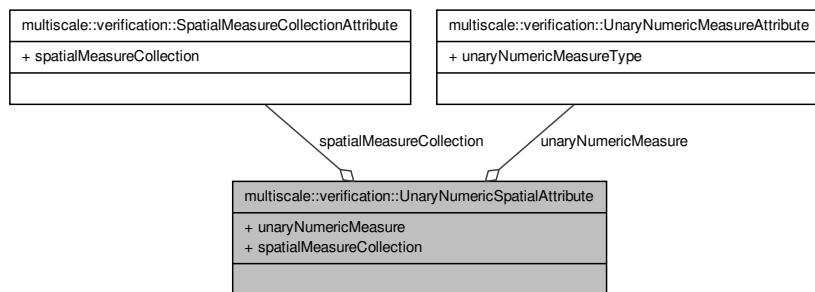
- UnaryNumericNumericAttribute.hpp

## 6.229 multiscale::verification::UnaryNumericSpatialAttribute Class Reference

Class for representing a unary numeric spatial measure collection attribute.

```
#include <UnaryNumericSpatialAttribute.hpp>
```

Collaboration diagram for multiscale::verification::UnaryNumericSpatialAttribute:



### Public Attributes

- [UnaryNumericMeasureAttribute unaryNumericMeasure](#)
- [SpatialMeasureCollectionAttribute spatialMeasureCollection](#)

### 6.229.1 Detailed Description

Class for representing a unary numeric spatial measure collection attribute.

Definition at line 15 of file UnaryNumericSpatialAttribute.hpp.

## **6.230 multiscale::verification::UnaryNumericTemporalAttribute Class Reference**

### **6.229.2 Member Data Documentation**

#### **6.229.2.1 SpatialMeasureCollectionAttribute multiscale::verification::Unary-NumericSpatialAttribute::spatialMeasureCollection**

The considered spatial measure collection

Definition at line 20 of file UnaryNumericSpatialAttribute.hpp.

Referenced by multiscale::verification::SpatialMeasureCollectionVisitor::operator()().

#### **6.229.2.2 UnaryNumericMeasureAttribute multiscale::verification::Unary-NumericSpatialAttribute::unaryNumericMeasure**

The unary numeric measure

Definition at line 19 of file UnaryNumericSpatialAttribute.hpp.

Referenced by multiscale::verification::SpatialMeasureCollectionVisitor::operator()().

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

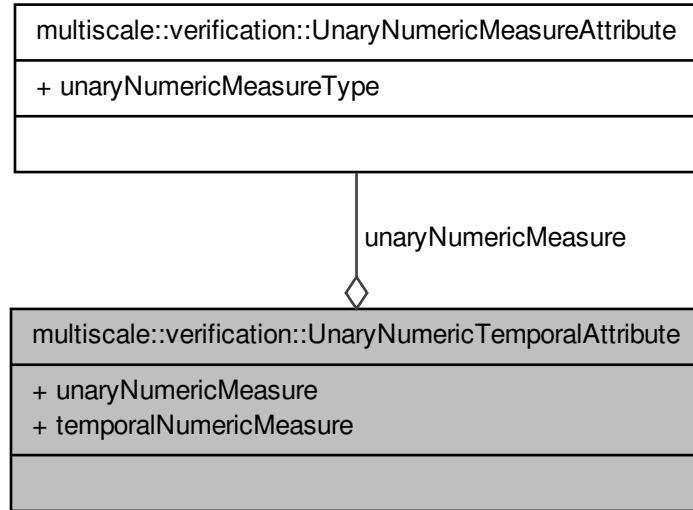
- UnaryNumericSpatialAttribute.hpp

## **6.230 multiscale::verification::UnaryNumericTemporalAttribute - Class Reference**

Class for representing a unary numeric temporal measure attribute.

```
#include <UnaryNumericTemporalAttribute.hpp>
```

Collaboration diagram for multiscale::verification::UnaryNumericTemporalAttribute:



## Public Attributes

- `UnaryNumericMeasureAttribute unaryNumericMeasure`
- `TemporalNumericMeasureType temporalNumericMeasure`

### 6.230.1 Detailed Description

Class for representing a unary numeric temporal measure attribute.

Definition at line 15 of file `UnaryNumericTemporalAttribute.hpp`.

### 6.230.2 Member Data Documentation

#### 6.230.2.1 `TemporalNumericMeasureType multiscale::verification::UnaryNumericTemporalAttribute::temporalNumericMeasure`

The considered temporal numeric measure

Definition at line 22 of file `UnaryNumericTemporalAttribute.hpp`.

Referenced by `multiscale::verification::TemporalNumericVisitor::operator()()`.

### 6.230.2.2 UnaryNumericMeasureAttribute multiscale::verification::Unary-NumericTemporalAttribute::unaryNumericMeasure

The unary numeric measure

Definition at line 20 of file UnaryNumericTemporalAttribute.hpp.

Referenced by multiscale::verification::TemporalNumericVisitor::operator()().

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

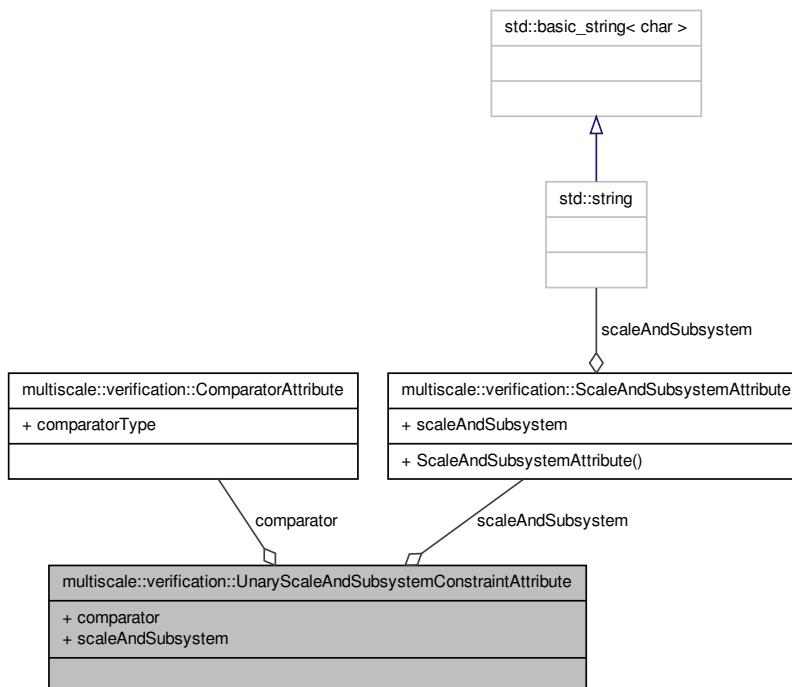
- UnaryNumericTemporalAttribute.hpp

## 6.231 multiscale::verification::UnaryScaleAndSubsystemConstraint-Attribute Class Reference

Class for representing a "unary" scale and subsystem constraint attribute.

```
#include <UnaryScaleAndSubsystemConstraintAttribute.hpp>
```

Collaboration diagram for multiscale::verification::UnaryScaleAndSubsystemConstraint-Attribute:



## Public Attributes

- [ComparatorAttribute comparator](#)
- [ScaleAndSubsystemAttribute scaleAndSubsystem](#)

### 6.231.1 Detailed Description

Class for representing a "unary" scale and subsystem constraint attribute.

Definition at line 15 of file UnaryScaleAndSubsystemConstraintAttribute.hpp.

### 6.231.2 Member Data Documentation

#### 6.231.2.1 ComparatorAttribute multiscale::verification::UnaryScaleAndSubsystemConstraintAttribute::comparator

The comparator

Definition at line 19 of file UnaryScaleAndSubsystemConstraintAttribute.hpp.

Referenced by multiscale::verification::ConstraintVisitor::operator()().

#### 6.231.2.2 ScaleAndSubsystemAttribute multiscale::verification::UnaryScaleAndSubsystemConstraintAttribute::scaleAndSubsystem

The considered scale and subsystem

Definition at line 20 of file UnaryScaleAndSubsystemConstraintAttribute.hpp.

Referenced by multiscale::verification::ConstraintVisitor::operator()().

