

Mercury library
1.0

Generated by Doxygen 1.8.7

Sun May 18 2014 20:59:43

Contents

1	Namespace Index	1
1.1	Packages	1
2	Hierarchical Index	3
2.1	Class Hierarchy	3
3	Class Index	7
3.1	Class List	7
4	File Index	11
4.1	File List	11
5	Namespace Documentation	13
5.1	Package Mercury	13
5.1.1	Enumeration Type Documentation	14
5.1.1.1	AnalyserStatus	14
5.1.1.2	FailReason	14
5.2	Package Mercury.Exceptions	14
5.3	Package Mercury.Interpreting	15
5.3.1	Enumeration Type Documentation	17
5.3.1.1	ArgumentType	17
5.3.1.2	ElementType	18
5.3.1.3	ParameterType	18
5.3.2	Function Documentation	18
5.3.2.1	RecursiveEval< T >	18
5.4	Package Mercury.Nucleus	19
5.5	Package Mercury.Nucleus.Collections	19
5.6	Package Mercury.Nucleus.Trees	19
5.7	Package Mercury.Scanning	20
5.7.1	Enumeration Type Documentation	20
5.7.1.1	TokenizerOptions	20
5.7.1.2	TokenType	20
5.8	Package Mercury.Syntax	21

5.8.1	Enumeration Type Documentation	22
5.8.1.1	Mapping	22
5.8.1.2	SymbolType	22
6	Class Documentation	23
6.1	Mercury.Interpreting.ActionCall< T > Class Template Reference	23
6.1.1	Detailed Description	23
6.1.2	Constructor & Destructor Documentation	24
6.1.2.1	ActionCall	24
6.1.3	Member Function Documentation	24
6.1.3.1	Equals	24
6.1.3.2	Equals	24
6.1.3.3	GetHashCode	24
6.1.3.4	ToString	24
6.1.4	Property Documentation	24
6.1.4.1	Action	24
6.1.4.2	FormalParameters	24
6.1.4.3	Type	25
6.2	Mercury.Interpreting.Alternative Class Reference	25
6.2.1	Detailed Description	26
6.2.2	Constructor & Destructor Documentation	26
6.2.2.1	Alternative	26
6.2.3	Member Function Documentation	26
6.2.3.1	Equals	26
6.2.3.2	Equals	26
6.2.3.3	GetHashCode	26
6.2.3.4	MatchesSymbol	26
6.2.3.5	ToString	27
6.2.4	Member Data Documentation	27
6.2.4.1	smset	27
6.2.5	Property Documentation	27
6.2.5.1	SymbolMask	27
6.2.5.2	Symbols	27
6.3	Mercury.Analyser< T > Class Template Reference	27
6.3.1	Detailed Description	28
6.3.2	Constructor & Destructor Documentation	29
6.3.2.1	Analyser	29
6.3.2.2	Analyser	29
6.4	Mercury.Analyser< T, TResult > Class Template Reference	29
6.4.1	Detailed Description	30

6.4.2	Constructor & Destructor Documentation	30
6.4.2.1	Analyser	30
6.4.2.2	Analyser	30
6.4.3	Member Function Documentation	31
6.4.3.1	Analyze	31
6.4.4	Property Documentation	31
6.4.4.1	Interpreter	31
6.4.4.2	Parser	31
6.4.4.3	Scanner	31
6.4.4.4	Tokenizer	31
6.5	Mercury.AnalyserResult< T, TResult > Class Template Reference	32
6.5.1	Detailed Description	32
6.5.2	Property Documentation	33
6.5.2.1	Exception	33
6.5.2.2	FailReason	33
6.5.2.3	Interpretations	33
6.5.2.4	InterpreterResults	33
6.5.2.5	InterpreterTime	33
6.5.2.6	ParserResult	33
6.5.2.7	ParserTime	33
6.5.2.8	Status	34
6.5.2.9	TotalTime	34
6.6	Mercury.Interpreting.Argument Class Reference	34
6.6.1	Detailed Description	34
6.6.2	Property Documentation	34
6.6.2.1	Type	34
6.6.2.2	Value	34
6.7	Mercury.Interpreting.ArithmeticAction< T > Class Template Reference	35
6.7.1	Detailed Description	35
6.7.2	Constructor & Destructor Documentation	35
6.7.2.1	ArithmeticAction	35
6.7.3	Member Function Documentation	36
6.7.3.1	InferReturnType	36
6.7.3.2	Invoke	36
6.8	Mercury.Interpreting.BasicActions< T > Class Template Reference	37
6.8.1	Detailed Description	41
6.8.2	Member Function Documentation	42
6.8.2.1	CreateTracer	42
6.8.2.2	GetAliases	42
6.8.2.3	GetAllActions	42

6.8.3	Member Data Documentation	43
6.8.3.1	Add	43
6.8.3.2	And	43
6.8.3.3	ArgsCount	43
6.8.3.4	Case	43
6.8.3.5	Ceiling	43
6.8.3.6	Concat	43
6.8.3.7	Depth	44
6.8.3.8	Div	44
6.8.3.9	Empty	44
6.8.3.10	Eq	44
6.8.3.11	Error	44
6.8.3.12	Eval	44
6.8.3.13	Fail	44
6.8.3.14	FDiv	45
6.8.3.15	Flatten	45
6.8.3.16	Floor	45
6.8.3.17	Geq	45
6.8.3.18	Gr	45
6.8.3.19	Identity	45
6.8.3.20	If	45
6.8.3.21	IsNull	46
6.8.3.22	Join	46
6.8.3.23	Leaves	46
6.8.3.24	Leq	46
6.8.3.25	Let	46
6.8.3.26	Ls	46
6.8.3.27	Mod	46
6.8.3.28	Mul	47
6.8.3.29	Name	47
6.8.3.30	Neq	47
6.8.3.31	Not	47
6.8.3.32	Or	47
6.8.3.33	ParseBoolean	47
6.8.3.34	ParseInteger	47
6.8.3.35	ParseReal	48
6.8.3.36	Pow	48
6.8.3.37	Preterminal	48
6.8.3.38	Round	48
6.8.3.39	Select	48

6.8.3.40	Simple	49
6.8.3.41	String	49
6.8.3.42	Sub	49
6.8.3.43	SubTree	49
6.8.3.44	Symbol	49
6.8.3.45	ToReal	49
6.8.3.46	Trace	50
6.8.3.47	Try	50
6.8.3.48	Var	50
6.8.3.49	Xor	50
6.9	Mercury.Syntax.BasicScanner Class Reference	50
6.9.1	Detailed Description	51
6.9.2	Constructor & Destructor Documentation	51
6.9.2.1	BasicScanner	51
6.9.3	Member Function Documentation	51
6.9.3.1	Scan	51
6.10	Mercury.Syntax.BasicScannerMap Class Reference	51
6.10.1	Detailed Description	52
6.10.2	Constructor & Destructor Documentation	52
6.10.2.1	BasicScannerMap	52
6.10.3	Member Function Documentation	52
6.10.3.1	GetMapping	52
6.10.3.2	SingleMap	53
6.10.4	Property Documentation	53
6.10.4.1	All	53
6.10.4.2	Identity	53
6.10.4.3	NumberOrString	53
6.10.4.4	this[TokenType ttype]	53
6.11	Mercury.Nucleus.Trees.BFSTreeEnumerator< T > Class Template Reference	53
6.11.1	Detailed Description	54
6.11.2	Constructor & Destructor Documentation	54
6.11.2.1	BFSTreeEnumerator	54
6.11.3	Member Function Documentation	54
6.11.3.1	Dispose	54
6.11.3.2	MoveNext	54
6.11.3.3	Reset	54
6.11.4	Property Documentation	54
6.11.4.1	Current	54
6.12	Mercury.Syntax.ChartParser Class Reference	55
6.12.1	Detailed Description	55

6.12.2	Constructor & Destructor Documentation	55
6.12.2.1	ChartParser	55
6.12.3	Member Function Documentation	56
6.12.3.1	Parse	56
6.12.3.2	Parse	56
6.12.4	Property Documentation	56
6.12.4.1	Grammar	56
6.13	Mercury.Syntax.CompositeScanner Class Reference	56
6.13.1	Detailed Description	57
6.13.2	Constructor & Destructor Documentation	57
6.13.2.1	CompositeScanner	57
6.13.2.2	CompositeScanner	57
6.13.3	Member Function Documentation	57
6.13.3.1	Add	57
6.13.3.2	GetEnumerator	58
6.13.3.3	Remove	58
6.13.3.4	Scan	58
6.14	Mercury.Interpreting.Constant Class Reference	58
6.14.1	Detailed Description	59
6.14.2	Constructor & Destructor Documentation	59
6.14.2.1	Constant	59
6.14.2.2	Constant	59
6.14.2.3	Constant	60
6.14.3	Member Function Documentation	60
6.14.3.1	Equals	60
6.14.3.2	Equals	60
6.14.3.3	Equals	60
6.14.3.4	GetHashCode	60
6.14.3.5	ToString	60
6.14.4	Property Documentation	60
6.14.4.1	Type	60
6.15	Mercury.Interpreting.ConstantAction< T, U > Class Template Reference	60
6.15.1	Detailed Description	61
6.15.2	Constructor & Destructor Documentation	61
6.15.2.1	ConstantAction	61
6.15.3	Member Function Documentation	61
6.15.3.1	Evaluate	61
6.16	Mercury.Interpreting.ConstantParameter Class Reference	62
6.16.1	Detailed Description	63
6.16.2	Constructor & Destructor Documentation	63

6.16.2.1	ConstantParameter	63
6.16.3	Member Function Documentation	63
6.16.3.1	AutoCreate	63
6.16.3.2	BooleanParameter	63
6.16.3.3	Equals	64
6.16.3.4	Equals	64
6.16.3.5	GetHashCode	64
6.16.3.6	IntegralParameter	64
6.16.3.7	RealParameter	64
6.16.3.8	StringParameter	64
6.16.3.9	ToString	64
6.16.4	Property Documentation	65
6.16.4.1	Type	65
6.16.4.2	Value	65
6.17	Mercury.Interpreting.ConvertAction< T, U, V > Class Template Reference	65
6.17.1	Detailed Description	65
6.17.2	Constructor & Destructor Documentation	66
6.17.2.1	ConvertAction	66
6.17.3	Member Function Documentation	66
6.17.3.1	Evaluate	66
6.18	Mercury.Interpreting.DataEntry Interface Reference	66
6.18.1	Detailed Description	66
6.19	Mercury.Defaults Class Reference	67
6.19.1	Detailed Description	67
6.19.2	Member Function Documentation	67
6.19.2.1	ParamToArgType	67
6.19.2.2	TypeToArgType< T, U >	68
6.19.3	Property Documentation	68
6.19.3.1	Epsilon	68
6.19.3.2	Root	68
6.20	Mercury.Nucleus.Trees.DFSTreeEnumerator< T > Class Template Reference	68
6.20.1	Detailed Description	69
6.20.2	Constructor & Destructor Documentation	69
6.20.2.1	DFSTreeEnumerator	69
6.20.3	Member Function Documentation	69
6.20.3.1	Dispose	69
6.20.3.2	MoveNext	69
6.20.3.3	Reset	69
6.20.4	Property Documentation	69
6.20.4.1	Current	69

6.21 Mercury.Syntax.Edge Class Reference	70
6.21.1 Detailed Description	70
6.21.2 Constructor & Destructor Documentation	71
6.21.2.1 Edge	71
6.21.3 Member Function Documentation	72
6.21.3.1 CompareTo	72
6.21.3.2 Equals	72
6.21.3.3 Equals	72
6.21.3.4 GetHashCode	72
6.21.3.5 IsValid	72
6.21.3.6 ToString	72
6.21.4 Property Documentation	73
6.21.4.1 Dot	73
6.21.4.2 IsActive	73
6.21.4.3 IsPassive	73
6.21.4.4 Left	73
6.21.4.5 NextSymbol	73
6.21.4.6 Right	73
6.21.4.7 Rule	73
6.22 Mercury.Interpreting.EmptyContextFactory Class Reference	73
6.22.1 Detailed Description	74
6.22.2 Member Function Documentation	74
6.22.2.1 CreateContext< T >	74
6.23 Mercury.Interpreting.EntryWriter< T > Class Template Reference	74
6.23.1 Detailed Description	74
6.23.2 Member Function Documentation	75
6.23.2.1 WriteEntries	75
6.24 Mercury.Exceptions.ErrorActionException Class Reference	75
6.24.1 Detailed Description	75
6.24.2 Constructor & Destructor Documentation	76
6.24.2.1 ErrorActionException	76
6.24.2.2 ErrorActionException	76
6.25 Mercury.Extensions Class Reference	76
6.25.1 Detailed Description	77
6.25.2 Member Function Documentation	77
6.25.2.1 ElementAtOrLast< T >	77
6.25.2.2 IntegralPow	77
6.25.2.3 Power2	77
6.25.2.4 RandomString	78
6.25.2.5 RandomString	78