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

- [UnaryScaleAndSubsystemConstraintAttribute.hpp](#)

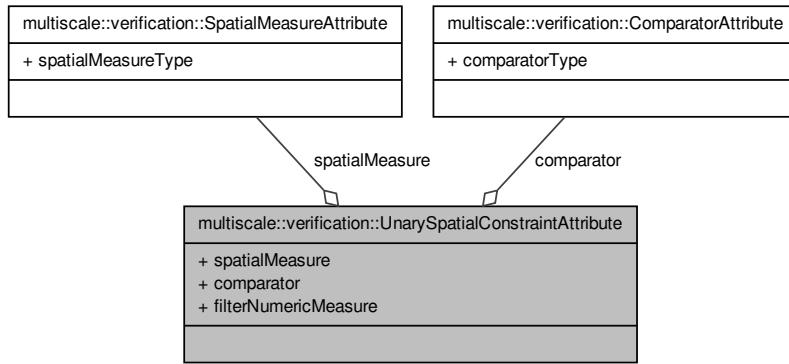
## 6.232 multiscale::verification::UnarySpatialConstraintAttribute - Class Reference

Class for representing a "unary" spatial constraint attribute.

```
#include <UnarySpatialConstraintAttribute.hpp>
```

## **6.232 multiscale::verification::UnarySpatialConstraintAttribute Class Reference**

Collaboration diagram for multiscale::verification::UnarySpatialConstraintAttribute:



### **Public Attributes**

- `SpatialMeasureAttribute spatialMeasure`
- `ComparatorAttribute comparator`
- `FilterNumericMeasureAttributeType filterNumericMeasure`

### **6.232.1 Detailed Description**

Class for representing a "unary" spatial constraint attribute.

Definition at line 16 of file `UnarySpatialConstraintAttribute.hpp`.

### **6.232.2 Member Data Documentation**

#### **6.232.2.1 ComparatorAttribute multiscale::verification::UnarySpatialConstraintAttribute::comparator**

The comparator

Definition at line 21 of file `UnarySpatialConstraintAttribute.hpp`.

Referenced by `multiscale::verification::ConstraintVisitor::operator()()`.

#### **6.232.2.2 FilterNumericMeasureAttributeType multiscale::verification::UnarySpatialConstraintAttribute::filterNumericMeasure**

The filter numeric measure

Definition at line 22 of file UnarySpatialConstraintAttribute.hpp.

Referenced by multiscale::verification::ConstraintVisitor::operator()().

### 6.232.2.3 **SpatialMeasureAttribute multiscale::verification::UnarySpatialConstraintAttribute::spatialMeasure**

The spatial measure

Definition at line 20 of file UnarySpatialConstraintAttribute.hpp.

Referenced by multiscale::verification::ConstraintVisitor::operator()().

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

- UnarySpatialConstraintAttribute.hpp

## 6.233 **multiscale::verification::UnaryStatisticalMeasureAttribute - Class Reference**

Class for representing a unary statistical measure attribute.

```
#include <UnaryStatisticalMeasureAttribute.hpp>
```

### Public Attributes

- [UnaryStatisticalMeasureType unaryStatisticalMeasureType](#)

#### 6.233.1 Detailed Description

Class for representing a unary statistical measure attribute.

Definition at line 40 of file UnaryStatisticalMeasureAttribute.hpp.

#### 6.233.2 Member Data Documentation

##### 6.233.2.1 **UnaryStatisticalMeasureType multiscale::verification::UnaryStatisticalMeasureAttribute::unaryStatisticalMeasureType**

The unary statistical measure type

Definition at line 44 of file UnaryStatisticalMeasureAttribute.hpp.

Referenced by multiscale::verification::TemporalNumericVisitor::operator()(), and multiscale::verification::NumericVisitor::operator()().

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

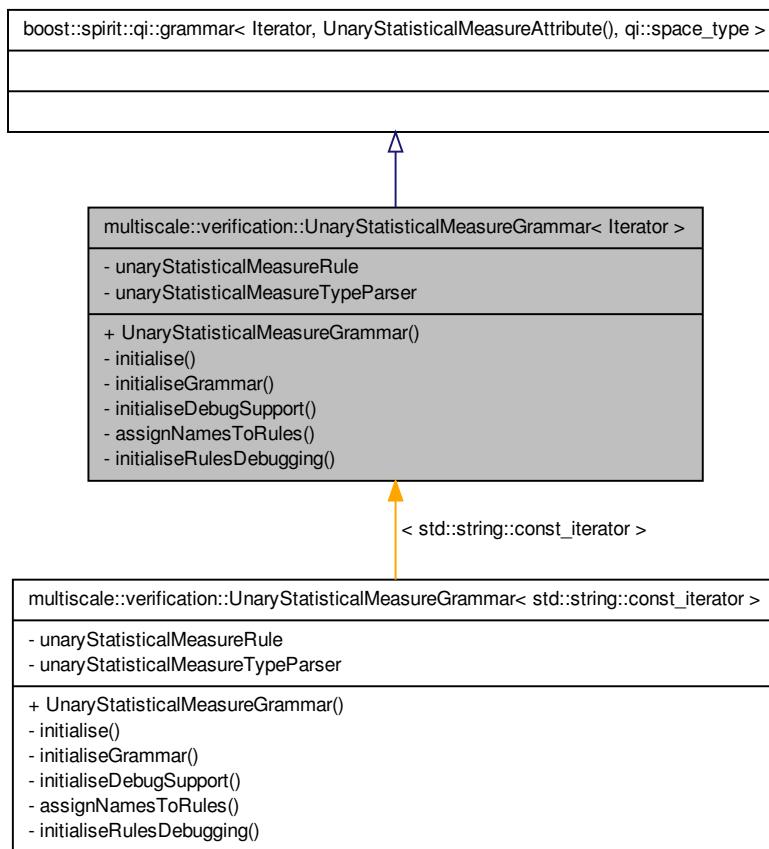
- UnaryStatisticalMeasureAttribute.hpp

## 6.234 multiscale::verification::UnaryStatisticalMeasureGrammar< Iterator > Class Template Reference

The grammar for parsing unary statistical measure statements.

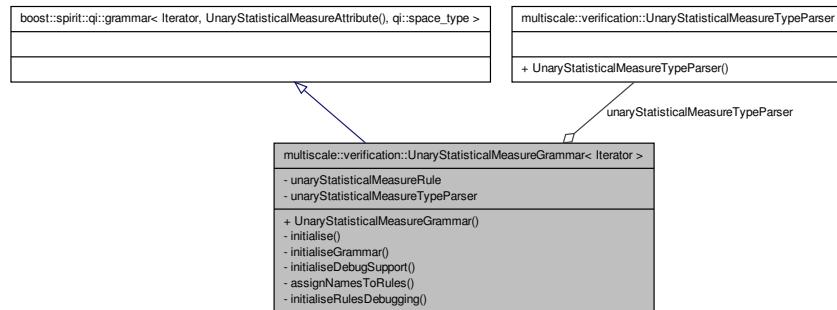
```
#include <UnaryStatisticalMeasureGrammar.hpp>
```

Inheritance diagram for multiscale::verification::UnaryStatisticalMeasureGrammar< - Iterator >:



Collaboration diagram for multiscale::verification::UnaryStatisticalMeasureGrammar< -

Iterator >:



## Public Member Functions

- [UnaryStatisticalMeasureGrammar \(\)](#)

## Private Member Functions

- void [initialise \(\)](#)  
*Initialisation function.*
- void [initialiseGrammar \(\)](#)  
*Initialise the grammar.*
- void [initialiseDebugSupport \(\)](#)  
*Initialise debug support.*
- void [assignNamesToRules \(\)](#)  
*Assign names to the rules.*
- void [initialiseRulesDebugging \(\)](#)  
*Initialise the debugging of rules.*

## Private Attributes

- `qi::rule< Iterator, UnaryStatisticalMeasureAttribute(), qi::space_type > unaryStatisticalMeasureRule`
- `UnaryStatisticalMeasureTypeParser unaryStatisticalMeasureTypeParser`

### 6.234.1 Detailed Description

```
template<typename Iterator>class multiscale::verification::UnaryStatisticalMeasureGrammar< Iterator >
```

The grammar for parsing unary statistical measure statements.

Definition at line 24 of file UnaryStatisticalMeasureGrammar.hpp.

### 6.234.2 Constructor & Destructor Documentation

6.234.2.1 template<typename Iterator> multiscale::verification::UnaryStatisticalMeasureGrammar< Iterator >::UnaryStatisticalMeasureGrammar( )

Definition at line 17 of file UnaryStatisticalMeasureGrammarDefinition.hpp.