6.25.2.6	Tokenize	78
6.25.2.7	ToMultiDictionary< TSource, TKey >	79
6.25.2.8	ToMultiDictionary< TSource, TKey, TValue >	80
6.25.2.9	Unescape	80
6.25.2.10	UniqueCharSeq	80
6.26	Mercury.Interpreting.FoldingAction< T, U, V > Class Template Reference	81
6.26.1	Detailed Description	81
6.26.2	Constructor & Destructor Documentation	82
6.26.2.1	FoldingAction	82
6.26.3	Member Function Documentation	83
6.26.3.1	Evaluate	83
6.27	Mercury.Interpreting.GenericAction< T > Class Template Reference	83
6.27.1	Detailed Description	84
6.27.2	Constructor & Destructor Documentation	84
6.27.2.1	GenericAction	84
6.27.3	Member Function Documentation	84
6.27.3.1	InferReturnType	84
6.28	Mercury.Syntax.Grammar Class Reference	85
6.28.1	Detailed Description	85
6.28.2	Property Documentation	85
6.28.2.1	Epsilon	85
6.28.2.2	Nonterminals	85
6.28.2.3	Root	85
6.28.2.4	Rules	85
6.28.2.5	Terminals	86
6.29	Mercury.Syntax.GrammarBuilder Class Reference	86
6.29.1	Detailed Description	86
6.29.2	Constructor & Destructor Documentation	87
6.29.2.1	GrammarBuilder	87
6.29.2.2	GrammarBuilder	87
6.29.3	Member Function Documentation	87
6.29.3.1	AddRule	87
6.29.3.2	AddRules	87
6.29.3.3	AddSymbol	87
6.29.3.4	GetGrammar	88
6.29.3.5	Include	88
6.29.4	Member Data Documentation	88
6.29.4.1	nonterminals	88
6.29.4.2	rules	88
6.29.4.3	terminals	88

6.29.5	Property Documentation	88
6.29.5.1	Epsilon	88
6.29.5.2	Root	89
6.30	Mercury.Interpreting.IElement Interface Reference	89
6.30.1	Detailed Description	89
6.30.2	Property Documentation	89
6.30.2.1	Type	89
6.31	Mercury.Interpreting.IFormalParameter Interface Reference	89
6.31.1	Detailed Description	90
6.31.2	Property Documentation	90
6.31.2.1	Type	90
6.32	Mercury.Interpreting.IInterpreter< T > Interface Template Reference	90
6.32.1	Detailed Description	91
6.32.2	Member Function Documentation	92
6.32.2.1	Interpret	92
6.32.2.2	Interpret	92
6.32.3	Property Documentation	92
6.32.3.1	RewriteRules	92
6.33	Mercury.Interpreting.IInterpreterContextFactory Interface Reference	92
6.33.1	Detailed Description	93
6.33.2	Member Function Documentation	93
6.33.2.1	CreateContext< T >	93
6.34	Mercury.Nucleus.Collections.IMultiDictionary< TKey, TValue > Interface Template Reference	93
6.34.1	Detailed Description	94
6.34.2	Member Function Documentation	94
6.34.2.1	Add	94
6.34.2.2	Add	94
6.34.2.3	Add	95
6.34.2.4	Clear	95
6.35	Mercury.Interpreting.Instance Class Reference	95
6.35.1	Detailed Description	96
6.35.2	Member Function Documentation	96
6.35.2.1	Equals	96
6.35.2.2	Equals	96
6.35.2.3	GetHashCode	96
6.35.3	Property Documentation	96
6.35.3.1	Value	96
6.35.3.2	Var	96
6.36	Mercury.Interpreting.InstanceList Class Reference	97
6.36.1	Detailed Description	97

6.36.2	Member Function Documentation	97
6.36.2.1	GetEnumerator	97
6.36.3	Property Documentation	97
6.36.3.1	Count	97
6.36.3.2	this[string name]	97
6.37	Mercury.Interpreting.Interpreter< T > Class Template Reference	98
6.37.1	Detailed Description	98
6.37.2	Constructor & Destructor Documentation	99
6.37.2.1	Interpreter	99
6.37.3	Member Function Documentation	99
6.37.3.1	Interpret	99
6.37.3.2	Interpret	99
6.37.4	Property Documentation	100
6.37.4.1	RewriteRules	100
6.38	Mercury.Interpreting.InterpreterAction< T > Class Template Reference	100
6.38.1	Detailed Description	101
6.38.2	Constructor & Destructor Documentation	101
6.38.2.1	InterpreterAction	101
6.38.3	Member Function Documentation	102
6.38.3.1	CheckDefinition	102
6.38.3.2	Invoke	102
6.38.4	Property Documentation	103
6.38.4.1	AllowsWildcard	103
6.38.4.2	ArgumentTypes	103
6.38.4.3	Arity	103
6.38.4.4	Name	103
6.38.4.5	ReturnType	103
6.39	Mercury.Interpreting.InterpreterAction< T, U > Class Template Reference	103
6.39.1	Detailed Description	104
6.39.2	Constructor & Destructor Documentation	104
6.39.2.1	InterpreterAction	104
6.39.3	Member Function Documentation	104
6.39.3.1	Evaluate	104
6.39.3.2	Invoke	105
6.40	Mercury.Interpreting.InterpreterContext< T > Class Template Reference	105
6.40.1	Detailed Description	105
6.40.2	Constructor & Destructor Documentation	106
6.40.2.1	InterpreterContext	106
6.40.3	Member Function Documentation	106
6.40.3.1	CopyFrom	106