References multiscale::verification::UnaryStatisticalMeasureGrammar< Iterator >::initialise().

### 6.234.3 Member Function Documentation

6.234.3.1 template<typename Iterator> void multiscale::verification::UnaryStatisticalMeasureGrammar< Iterator >::assignNamesToRules( )  
[private]

Assign names to the rules.

Definition at line 50 of file UnaryStatisticalMeasureGrammarDefinition.hpp.

6.234.3.2 template<typename Iterator> void multiscale::verification::UnaryStatisticalMeasureGrammar< Iterator >::initialise( )  
[private]

Initialisation function.

Definition at line 27 of file UnaryStatisticalMeasureGrammarDefinition.hpp.

Referenced by multiscale::verification::UnaryStatisticalMeasureGrammar< Iterator >::UnaryStatisticalMeasureGrammar().

6.234.3.3 template<typename Iterator> void multiscale::verification::UnaryStatisticalMeasureGrammar< Iterator >::initialiseDebugSupport( )  
[private]

Initialise debug support.

Definition at line 41 of file UnaryStatisticalMeasureGrammarDefinition.hpp.

6.234.3.4 template<typename Iterator> void multiscale::verification::UnaryStatisticalMeasureGrammar< Iterator >::initialiseGrammar( )  
[private]

Initialise the grammar.

Definition at line 34 of file UnaryStatisticalMeasureGrammarDefinition.hpp.

---

6.234.3.5 template<typename Iterator > void multiscale::verification::UnaryStatisticalMeasureGrammar< Iterator >::initialiseRulesDebugging( )  
[private]

Initialise the debugging of rules.

Definition at line 56 of file UnaryStatisticalMeasureGrammarDefinition.hpp.

#### 6.234.4 Member Data Documentation

6.234.4.1 template<typename Iterator> qi::rule<Iterator, UnaryStatisticalMeasureAttribute(), qi::space\_type> multiscale::verification::UnaryStatisticalMeasureGrammar< Iterator >::unaryStatisticalMeasureRule [private]

The rule for parsing a unary statistical measure

Definition at line 30 of file UnaryStatisticalMeasureGrammar.hpp.

6.234.4.2 template<typename Iterator> UnaryStatisticalMeasureTypeParser multiscale::verification::UnaryStatisticalMeasureGrammar< Iterator >::unaryStatisticalMeasureTypeParser [private]

The unary statistical measure type parser

Definition at line 36 of file UnaryStatisticalMeasureGrammar.hpp.

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

- UnaryStatisticalMeasureGrammar.hpp
- UnaryStatisticalMeasureGrammarDefinition.hpp

### 6.235 multiscale::verification::UnaryStatisticalMeasureTypeParser Struct Reference

Symbol table and parser for the unary statistical measure type.

```
#include <SymbolTables.hpp>
```

#### Public Member Functions

- [UnaryStatisticalMeasureTypeParser \(\)](#)

#### 6.235.1 Detailed Description

Symbol table and parser for the unary statistical measure type.

## [6.236 multiscale::verification::UnaryStatisticalNumericAttribute Class Reference](#)

Definition at line 214 of file SymbolTables.hpp.

### 6.235.2 Constructor & Destructor Documentation

**6.235.2.1 multiscale::verification::UnaryStatisticalMeasureTypeParser::UnaryStatisticalMeasureTypeParser ( )**  
[inline]

Definition at line 217 of file SymbolTables.hpp.

References multiscale::verification::Count, multiscale::verification::Harmean, multiscale::verification::Max, multiscale::verification::Min, multiscale::verification::Product, multiscale::verification::Stdev, and multiscale::verification::Var.

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

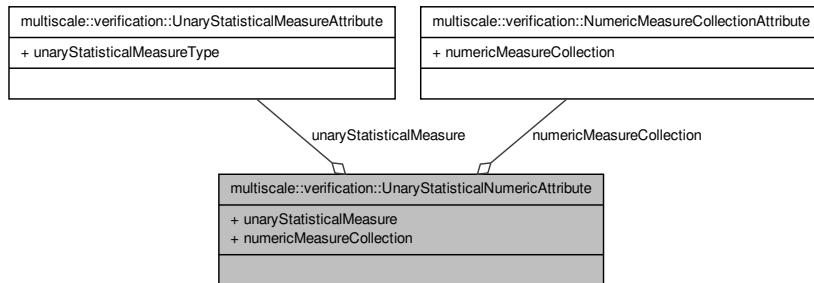
- SymbolTables.hpp

## [6.236 multiscale::verification::UnaryStatisticalNumericAttribute - Class Reference](#)

Class for representing a unary statistical numeric attribute.

```
#include <UnaryStatisticalNumericAttribute.hpp>
```

Collaboration diagram for multiscale::verification::UnaryStatisticalNumericAttribute:



### Public Attributes

- `UnaryStatisticalMeasureAttribute unaryStatisticalMeasure`
- `NumericMeasureCollectionAttribute numericMeasureCollection`

### 6.236.1 Detailed Description

Class for representing a unary statistical numeric attribute.

Definition at line 15 of file UnaryStatisticalNumericAttribute.hpp.

### 6.236.2 Member Data Documentation

#### 6.236.2.1 NumericMeasureCollectionAttribute multiscale::verification::UnaryStatisticalNumericAttribute::numericMeasureCollection

The considered numeric measure collection

Definition at line 20 of file UnaryStatisticalNumericAttribute.hpp.

Referenced by multiscale::verification::TemporalNumericVisitor::operator()().

#### 6.236.2.2 UnaryStatisticalMeasureAttribute multiscale::verification::UnaryStatisticalNumericAttribute::unaryStatisticalMeasure

The unary statistical measure

Definition at line 19 of file UnaryStatisticalNumericAttribute.hpp.

Referenced by multiscale::verification::TemporalNumericVisitor::operator()().

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

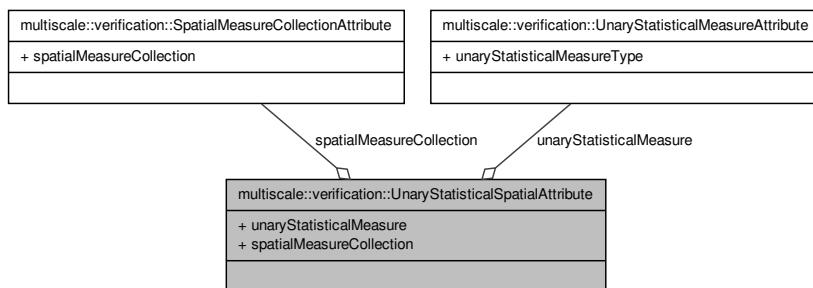
- UnaryStatisticalNumericAttribute.hpp

## 6.237 multiscale::verification::UnaryStatisticalSpatialAttribute - Class Reference

Class for representing a unary statistical spatial attribute.

```
#include <UnaryStatisticalSpatialAttribute.hpp>
```

Collaboration diagram for multiscale::verification::UnaryStatisticalSpatialAttribute:



## Public Attributes

- `UnaryStatisticalMeasureAttribute unaryStatisticalMeasure`
- `SpatialMeasureCollectionAttribute spatialMeasureCollection`

### 6.237.1 Detailed Description

Class for representing a unary statistical spatial attribute.

Definition at line 15 of file `UnaryStatisticalSpatialAttribute.hpp`.

### 6.237.2 Member Data Documentation

#### 6.237.2.1 SpatialMeasureCollectionAttribute multiscale::verification::UnaryStatisticalSpatialAttribute::spatialMeasureCollection

The considered spatial measure collection

Definition at line 20 of file `UnaryStatisticalSpatialAttribute.hpp`.

Referenced by `multiscale::verification::NumericVisitor::operator()()`.

#### 6.237.2.2 UnaryStatisticalMeasureAttribute multiscale::verification::UnaryStatisticalSpatialAttribute::unaryStatisticalMeasure

The unary statistical measure

Definition at line 19 of file `UnaryStatisticalSpatialAttribute.hpp`.

Referenced by `multiscale::verification::NumericVisitor::operator()()`.