6.40.3.2	DeepClone	106
6.40.3.3	LogEntry	106
6.40.4	Property Documentation	106
6.40.4.1	RootEntry	106
6.41	Mercury.Interpreting.InterpreterResult< T > Class Template Reference	107
6.41.1	Detailed Description	107
6.41.2	Property Documentation	107
6.41.2.1	Context	107
6.41.2.2	Exception	107
6.41.2.3	Input	107
6.41.2.4	Result	108
6.41.2.5	Success	108
6.42	Mercury.Exceptions.InterpretingException Class Reference	108
6.42.1	Detailed Description	108
6.42.2	Constructor & Destructor Documentation	108
6.42.2.1	InterpretingException	108
6.42.2.2	InterpretingException	109
6.42.2.3	InterpretingException	109
6.42.2.4	InterpretingException	109
6.43	Mercury.Exceptions.InvalidActionArgumentException Class Reference	109
6.43.1	Detailed Description	110
6.43.2	Constructor & Destructor Documentation	110
6.43.2.1	InvalidActionArgumentException	110
6.44	Mercury.Exceptions.InvalidActionCallException Class Reference	110
6.44.1	Detailed Description	111
6.44.2	Constructor & Destructor Documentation	111
6.44.2.1	InvalidActionCallException	111
6.45	Mercury.Exceptions.InvalidActionException Class Reference	111
6.45.1	Detailed Description	111
6.45.2	Constructor & Destructor Documentation	112
6.45.2.1	InvalidActionException	112
6.46	Mercury.Exceptions.InvalidEdgeException Class Reference	112
6.46.1	Detailed Description	112
6.46.2	Constructor & Destructor Documentation	112
6.46.2.1	InvalidEdgeException	112
6.47	Mercury.Exceptions.InvalidElementException Class Reference	113
6.47.1	Detailed Description	113
6.47.2	Constructor & Destructor Documentation	113
6.47.2.1	InvalidElementException	113
6.48	Mercury.Exceptions.InvalidRewriteRuleException Class Reference	114

6.48.1 Detailed Description	114
6.48.2 Constructor & Destructor Documentation	114
6.48.2.1 InvalidRewriteRuleException	114
6.49 Mercury.Exceptions.InvalidRuleException Class Reference	114
6.49.1 Detailed Description	115
6.49.2 Constructor & Destructor Documentation	115
6.49.2.1 InvalidRuleException	115
6.50 Mercury.Exceptions.InvalidSymbolException Class Reference	115
6.50.1 Detailed Description	116
6.50.2 Constructor & Destructor Documentation	116
6.50.2.1 InvalidSymbolException	116
6.51 Mercury.Syntax.IParser Interface Reference	116
6.51.1 Detailed Description	117
6.51.2 Member Function Documentation	117
6.51.2.1 Parse	117
6.51.2.2 Parse	117
6.51.3 Property Documentation	117
6.51.3.1 Grammar	117
6.52 Mercury.Nucleus.Collections.IReadOnlyMultiDictionary< TKey, TValue > Interface Template Reference	118
6.52.1 Detailed Description	118
6.52.2 Member Function Documentation	118
6.52.2.1 Contains	118
6.52.2.2 ContainsKey	119
6.52.2.3 TryGetValue	119
6.52.3 Property Documentation	119
6.52.3.1 Keys	119
6.52.3.2 this[TKey key]	120
6.53 Mercury.Syntax.IScanner Interface Reference	121
6.53.1 Detailed Description	121
6.53.2 Member Function Documentation	121
6.53.2.1 Scan	121
6.54 Mercury.Scanning.ITokenizer Interface Reference	122
6.54.1 Detailed Description	122
6.54.2 Member Function Documentation	122
6.54.2.1 Tokenize	122
6.55 Mercury.LanguageInfo< T > Class Template Reference	123
6.55.1 Detailed Description	123
6.55.2 Constructor & Destructor Documentation	123
6.55.2.1 LanguageInfo	123

6.55.3	Property Documentation	124
6.55.3.1	Grammar	124
6.55.3.2	Name	124
6.55.3.3	RewriteRules	124
6.55.3.4	Scanner	124
6.55.3.5	Tokenizer	124
6.56	Mercury.Nucleus.Collections.ListMultiDictionary< TKey, TValue > Class Template Reference	124
6.56.1	Detailed Description	125
6.56.2	Constructor & Destructor Documentation	126
6.56.2.1	ListMultiDictionary	126
6.56.2.2	ListMultiDictionary	126
6.56.3	Member Function Documentation	126
6.56.3.1	Add	126
6.56.3.2	Add	126
6.56.3.3	Add	127
6.56.3.4	Clear	127
6.56.3.5	Contains	127
6.56.3.6	ContainsKey	128
6.56.3.7	GetEnumerator	128
6.56.3.8	GetValue	128
6.56.3.9	GetValuesCount	128
6.56.3.10	TryGetValue	129
6.56.4	Property Documentation	129
6.56.4.1	Keys	129
6.56.4.2	this[TKey key]	129
6.57	Mercury.Interpreting.MemoEntry< T > Class Template Reference	129
6.57.1	Detailed Description	130
6.57.2	Property Documentation	130
6.57.2.1	MemoizedValue	130
6.58	Mercury.Exceptions.MercuryException Class Reference	130
6.58.1	Detailed Description	131
6.58.2	Constructor & Destructor Documentation	131
6.58.2.1	MercuryException	131
6.58.2.2	MercuryException	131
6.58.2.3	MercuryException	131
6.58.2.4	MercuryException	131
6.59	Mercury.Interpreting.OperationAction< T, U, V > Class Template Reference	132
6.59.1	Detailed Description	132
6.59.2	Constructor & Destructor Documentation	132
6.59.2.1	OperationAction	132

6.59.3	Member Function Documentation	133
6.59.3.1	Evaluate	133
6.60	Mercury.Syntax.ParserResult Class Reference	133
6.60.1	Detailed Description	133
6.60.2	Property Documentation	134
6.60.2.1	Accepted	134
6.60.2.2	Chart	134
6.60.2.3	InputTokens	134
6.60.2.4	Trees	134
6.61	Mercury.Interpreting.RewriteRule< T > Class Template Reference	134
6.61.1	Detailed Description	135
6.61.2	Constructor & Destructor Documentation	135
6.61.2.1	RewriteRule	135
6.61.3	Member Function Documentation	135
6.61.3.1	Equals	135
6.61.3.2	Equals	136
6.61.3.3	GetHashCode	136
6.61.3.4	ToString	136
6.61.4	Property Documentation	136
6.61.4.1	Head	136
6.61.4.2	LHS	136
6.61.4.3	RHS	136
6.61.4.4	Variables	136
6.62	Mercury.Interpreting.RewriteRuleCollection< T > Class Template Reference	136
6.62.1	Detailed Description	137
6.62.2	Member Function Documentation	137
6.62.2.1	GetEnumerator	137
6.62.2.2	RulesFor	137
6.62.3	Property Documentation	137
6.62.3.1	Count	137
6.63	Mercury.Interpreting.RewriteRuleCollectionBuilder< T > Class Template Reference	137
6.63.1	Detailed Description	138
6.63.2	Constructor & Destructor Documentation	138
6.63.2.1	RewriteRuleCollectionBuilder	138
6.63.3	Member Function Documentation	138
6.63.3.1	AddRewriteRule	138
6.63.3.2	AddRules	138
6.63.3.3	GetCollection	139
6.64	Mercury.Syntax.Rule Class Reference	139
6.64.1	Detailed Description	140

6.64.2	Constructor & Destructor Documentation	140
6.64.2.1	Rule	140
6.64.3	Member Function Documentation	140
6.64.3.1	CompareTo	140
6.64.3.2	Equals	140
6.64.3.3	Equals	141
6.64.3.4	GetHashCode	141
6.64.3.5	ToString	141
6.64.4	Property Documentation	141
6.64.4.1	IsEpsilon	141
6.64.4.2	LHS	141
6.64.4.3	RHS	141
6.65	Mercury.Interpreting.RuleEntry< T > Class Template Reference	141
6.65.1	Detailed Description	142
6.65.2	Property Documentation	142
6.65.2.1	Error	142
6.65.2.2	Instances	142
6.65.2.3	Match	142
6.65.2.4	RewriteRule	142
6.65.2.5	SubEntries	143
6.65.2.6	Value	143
6.66	Mercury.Interpreting.SemanticInterpreter< T > Class Template Reference	143
6.66.1	Detailed Description	143
6.66.2	Constructor & Destructor Documentation	144
6.66.2.1	SemanticInterpreter	144
6.66.3	Member Function Documentation	144
6.66.3.1	Interpret	144
6.66.3.2	Interpret	144
6.66.4	Property Documentation	145
6.66.4.1	RewriteRules	145
6.66.4.2	SemanticRules	145
6.67	Mercury.Nucleus.Collections.SetMultiDictionary< TKey, TValue > Class Template Reference	145
6.67.1	Detailed Description	146
6.67.2	Member Function Documentation	146
6.67.2.1	Add	146
6.67.2.2	Add	147
6.67.2.3	Add	147
6.67.2.4	Clear	147
6.67.2.5	Contains	147
6.67.2.6	ContainsKey	148

6.67.2.7	GetEnumerator	148
6.67.2.8	TryGetValue	148
6.67.3	Member Data Documentation	149
6.67.3.1	data	149
6.67.4	Property Documentation	149
6.67.4.1	Keys	149
6.67.4.2	this[TKey key]	149
6.68	Mercury.Interpreting.Structure Class Reference	149
6.68.1	Detailed Description	150
6.68.2	Constructor & Destructor Documentation	150
6.68.2.1	Structure	150
6.68.2.2	Structure	150
6.68.3	Member Function Documentation	150
6.68.3.1	Equals	150
6.68.4	Property Documentation	150
6.68.4.1	HasWildcard	150
6.68.4.2	Type	150
6.69	Mercury.Syntax.Symbol Class Reference	151
6.69.1	Detailed Description	152
6.69.2	Constructor & Destructor Documentation	152
6.69.2.1	Symbol	152
6.69.2.2	Symbol	152
6.69.2.3	Symbol	152
6.69.2.4	Symbol	153
6.69.3	Member Function Documentation	153
6.69.3.1	CompareTo	153
6.69.3.2	Equals	153
6.69.3.3	Equals	153
6.69.3.4	GetHashCode	153
6.69.3.5	InferType	153
6.69.3.6	ToString	154
6.69.4	Property Documentation	154
6.69.4.1	IsEpsilon	154
6.69.4.2	Name	154
6.69.4.3	Token	154
6.69.4.4	Type	154
6.70	Mercury.Exceptions.SyntaxException Class Reference	154
6.70.1	Detailed Description	155
6.70.2	Constructor & Destructor Documentation	155
6.70.2.1	SyntaxException	155

6.70.2.2	SyntaxException	155
6.70.2.3	SyntaxException	155
6.70.2.4	SyntaxException	156
6.71	Mercury.Interpreting.TextEntry Class Reference	156
6.71.1	Detailed Description	156
6.71.2	Property Documentation	156
6.71.2.1	ActionName	156
6.71.2.2	Text	157
6.72	Mercury.Scanning.Token Class Reference	157
6.72.1	Detailed Description	157
6.72.2	Constructor & Destructor Documentation	158
6.72.2.1	Token	158
6.72.3	Member Function Documentation	159
6.72.3.1	Equals	159
6.72.3.2	Equals	159
6.72.3.3	GetHashCode	159
6.72.3.4	ToString	159
6.72.4	Property Documentation	159
6.72.4.1	Position	159
6.72.4.2	Quoted	159
6.72.4.3	Type	159
6.72.4.4	Value	159
6.73	Mercury.Scanning.Tokenizer Class Reference	160
6.73.1	Detailed Description	160
6.73.2	Constructor & Destructor Documentation	161
6.73.2.1	Tokenizer	161
6.73.2.2	Tokenizer	161
6.73.2.3	Tokenizer	161
6.73.3	Member Function Documentation	162
6.73.3.1	Tokenize	162
6.73.4	Property Documentation	162
6.73.4.1	Alpha	162
6.73.4.2	Keywords	162
6.73.4.3	Options	162
6.74	Mercury.Nucleus.Trees.Tree< T > Class Template Reference	162
6.74.1	Detailed Description	163
6.74.2	Constructor & Destructor Documentation	164
6.74.2.1	Tree	164
6.74.2.2	Tree	164
6.74.2.3	Tree	164