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

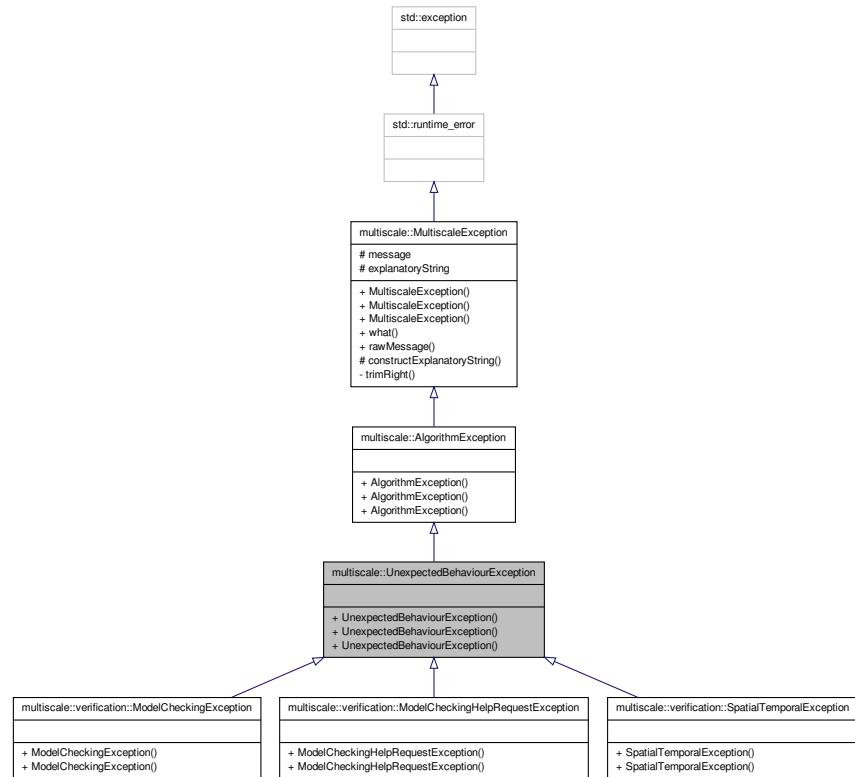
- UnaryStatisticalSpatialAttribute.hpp

## 6.238 multiscale::UnexpectedBehaviourException Class Reference

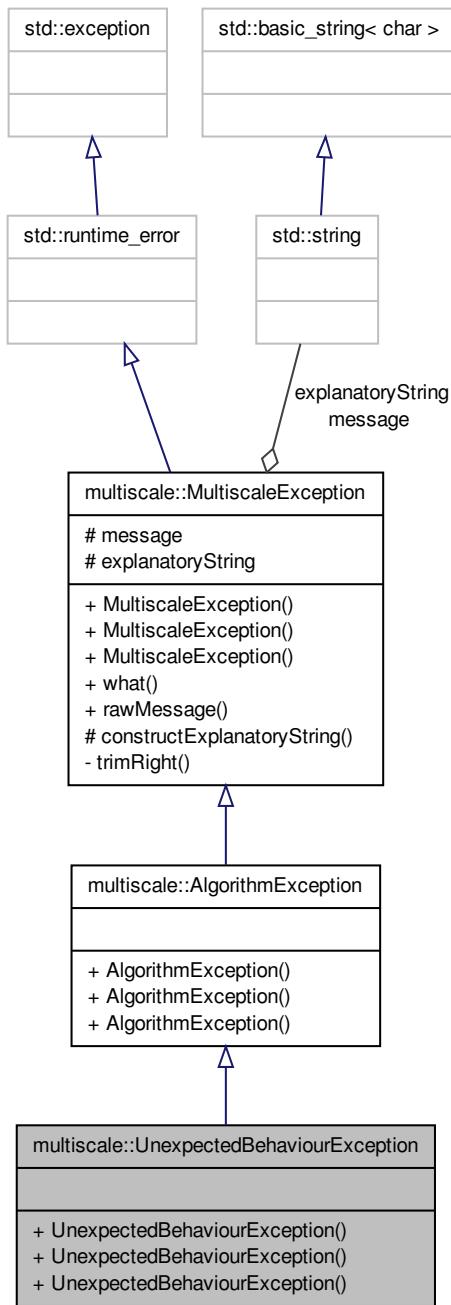
Class for representing unexpected behaviour exceptions.

```
#include <UnexpectedBehaviourException.hpp>
```

Inheritance diagram for multiscale::UnexpectedBehaviourException:



Collaboration diagram for multiscale::UnexpectedBehaviourException:



## Public Member Functions

- [UnexpectedBehaviourException \(\)](#)
- [UnexpectedBehaviourException \(const std::string &file, int line, const std::string &msg\)](#)
- [UnexpectedBehaviourException \(const std::string &file, int line, const char \\*msg\)](#)

### 6.238.1 Detailed Description

Class for representing unexpected behaviour exceptions.

Definition at line 12 of file UnexpectedBehaviourException.hpp.

### 6.238.2 Constructor & Destructor Documentation

#### 6.238.2.1 multiscale::UnexpectedBehaviourException::UnexpectedBehaviourException( ) [inline]

Definition at line 16 of file UnexpectedBehaviourException.hpp.

#### 6.238.2.2 multiscale::UnexpectedBehaviourException::UnexpectedBehaviourException( const std::string & file, int line, const std::string & msg ) [inline, explicit]

Definition at line 18 of file UnexpectedBehaviourException.hpp.

#### 6.238.2.3 multiscale::UnexpectedBehaviourException::UnexpectedBehaviourException( const std::string & file, int line, const char \* msg ) [inline, explicit]

Definition at line 23 of file UnexpectedBehaviourException.hpp.

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

- [UnexpectedBehaviourException.hpp](#)

## 6.239 multiscale::verification::UnexpectedErrorHandler - Struct Reference

Structure for defining the error handler for unexpected token errors.

```
#include <UnexpectedErrorHandler.hpp>
```

## Classes

- struct [result](#)  
*Structure for specifying the type of the result.*

## Public Member Functions

- template<typename Iterator>  
void [operator\(\)](#) (qi::info const &expectedToken, Iterator errorPosition, Iterator last)  
const  
*Overloaded operator.*

## Private Member Functions

- std::string [getExpectedTokenAsString](#) (qi::info const &expectedToken) const  
*Convert the expected token to a string.*

### 6.239.1 Detailed Description

Structure for defining the error handler for unexpected token errors.

Definition at line 17 of file UnexpectedErrorHandler.hpp.

### 6.239.2 Member Function Documentation

#### 6.239.2.1 std::string multiscale::verification::UnexpectedErrorHandler::get- ExpectedTokenAsString ( qi::info const & *expectedToken* ) const [inline, private]

Convert the expected token to a string.

Convert the expected token to a string and remove enclosing quotes

##### Parameters

|                                  |                                   |
|----------------------------------|-----------------------------------|
| <code>expected-<br/>Token</code> | The expected token (not a string) |
|----------------------------------|-----------------------------------|

Definition at line 46 of file UnexpectedErrorHandler.hpp.

Referenced by [operator\(\)\(\)](#).

#### 6.239.2.2 template<typename Iterator> void multiscale::verification::UnexpectedErrorHandler::operator() ( qi::info const & *expectedToken*, Iterator *errorPosition*, Iterator *last* ) const [inline]

Overloaded operator.

**Parameters**

|                            |                                           |
|----------------------------|-------------------------------------------|
| <i>expected-<br/>Token</i> | The expected token                        |
| <i>errorPosition</i>       | Iterator pointing to the error position   |
| <i>last</i>                | Iterator pointing to the end of the query |

Definition at line 32 of file UnexpectedTokenErrorHandler.hpp.

References getExpectedTokenAsString().

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

- UnexpectedTokenErrorHandler.hpp

## 6.240 multiscale::verification::TimeseriesComponentEvaluator::- UniformHomogeneousComponentEvaluator< Relation > Class Template Reference

```
#include <TimeseriesComponentEvaluator.hpp>
```

### Public Member Functions

- bool `isStartIndex` (std::size\_t index, std::size\_t nrOfValues, const std::vector< double > &values, const Relation &relation) const  
*Check if the given index is a uniform homogeneous component start index.*
- std::size\_t `computeEndIndex` (std::size\_t startIndex, std::size\_t nrOfValues, const std::vector< double > &values, const Relation &relation) const  
*Compute the uniform homogeneous component end index considering the given start index.*

### Private Member Functions

- bool `hasValidSuccessors` (std::size\_t index, std::size\_t nrOfValues, const std::vector< double > &values, const Relation &relation) const  
*Check if a valid uniform homogeneous subcomponent starts from the given index.*

### 6.240.1 Detailed Description

```
template<typename Relation>class multiscale::verification::TimeseriesComponentEvaluator::-  
UniformHomogeneousComponentEvaluator< Relation >
```

Definition at line 210 of file TimeseriesComponentEvaluator.hpp.