6.74.3	Member Function Documentation	164
6.74.3.1	CompareTo	164
6.74.3.2	Equals	164
6.74.3.3	Equals	164
6.74.3.4	GetEnumerator	164
6.74.3.5	GetHashCode	165
6.74.3.6	ToString	165
6.74.3.7	ToString	165
6.74.4	Property Documentation	165
6.74.4.1	BreadthFirstEnumerator	165
6.74.4.2	Children	165
6.74.4.3	Depth	165
6.74.4.4	DepthFirstEnumerator	165
6.74.4.5	IsLeaf	165
6.74.4.6	LeavesCount	165
6.74.4.7	NodesCount	166
6.74.4.8	this[int ix]	166
6.74.4.9	Value	166
6.75	Mercury.Interpreting.TreeEntry< T > Class Template Reference	166
6.75.1	Detailed Description	166
6.75.2	Property Documentation	167
6.75.2.1	RuleEntries	167
6.75.2.2	Succeeded	167
6.75.2.3	Tree	167
6.76	Mercury.Nucleus.Collections.UniqueListMultiDictionary< TKey, TValue > Class Template Reference	167
6.76.1	Detailed Description	168
6.76.2	Member Function Documentation	168
6.76.2.1	Add	168
6.76.2.2	Clear	168
6.76.2.3	Contains	168
6.76.3	Member Data Documentation	169
6.76.3.1	data	169
6.77	Mercury.Interpreting.Variable Class Reference	169
6.77.1	Detailed Description	170
6.77.2	Constructor & Destructor Documentation	170
6.77.2.1	Variable	170
6.77.2.2	Variable	170
6.77.3	Member Function Documentation	170
6.77.3.1	Equals	170
6.77.3.2	Equals	170

6.77.3.3	Equals	171
6.77.3.4	GetHashCode	171
6.77.3.5	IsValidName	171
6.77.3.6	ToString	171
6.77.4	Property Documentation	171
6.77.4.1	Name	171
6.77.4.2	Type	171
6.78	Mercury.Interpreting.VariableParameter Class Reference	171
6.78.1	Detailed Description	172
6.78.2	Constructor & Destructor Documentation	172
6.78.2.1	VariableParameter	172
6.78.3	Member Function Documentation	172
6.78.3.1	Equals	172
6.78.3.2	Equals	172
6.78.3.3	GetHashCode	173
6.78.3.4	ToString	173
6.78.4	Property Documentation	173
6.78.4.1	Name	173
6.78.4.2	Type	173
6.79	Mercury.Interpreting.Wildcard Class Reference	173
6.79.1	Detailed Description	174
6.79.2	Constructor & Destructor Documentation	174
6.79.2.1	Wildcard	174
6.79.2.2	Wildcard	174
6.79.3	Member Function Documentation	174
6.79.3.1	Equals	174
6.79.3.2	Equals	174
6.79.3.3	Equals	174
6.79.3.4	GetHashCode	174
6.79.3.5	ToString	174
7	File Documentation	175
7.1	Mercury/Mercury/Analyser.cs File Reference	175
7.2	Mercury/Mercury/AnalyserResult.cs File Reference	175
7.3	Mercury/Mercury/Defaults.cs File Reference	176
7.4	Mercury/Mercury/Exceptions/InterpretingExceptions.cs File Reference	176
7.5	Mercury/Mercury/Exceptions/MercuryException.cs File Reference	176
7.6	Mercury/Mercury/Exceptions/SyntaxExceptions.cs File Reference	177
7.7	Mercury/Mercury/Extensions.cs File Reference	177
7.8	Mercury/Mercury/Interpreting/ActionCall.cs File Reference	177

7.9	Mercury/Mercury/Interpreting/Argument.cs File Reference	177
7.10	Mercury/Mercury/Interpreting/BasicActions.cs File Reference	178
7.11	Mercury/Mercury/Interpreting/Element.cs File Reference	179
7.12	Mercury/Mercury/Interpreting/Entries.cs File Reference	179
7.13	Mercury/Mercury/Interpreting/Evaluator.cs File Reference	180
7.14	Mercury/Mercury/Interpreting/FormalParameter.cs File Reference	180
7.15	Mercury/Mercury/Interpreting/Instance.cs File Reference	180
7.16	Mercury/Mercury/Interpreting/Interpreter.cs File Reference	181
7.17	Mercury/Mercury/Interpreting/InterpreterAction.cs File Reference	181
7.18	Mercury/Mercury/Interpreting/InterpreterContext.cs File Reference	182
7.19	Mercury/Mercury/Interpreting/InterpreterResult.cs File Reference	182
7.20	Mercury/Mercury/Interpreting/RewriteRule.cs File Reference	182
7.21	Mercury/Mercury/Interpreting/RewriteRuleCollection.cs File Reference	182
7.22	Mercury/Mercury/LanguageInfo.cs File Reference	183
7.23	Mercury/Mercury/Nucleus/Collections/MultiDictionary.cs File Reference	183
7.24	Mercury/Mercury/Nucleus/Trees/Tree.cs File Reference	183
7.25	Mercury/Mercury/Nucleus/Trees/TreeEnumerators.cs File Reference	184
7.26	Mercury/Mercury/obj/Debug/TemporaryGeneratedFile_036C0B5B-1481-4323-8D20-8F5ADCB23D92.cs File Reference	184
7.27	Mercury/Mercury/obj/Release/TemporaryGeneratedFile_036C0B5B-1481-4323-8D20-8F5ADC↵B23D92.cs File Reference	184
7.28	Mercury/Mercury/obj/Debug/TemporaryGeneratedFile_5937a670-0e60-4077-877b-f7221da3dda1.cs File Reference	184
7.29	Mercury/Mercury/obj/Release/TemporaryGeneratedFile_5937a670-0e60-4077-877b-f7221da3dda1.cs File Reference	184
7.30	Mercury/Mercury/obj/Debug/TemporaryGeneratedFile_E7A71F73-0F8D-4B9B-B56E-8E70B10BC5D3.cs File Reference	184
7.31	Mercury/Mercury/obj/Release/TemporaryGeneratedFile_E7A71F73-0F8D-4B9B-B56E-8E70B10↵BC5D3.cs File Reference	184
7.32	Mercury/Mercury/Properties/AssemblyInfo.cs File Reference	184
7.33	Mercury/Mercury/Scanning/Token.cs File Reference	184
7.34	Mercury/Mercury/Scanning/Tokenizer.cs File Reference	185
7.35	Mercury/Mercury/Syntax/Edge.cs File Reference	185
7.36	Mercury/Mercury/Syntax/EdgeDerivation.cs File Reference	185
7.37	Mercury/Mercury/Syntax/Grammar.cs File Reference	186
7.38	Mercury/Mercury/Syntax/GrammarBuilder.cs File Reference	186
7.39	Mercury/Mercury/Syntax/Parser.cs File Reference	186
7.40	Mercury/Mercury/Syntax/ParserResult.cs File Reference	187
7.41	Mercury/Mercury/Syntax/Rule.cs File Reference	187
7.42	Mercury/Mercury/Syntax/Scanner.cs File Reference	187
7.43	Mercury/Mercury/Syntax/Symbol.cs File Reference	188