**6.240.2 Member Function Documentation**

```
6.240.2.1 template<typename Relation> std::size_t multiscale::verification::TimeseriesComponentEvaluator::UniformHomogeneousComponentEvaluator< Relation >::computeEndIndex ( std::size_t startIndex, std::size_t nrOfValues, const std::vector< double > & values, const Relation & relation ) const [inline]
```

Compute the uniform homogeneous component end index considering the given start index.

A value located at index  $i$  in the collection,  $1 < i < (\text{values.size()} - 1)$ , is a nonuniform homogeneous component ending index if and only if  $\text{relation}(\text{values}[i - 2], \text{values}[i - 1])$ ,  $\text{relation}(\text{values}[i - 1], \text{values}[i])$ ,  $(\text{values}[i - 2] - \text{values}[i - 1]) = \text{values}[i - 1] - \text{values}[i]$  and  $\text{!relation}(\text{values}[i], \text{values}[i + 1])$  or  $(\text{values}[i - 1] - \text{values}[i]) != (\text{values}[i] - \text{values}[i + 1])$ , respectively  $\text{relation}(\text{values}[i - 2], \text{values}[i - 1])$ ,  $\text{relation}(\text{values}[i - 1], \text{values}[i])$  and  $(\text{values}[i - 2] - \text{values}[i - 1]) = (\text{values}[i - 1] - \text{values}[i])$  if  $i = (\text{values.size()} - 1)$ .

**Parameters**

|                   |                                 |
|-------------------|---------------------------------|
| <i>startIndex</i> | The given start index           |
| <i>nrOfValues</i> | The number of considered values |
| <i>values</i>     | The collection of values        |
| <i>relation</i>   | The considered relation         |

Definition at line 265 of file TimeseriesComponentEvaluator.hpp.

References multiscale::verification::TimeseriesComponentEvaluator::UniformHomogeneousComponentEvaluator< Relation >::isValidSuccessors(), and multiscale::verification::TimeseriesComponentEvaluator::UniformHomogeneousComponentEvaluator< Relation >::isStartIndex().

```
6.240.2.2 template<typename Relation> bool multiscale::verification::TimeseriesComponentEvaluator::UniformHomogeneousComponentEvaluator< Relation >::isValidSuccessors ( std::size_t index, std::size_t nrOfValues, const std::vector< double > & values, const Relation & relation ) const [inline, private]
```

Check if a valid uniform homogeneous subcomponent starts from the given index.

**Parameters**

|                   |                                 |
|-------------------|---------------------------------|
| <i>index</i>      | The given index                 |
| <i>nrOfValues</i> | The number of considered values |
| <i>values</i>     | The collection of values        |
| <i>relation</i>   | The considered relation         |

Definition at line 294 of file TimeseriesComponentEvaluator.hpp.

References multiscale::Numeric::almostEqual().

Referenced by multiscale::verification::TimeseriesComponentEvaluator::UniformHomogeneousComponentEvaluator< Relation >::computeEndIndex(), and multiscale::verification::TimeseriesComponentEvaluator::UniformHomogeneousComponentEvaluator< Relation >::isStartIndex().

```
6.240.2.3 template<typename Relation> bool multiscale::verification::Timeseries-
ComponentEvaluator::UniformHomogeneousComponentEvaluator<
Relation>::isStartIndex ( std::size_t index, std::size_t nrOfValues, const
std::vector< double > & values, const Relation & relation ) const [inline]
```

Check if the given index is a uniform homogeneous component start index.

A value located at index  $i$  in the collection,  $0 < i < (\text{values.size()} - 2)$ , is a uniform homogeneous component starting index if and only if  $(\text{relation}(\text{values}[i - 1], \text{values}[i]))$  or  $(\text{values}[i - 2] - \text{values}[i - 1]) != (\text{values}[i - 1] - \text{values}[i])$ ,  $\text{relation}(\text{values}[i], \text{values}[i + 1])$ ,  $\text{relation}(\text{values}[i + 1], \text{values}[i + 2])$  and  $(\text{values}[i] - \text{values}[i + 1]) = (\text{values}[i + 1] - \text{values}[i + 2])$ , respectively  $\text{relation}(\text{values}[i], \text{values}[i + 1])$ ,  $\text{relation}(\text{values}[i + 1], \text{values}[i + 2])$ , and  $(\text{values}[i] - \text{values}[i + 1]) = (\text{values}[i + 1] - \text{values}[i + 2])$  if  $i = 0$ .

#### Parameters

|                   |                                 |
|-------------------|---------------------------------|
| <i>index</i>      | The given index                 |
| <i>nrOfValues</i> | The number of considered values |
| <i>values</i>     | The collection of values        |
| <i>relation</i>   | The considered relation         |

Definition at line 228 of file TimeseriesComponentEvaluator.hpp.

References multiscale::verification::TimeseriesComponentEvaluator::UniformHomogeneousComponentEvaluator< Relation >::hasValidSuccessors().

Referenced by multiscale::verification::TimeseriesComponentEvaluator::UniformHomogeneousComponentEvaluator< Relation >::computeEndIndex().

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

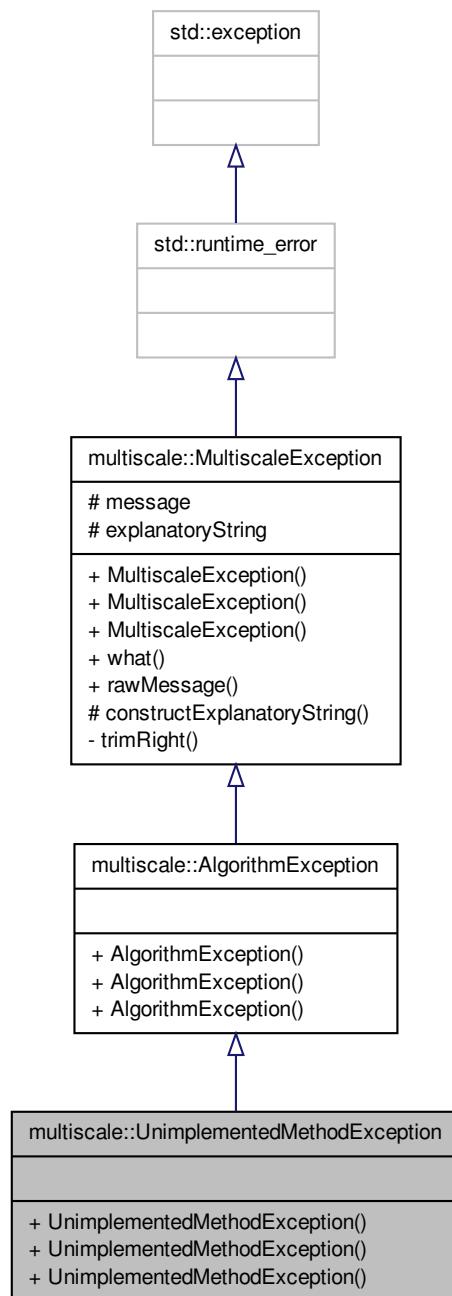
- TimeseriesComponentEvaluator.hpp

## 6.241 multiscale::UnimplementedMethodException Class Reference

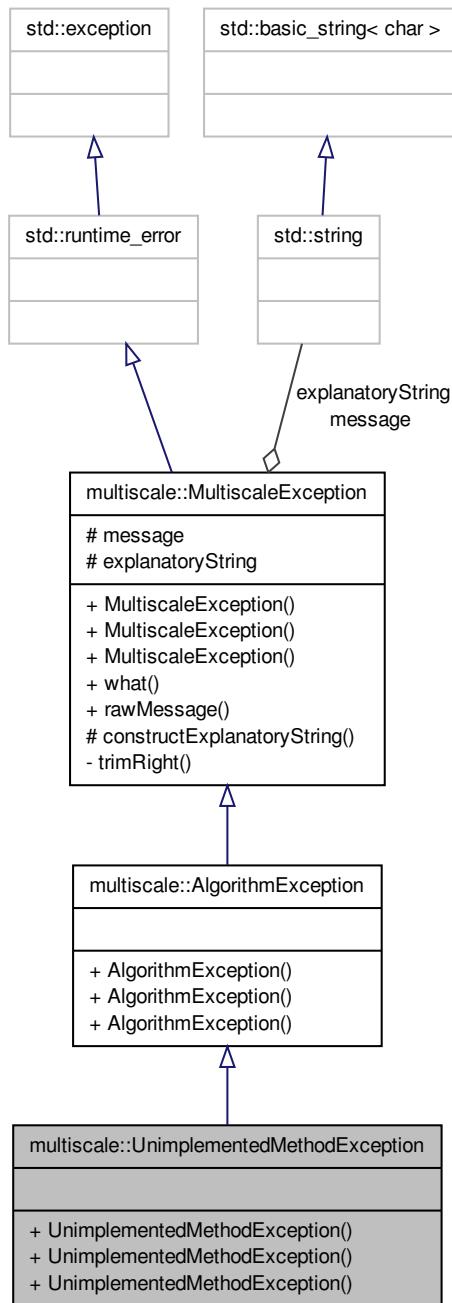
Class for representing unimplemented method exceptions.

```
#include <UnimplementedMethodException.hpp>
```