Chapter 1

Namespace Index

1.1 Packages

Here are the packages with brief descriptions (if available):

Mercury	13
Mercury.Exceptions	14
Mercury.Interpreting	15
Mercury.Nucleus	19
Mercury.Nucleus.Collections	19
Mercury.Nucleus.Trees	19
Mercury.Scanning	20
Mercury.Syntax	21

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

Mercury.Analyser< T, TResult >	29
Mercury.Analyser< T >	27
Mercury.AnalyserResult< T, TResult >	32
Mercury.Interpreting.Argument	34
Mercury.Interpreting.BasicActions< T >	37
Mercury.Syntax.BasicScannerMap	51
Mercury.Interpreting.DataEntry	66
Mercury.Interpreting.MemoEntry< T >	129
Mercury.Interpreting.RuleEntry< T >	141
Mercury.Interpreting.TextEntry	156
Mercury.Interpreting.TreeEntry< T >	166
Mercury.Defaults	67
Mercury.Interpreting.EntryWriter< T >	74
Exception	
Mercury.Exceptions.MercuryException	130
Mercury.Exceptions.InterpretingException	108
Mercury.Exceptions.ErrorActionException	75
Mercury.Exceptions.InvalidActionArgumentException	109
Mercury.Exceptions.InvalidActionCallException	110
Mercury.Exceptions.InvalidActionException	111
Mercury.Exceptions.InvalidElementException	113
Mercury.Exceptions.InvalidRewriteRuleException	114
Mercury.Exceptions.SyntaxException	154
Mercury.Exceptions.InvalidEdgeException	112
Mercury.Exceptions.InvalidRuleException	114
Mercury.Exceptions.InvalidSymbolException	115
Mercury.Extensions	76
Mercury.Syntax.Grammar	85
Mercury.Syntax.GrammarBuilder	86
Comparable< Edge >	
Mercury.Syntax.Edge	70
Comparable< Rule >	
Mercury.Syntax.Rule	139
Comparable< Symbol >	
Mercury.Syntax.Symbol	151
Comparable< Tree< T >>	
Mercury.Nucleus.Trees.Tree< T >	162

IEnumerable	
Mercury.Nucleus.Trees.Tree< T >	162
IEnumerable< Instance >	
Mercury.Interpreting.InstanceList	97
IEnumerable< IScanner >	
Mercury.Syntax.CompositeScanner	56
IEnumerable< KeyValuePair< TKey, TValue >>	
Mercury.Nucleus.Collections.IReadOnlyMultiDictionary< TKey, TValue >	118
Mercury.Nucleus.Collections.IMultiDictionary< TKey, TValue >	93
Mercury.Nucleus.Collections.ListMultiDictionary< TKey, TValue >	124
Mercury.Nucleus.Collections.UniqueListMultiDictionary< TKey, TValue >	167
Mercury.Nucleus.Collections.SetMultiDictionary< TKey, TValue >	145
Mercury.Nucleus.Collections.UniqueListMultiDictionary< TKey, TValue >	167
IEnumerable< T >	
Mercury.Nucleus.Trees.Tree< T >	162
IEnumerator	
Mercury.Nucleus.Trees.BFSTreeEnumerator< T >	53
Mercury.Nucleus.Trees.DFSTreeEnumerator< T >	68
IEnumerator< T >	
Mercury.Nucleus.Trees.BFSTreeEnumerator< T >	53
Mercury.Nucleus.Trees.DFSTreeEnumerator< T >	68
IEquatable< ActionCall< T >>	
Mercury.Interpreting.ActionCall< T >	23
IEquatable< Alternative >	
Mercury.Interpreting.Alternative	25
Mercury.Interpreting.Constant	58
Mercury.Interpreting.Variable	169
Mercury.Interpreting.Wildcard	173
IEquatable< Constant >	
Mercury.Interpreting.Constant	58
IEquatable< ConstantParameter >	
Mercury.Interpreting.ConstantParameter	62
IEquatable< Edge >	
Mercury.Syntax.Edge	70
IEquatable< IElement >	
Mercury.Interpreting.IElement	89
Mercury.Interpreting.Constant	58
Mercury.Interpreting.Structure	149
Mercury.Interpreting.Variable	169
IEquatable< Instance >	
Mercury.Interpreting.Instance	95
IEquatable< RewriteRule< T >>	
Mercury.Interpreting.RewriteRule< T >	134
IEquatable< Rule >	
Mercury.Syntax.Rule	139
IEquatable< Symbol >	
Mercury.Syntax.Symbol	151
IEquatable< Token >	
Mercury.Scanning.Token	157
IEquatable< Tree< T >>	
Mercury.Nucleus.Trees.Tree< T >	162
IEquatable< Variable >	
Mercury.Interpreting.Variable	169
IEquatable< VariableParameter >	
Mercury.Interpreting.VariableParameter	171
Mercury.Interpreting.IFormalParameter	89
Mercury.Interpreting.ActionCall< T >	23
Mercury.Interpreting.ConstantParameter	62