Inheritance diagram for multiscale::UnimplementedMethodException:



Collaboration diagram for multiscale::UnimplementedMethodException:



## Public Member Functions

- [UnimplementedMethodException \(\)](#)
- [UnimplementedMethodException \(const std::string &file, int line, const std::string &msg\)](#)
- [UnimplementedMethodException \(const std::string &file, int line, const char \\*msg\)](#)

### 6.241.1 Detailed Description

Class for representing unimplemented method exceptions.

Definition at line 12 of file UnimplementedMethodException.hpp.

### 6.241.2 Constructor & Destructor Documentation

#### 6.241.2.1 **multiscale::UnimplementedMethodException::UnimplementedMethodException ( ) [inline]**

Definition at line 16 of file UnimplementedMethodException.hpp.

#### 6.241.2.2 **multiscale::UnimplementedMethodException::UnimplementedMethodException ( const std::string & file, int line, const std::string & msg ) [inline, explicit]**

Definition at line 18 of file UnimplementedMethodException.hpp.

#### 6.241.2.3 **multiscale::UnimplementedMethodException::UnimplementedMethodException ( const std::string & file, int line, const char \* msg ) [inline, explicit]**

Definition at line 23 of file UnimplementedMethodException.hpp.

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

- [UnimplementedMethodException.hpp](#)

## 6.242 multiscale::verification::UntilLogicPropertyAttribute Class - Reference

Class for representing an "until" logic property attribute.

```
#include <UntilLogicPropertyAttribute.hpp>
```

## Public Attributes

- unsigned long [startTimepoint](#)
- unsigned long [endTimepoint](#)
- [LogicPropertyAttributeType logicProperty](#)

### 6.242.1 Detailed Description

Class for representing an "until" logic property attribute.

Definition at line 14 of file UntilLogicPropertyAttribute.hpp.

### 6.242.2 Member Data Documentation

#### 6.242.2.1 unsigned long multiscale::verification::UntilLogicPropertyAttribute::end-Timepoint

The considered end timepoint

Definition at line 19 of file UntilLogicPropertyAttribute.hpp.

#### 6.242.2.2 LogicPropertyAttributeType multiscale::verification::UntilLogic-PropertyAttribute::logicProperty

The logic property following the "until" operator

Definition at line 20 of file UntilLogicPropertyAttribute.hpp.

Referenced by multiscale::verification::LogicPropertyVisitor::evaluateTemporalLogic-PropertyWithStartAndEndTimepoints().

#### 6.242.2.3 unsigned long multiscale::verification::UntilLogicPropertyAttribute::start-Timepoint

The considered start timepoint

Definition at line 18 of file UntilLogicPropertyAttribute.hpp.

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

- UntilLogicPropertyAttribute.hpp

### 6.243 multiscale::UserDefinedTypeName< T > Class Template - Reference

Class for representing a user defined type name.

```
#include <UserDefinedTypeName.hpp>
```

## **6.244 multiscale::XmlValidator::XmlValidationErrorHandler Class Reference 1255**

---

### **Static Public Member Functions**

- static std::string [name \(\)](#)

*Retrieve the name of the given type.*

#### **6.243.1 Detailed Description**

```
template<typename T>class multiscale::UserDefinedTypeName< T >
```

Class for representing a user defined type name.

Definition at line 12 of file UserDefinedTypeName.hpp.

#### **6.243.2 Member Function Documentation**

```
6.243.2.1 template<typename T > static std::string multiscale::UserDefinedTypeName<  
T >::name( ) [inline, static]
```

Retrieve the name of the given type.

Definition at line 17 of file UserDefinedTypeName.hpp.

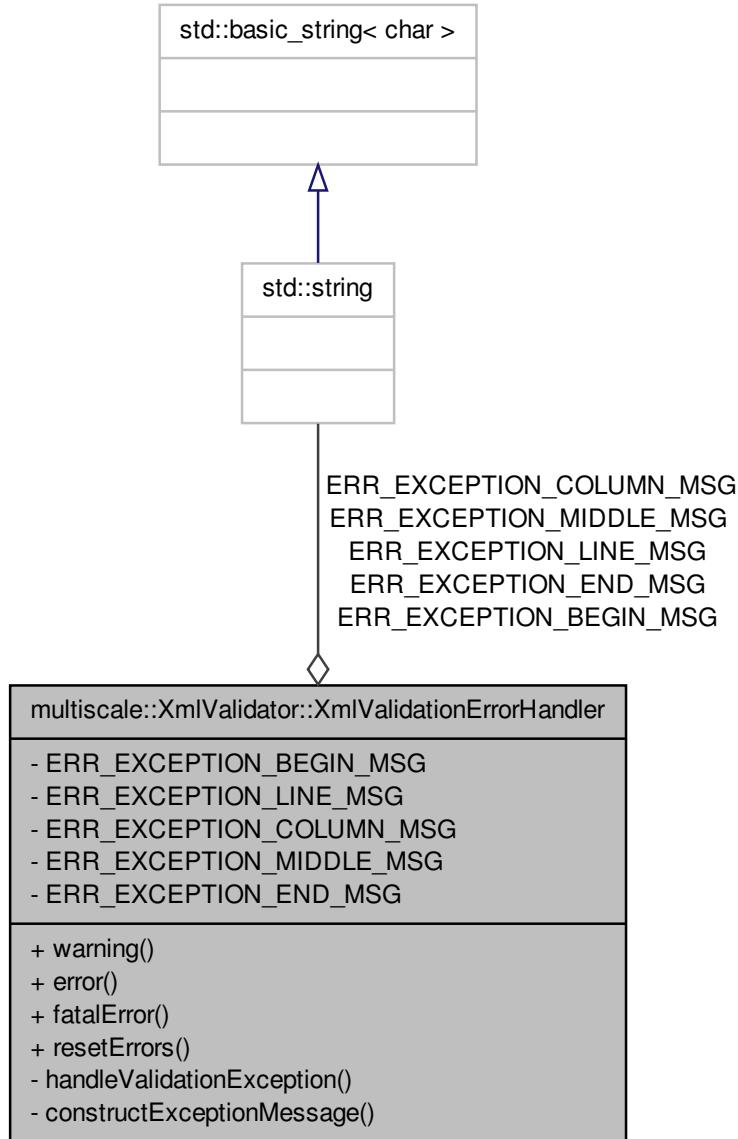
The documentation for this class was generated from the following file:

- UserDefinedTypeName.hpp

## **6.244 multiscale::XmlValidator::XmlValidationErrorHandler Class - Reference**

Class used for handling errors during the xml file validation process.

Collaboration diagram for multiscale::XmlValidator::XmlValidationErrorHandler:



### Public Member Functions

- void `warning` (const SAXParseException &ex) override

## **6.244 multiscale::XmlValidator::XmlValidationErrorHandler Class Reference 1257**

---

- *Handle warning messages.*
  - void **error** (const SAXParseException &ex) override
    - *Handle recoverable error messages.*
  - void **fatalError** (const SAXParseException &ex) override
    - *Handle non-recoverable error messages.*
  - void **resetErrors** () override
    - *Reinitialise the error handler.*

### **Private Member Functions**

- void **handleValidationException** (const SAXParseException &ex)
  - *Handle the exception thrown during the validation process.*
- std::string **constructExceptionMessage** (const SAXParseException &ex)
  - *Construct the exception message for the given exception.*

### **Static Private Attributes**

- static const std::string **ERR\_EXCEPTION\_BEGIN\_MSG** = "An **error** occurred at "
  - static const std::string **ERR\_EXCEPTION\_LINE\_MSG** = "line "
  - static const std::string **ERR\_EXCEPTION\_COLUMN\_MSG** = ", column "
  - static const std::string **ERR\_EXCEPTION\_MIDDLE\_MSG** = " and the **error** message is \""
  - static const std::string **ERR\_EXCEPTION\_END\_MSG** = "\.".

#### **6.244.1 Detailed Description**

Class used for handling errors during the xml file validation process.

Definition at line 111 of file XmlValidator.hpp.

#### **6.244.2 Member Function Documentation**

##### **6.244.2.1 std::string XmlValidator::XmlValidationErrorHandler- ::constructExceptionMessage ( const SAXParseException & ex ) [private]**

Construct the exception message for the given exception.

###### **Parameters**

|           |                                                    |
|-----------|----------------------------------------------------|
| <b>ex</b> | The exception thrown during the validation process |
|-----------|----------------------------------------------------|

Definition at line 108 of file XmlValidator.cpp.

References multiscale::StringManipulator::toString().

**6.244.2.2 void XmlValidator::XmlValidationErrorHandler::error ( const SAXParseException & ex ) [override]**

Handle recoverable error messages.

**Parameters**

|    |                                                    |
|----|----------------------------------------------------|
| ex | The exception thrown during the validation process |
|----|----------------------------------------------------|

Definition at line 92 of file XmlValidator.cpp.

**6.244.2.3 void XmlValidator::XmlValidationErrorHandler::fatalError ( const SAXParseException & ex ) [override]**

Handle non-recoverable error messages.

**Parameters**

|    |                                                    |
|----|----------------------------------------------------|
| ex | The exception thrown during the validation process |
|----|----------------------------------------------------|

Definition at line 96 of file XmlValidator.cpp.

**6.244.2.4 void XmlValidator::XmlValidationErrorHandler::handle-ValidationException ( const SAXParseException & ex ) [private]**

Handle the exception thrown during the validation process.

**Parameters**

|    |                                                    |
|----|----------------------------------------------------|
| ex | The exception thrown during the validation process |
|----|----------------------------------------------------|

Definition at line 102 of file XmlValidator.cpp.

Referenced by warning().

**6.244.2.5 void XmlValidator::XmlValidationErrorHandler::resetErrors ( ) [override]**

Reinitialise the error handler.

Definition at line 100 of file XmlValidator.cpp.

## **6.244 multiscale::XmlValidator::XmlValidationErrorHandler Class Reference 1259**

---

**6.244.2.6 void XmlValidator::XmlValidationErrorHandler::warning ( const SAXParseException & ex ) [override]**

Handle warning messages.

### Parameters

|           |                                                    |
|-----------|----------------------------------------------------|
| <b>ex</b> | The exception thrown during the validation process |
|-----------|----------------------------------------------------|

Definition at line 88 of file XmlValidator.cpp.

References handleValidationException().

### **6.244.3 Member Data Documentation**

**6.244.3.1 const std::string XmlValidator::XmlValidationErrorHandler::ERR\_EXCEPTION\_BEGIN\_MSG = "An error occurred at " [static, private]**

Definition at line 151 of file XmlValidator.hpp.

**6.244.3.2 const std::string XmlValidator::XmlValidationErrorHandler::ERR\_EXCEPTION\_COLUMN\_MSG = ", column " [static, private]**

Definition at line 154 of file XmlValidator.hpp.

**6.244.3.3 const std::string XmlValidator::XmlValidationErrorHandler::ERR\_EXCEPTION\_END\_MSG = "\n" [static, private]**

Definition at line 157 of file XmlValidator.hpp.

**6.244.3.4 const std::string XmlValidator::XmlValidationErrorHandler::ERR\_EXCEPTION\_LINE\_MSG = "line " [static, private]**

Definition at line 153 of file XmlValidator.hpp.

**6.244.3.5 const std::string XmlValidator::XmlValidationErrorHandler::ERR\_EXCEPTION\_MIDDLE\_MSG = " and the error message is \n" [static, private]**

Definition at line 155 of file XmlValidator.hpp.

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

- XmlValidator.hpp

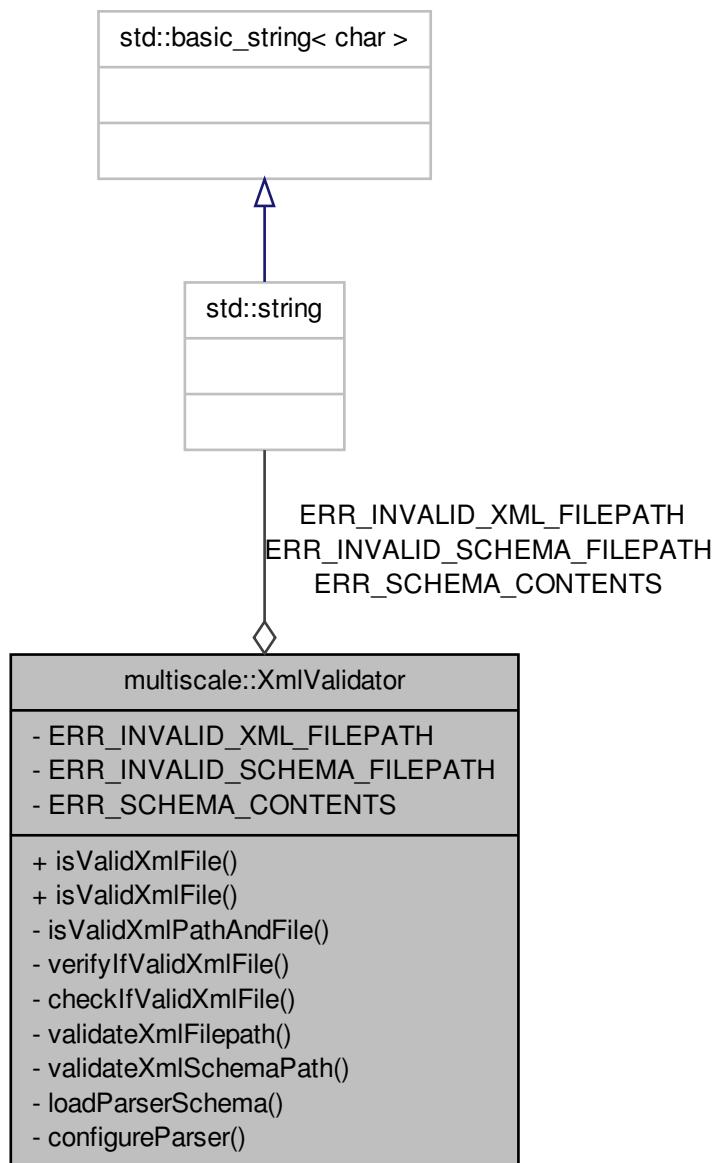
- XmlValidator.cpp

## 6.245 multiscale::XmlValidator Class Reference

Class used to validate xml files.

```
#include <XmlValidator.hpp>
```

Collaboration diagram for multiscale::XmlValidator:



## Classes

- class [XmlValidationErrorHandler](#)

*Class used for handling errors during the xml file validation process.*

## Static Public Member Functions

- static bool [isValidXmlFile](#) (const std::string &xmlFilepath, const std::string &xmlSchemaPath)  
*Check if the given xml file is valid considering the provided xml schema (xsd file)*
- static bool [isValidXmlFile](#) (const std::string &xmlFilepath, const std::string &xmlSchemaPath, std::string &xmlErrorMessage)  
*Check if the given xml file is valid considering the provided xml schema (xsd file)*

## Static Private Member Functions

- static bool [isValidXmlPathAndFile](#) (const std::string &xmlFilepath, const std::string &xmlSchemaPath, std::string &xmlErrorMessage)  
*Check if the given xml file is valid considering the provided xml schema (xsd file)*
- static bool [verifyIfValidXmlFile](#) (const std::string &xmlFilepath, const std::string &xmlSchemaPath, std::string &xmlErrorMessage)  
*Check if the given xml file is valid considering the provided xml schema (xsd file)*
- static bool [checkIfValidXmlFile](#) (const std::string &xmlFilepath, const std::string &xmlSchemaPath)  
*Check if the given xml file is valid considering the provided xml schema (xsd file)*
- static void [validateXmlFilepath](#) (const std::string &xmlFilepath)  
*Check if the provided xml file path is valid.*
- static void [validateXmlSchemaPath](#) (const std::string &xmlSchemaPath)  
*Check if the provided xml schema file path is valid.*
- static void [loadParserSchema](#) (const std::string &xmlSchemaPath, XercesDOMParser &parser)  
*Load the xml schema using the given parser.*
- static void [configureParser](#) (XercesDOMParser &parser)  
*Configure the given parser.*

## Static Private Attributes

- static const std::string [ERR\\_INVALID\\_XML\\_FILEPATH](#) = "The provided xml file path is invalid. Please change."
- static const std::string [ERR\\_INVALID\\_SCHEMA\\_FILEPATH](#) = "The provided xml schema file path is invalid. Please change."
- static const std::string [ERR\\_SCHEMA\\_CONTENTS](#) = "The provided xml schema is invalid. Please verify the xml schema contents."

### 6.245.1 Detailed Description

Class used to validate xml files.

Definition at line 18 of file XmlValidator.hpp.

### 6.245.2 Member Function Documentation

**6.245.2.1 bool XmlValidator::checkIfValidXmlFile ( const std::string & *xmlFilepath*, const std::string & *xmlSchemaPath* ) [static, private]**

Check if the given xml file is valid considering the provided xml schema (xsd file)

The validation is performed using the Xerces C++ library.

#### Parameters

|                       |                                 |
|-----------------------|---------------------------------|
| <i>xmlFilepath</i>    | The path to the xml file        |
| <i>xmlSchema-Path</i> | The path to the xml schema file |

Definition at line 48 of file XmlValidator.cpp.

References configureParser(), and loadParserSchema().

Referenced by verifyIfValidXmlFile().

**6.245.2.2 void XmlValidator::configureParser ( XercesDOMParser & *parser* ) [static, private]**

Configure the given parser.

#### Parameters

|               |                          |
|---------------|--------------------------|
| <i>parser</i> | The given xml DOM parser |
|---------------|--------------------------|

Definition at line 77 of file XmlValidator.cpp.

Referenced by checkIfValidXmlFile().

**6.245.2.3 bool XmlValidator::isValidXmlFile ( const std::string & *xmlFilepath*, const std::string & *xmlSchemaPath* ) [static]**

Check if the given xml file is valid considering the provided xml schema (xsd file)

The validation is performed using the Xerces C++ library.

#### Parameters

|                       |                                 |
|-----------------------|---------------------------------|
| <i>xmlFilepath</i>    | The path to the xml file        |
| <i>xmlSchema-Path</i> | The path to the xml schema file |

Definition at line 12 of file XmlValidator.cpp.

Referenced by multiscale::verification::SpatialTemporalDataReader::isValidInputFile(), and multiscale::verification::MultiscaleArchitectureGraph::readFromValidFile().

**6.245.2.4 bool XmlValidator::isValidXmlFile ( const std::string & *xmlFilepath*, const std::string & *xmlSchemaPath*, std::string & *xmlErrorMessage* ) [static]**

Check if the given xml file is valid considering the provided xml schema (xsd file)

The validation is performed using the Xerces C++ library.

In case the xml file is not valid the error message is retrieved in *xmlErrorMessage*.

#### Parameters

|                          |                                                            |
|--------------------------|------------------------------------------------------------|
| <i>xmlFilepath</i>       | The path to the xml file                                   |
| <i>xmlSchema- Path</i>   | The path to the xml schema file                            |
| <i>xmlError- Message</i> | The error message explaining why the xml file is not valid |

Definition at line 18 of file XmlValidator.cpp.

References isValidXmlPathAndFile().

**6.245.2.5 bool XmlValidator::isValidXmlPathAndFile ( const std::string & *xmlFilepath*, const std::string & *xmlSchemaPath*, std::string & *xmlErrorMessage* ) [static, private]**

Check if the given xml file is valid considering the provided xml schema (xsd file)

The validation is performed using the Xerces C++ library.

In case the xml file is not valid the error message is retrieved in *xmlErrorMessage*.

#### Parameters

|                          |                                                            |
|--------------------------|------------------------------------------------------------|
| <i>xmlFilepath</i>       | The path to the xml file                                   |
| <i>xmlSchema- Path</i>   | The path to the xml schema file                            |
| <i>xmlError- Message</i> | The error message explaining why the xml file is not valid |

Definition at line 29 of file XmlValidator.cpp.

References validateXmlFilepath(), validateXmlSchemaPath(), and verifyIfValidXmlFile().

Referenced by isValidXmlFile().

**6.245.2.6 void XmlValidator::loadParserSchema ( const std::string & *xmlSchemaPath*, XercesDOMParser & *parser* ) [static, private]**

Load the xml schema using the given parser.

**Parameters**

|                       |                                 |
|-----------------------|---------------------------------|
| <i>parser</i>         | The given xml DOM parser        |
| <i>xmlSchema-Path</i> | The file path to the xml schema |

Definition at line 71 of file XmlValidator.cpp.

References ERR\_SCHEMA\_CONTENTS.

Referenced by checkIfValidXmlFile().

**6.245.2.7 void XmlValidator::validateXmlFilepath ( const std::string & *xmlFilepath* ) [static, private]**

Check if the provided xml file path is valid.

**Parameters**

|                    |                          |
|--------------------|--------------------------|
| <i>xmlFilepath</i> | The path to the xml file |
|--------------------|--------------------------|

Definition at line 59 of file XmlValidator.cpp.

References ERR\_INVALID\_XML\_FILEPATH, and multiscale::Filesystem::isValidFilePath().

Referenced by isValidXmlPathAndFile().

**6.245.2.8 void XmlValidator::validateXmlSchemaPath ( const std::string & *xmlSchemaPath* ) [static, private]**

Check if the provided xml schema file path is valid.

**Parameters**

|                       |                            |
|-----------------------|----------------------------|
| <i>xmlSchema-Path</i> | The path to the xml schema |
|-----------------------|----------------------------|

Definition at line 65 of file XmlValidator.cpp.

References ERR\_INVALID\_SCHEMA\_FILEPATH, and multiscale::Filesystem::isValidFilePath().

Referenced by isValidXmlPathAndFile().

---

**6.245.2.9** `bool XmlValidator::verifyIfValidXmlFile ( const std::string & xmlFilepath, const std::string & xmlSchemaPath, std::string & xmlErrorMessage ) [static, private]`

Check if the given xml file is valid considering the provided xml schema (xsd file)

The validation is performed using the Xerces C++ library.

In case the xml file is not valid the error message is retrieved in *xmlErrorMessage*.

#### Parameters

|                                       |                                                            |
|---------------------------------------|------------------------------------------------------------|
| <code><i>xmlFilepath</i></code>       | The path to the xml file                                   |
| <code><i>xmlSchema- Path</i></code>   | The path to the xml schema file                            |
| <code><i>xmlError- Message</i></code> | The error message explaining why the xml file is not valid |

Definition at line 37 of file XmlValidator.cpp.

References `checkIfValidXmlFile()`, and `multiscale::MultiscaleException::rawMessage()`.

Referenced by `isValidXmlPathAndFile()`.

### 6.245.3 Member Data Documentation

**6.245.3.1** `const std::string XmlValidator::ERR_INVALID_SCHEMA_FILEPATH = "The provided xml schema file path is invalid. Please change." [static, private]`

Definition at line 104 of file XmlValidator.hpp.

Referenced by `validateXmlSchemaPath()`.

**6.245.3.2** `const std::string XmlValidator::ERR_INVALID_XML_FILEPATH = "The provided xml file path is invalid. Please change." [static, private]`

Definition at line 103 of file XmlValidator.hpp.

Referenced by `validateXmlFilepath()`.

**6.245.3.3** `const std::string XmlValidator::ERR_SCHEMA_CONTENTS = "The provided xml schema is invalid. Please verify the xml schema contents." [static, private]`

Definition at line 106 of file XmlValidator.hpp.

Referenced by `loadParserSchema()`.

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

- `XmlValidator.hpp`

- `XmlValidator.cpp`