Mercury.Interpreting.VariableParameter	171
Mercury.Interpreting.IInterpreter< T >	90
Mercury.Interpreting.Interpreter< T >	98
Mercury.Interpreting.SemanticInterpreter< T >	143
Mercury.Interpreting.IInterpreterContextFactory	92
Mercury.Interpreting.EmptyContextFactory	73
Mercury.Interpreting.IInterpreterAction< T >	100
Mercury.Interpreting.GenericAction< T >	83
Mercury.Interpreting.ArithmeticAction< T >	35
Mercury.Interpreting.IInterpreterAction< T, U >	103
Mercury.Interpreting.ConstantAction< T, U >	60
Mercury.Interpreting.ConvertAction< T, U, V >	65
Mercury.Interpreting.FoldingAction< T, U, V >	81
Mercury.Interpreting.OperationAction< T, U, V >	132
Mercury.Interpreting.IInterpreterContext< T >	105
Mercury.Interpreting.IInterpreterResult< T >	107
Mercury.Syntax.IParser	116
Mercury.Syntax.ChartParser	55
IReadOnlyCollection< RewriteRule< T >>	
Mercury.Interpreting.RewriteRuleCollection< T >	136
Mercury.Syntax.IScanner	121
Mercury.Syntax.BasicScanner	50
Mercury.Syntax.CompositeScanner	56
Mercury.Scanning.ITokenizer	122
Mercury.Scanning.Tokenizer	160
Mercury.LanguageInfo< T >	123
Mercury.Syntax.ParserResult	133
Mercury.Interpreting.RewriteRuleCollectionBuilder< T >	137
Tree< IElement >	
Mercury.Interpreting.Structure	149

Chapter 3

Class Index

3.1 Class List

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

Mercury.Interpreting.ActionCall< T >	
This class represents the application of Mercury.Interpreting.InterpreterAction{T} on the Mercury.Interpreting.IFormalParameter list.	23
Mercury.Interpreting.Alternative	
Represents an alternative of elements. It may match specified symbols or symboltypes (Nonterminals, ...).	25
Mercury.Analyser< T >	
Wrapper class for each step of analysis	27
Mercury.Analyser< T, TResult >	
Wrapper class for each step of analysis.	29
Mercury.AnalyserResult< T, TResult >	
Holds the result of analysis.	
The fields hold information according to the following table:	
32	
Mercury.Interpreting.Argument	
Represents an argument passed to the Mercury.Interpreting.InterpreterAction{T} , the actual parameter given to the Mercury.Interpreting.Evaluator{T} in place of a formal parameter.	34
Mercury.Interpreting.ArithmeticAction< T >	
Represents the arithmetic action. It is similar to the Mercury.Interpreting.FoldingAction{T,U,V} but does not require initial value. The minimal number of arguments is therefore 2. For instance, "1 2 3" would be evaluated as $f(f(1, 2), 3)$	35
Mercury.Interpreting.BasicActions< T >	
Contains basic actions for interpretation.	
Type specifiers of actions are defined as follows:	
37	
Mercury.Syntax.BasicScanner	
Generates four Nonterminals @ Symbol , @ Keyword , @ Number and @ String based on the given map and types of input tokens.	50
Mercury.Syntax.BasicScannerMap	
Contains mapping between TokenType and Mercury.Syntax.Mapping of Mercury.Syntax.BasicScanner	51
Mercury.Nucleus.Trees.BFSTreeEnumerator< T >	
Breadth-First Search tree enumerator	53
Mercury.Syntax.ChartParser	
Thread-safe wrapper for ChartParserInternal class.	55

Mercury.Syntax.CompositeScanner	
Mercury.Syntax.IScanner implementation that behaves like container of scanners using the Composite Design Pattern	56
Mercury.Interpreting.Constant	
Constant element. Basically equivalent of Mercury.Interpreting.Alternative	58
Mercury.Interpreting.ConstantAction< T, U >	
Wraps a value as an action with no parameters	60
Mercury.Interpreting.ConstantParameter	
Constant parameter.	62
Mercury.Interpreting.ConvertAction< T, U, V >	
Applies the lambda function on a single value.	65
Mercury.Interpreting.DataEntry	
Entry base (dummy) interface.	66
Mercury.Defaults	
Global class containing cool static / convenience stuff.	67
Mercury.Nucleus.Trees.DFSTreeEnumerator< T >	
Depth-First Search tree enumerator	68
Mercury.Syntax.Edge	
Represents an edge in the chart created by the ChartParser . This is an immutable class.	
The Edge contains four values:	
70	
Mercury.Interpreting.EmptyContextFactory	
Produces contexts with no custom data	73
Mercury.Interpreting.EntryWriter< T >	
An auxiliary class that formats the DataEntry structure and writes it into a stream.	74
Mercury.Exceptions.ErrorActionException	
Exception thrown by basic actions when a problem was encountered.	75
Mercury.Extensions	
Provides extension methods for the Mercury library.	76
Mercury.Interpreting.FoldingAction< T, U, V >	
"Folds" all its arguments into one value. For instance, for "x y z" arguments this action computes $f(f(f(s, x), y), z)$ where f is the lambda function and s is the <code>seed</code> parameter of the constructor	81
Mercury.Interpreting.GenericAction< T >	
Represents an action that infers its return type only after it is provided with types of its arguments.	83
Mercury.Syntax.Grammar	
Represents a Context-Free Grammar . This is an immutable class, to create a grammar use Mercury.Syntax.GrammarBuilder	85
Mercury.Syntax.GrammarBuilder	
Class that creates grammars.	86
Mercury.Interpreting.IElement	
Element interface	89
Mercury.Interpreting.IFormalParameter	
Represents formal parameters.	89
Mercury.Interpreting.IInterpreter< T >	
Interface for interpreters that create objects from derivation trees according to RewriteRules.	90
Mercury.Interpreting.IInterpreterContextFactory	
Factory of context.	92
Mercury.Nucleus.Collections.IMultiDictionary< TKey, TValue >	
Represents a multi-valued dictionary, in other words a map between a key and a collection of values.	93
Mercury.Interpreting.Instance	
Represents a binding between a Mercury.Interpreting.Variable and its value	95
Mercury.Interpreting.InstanceList	
Represents the list of instances	97

Mercury.Interpreting.Interpreter< T >	
Interprets derivation trees using rewrite rules. The type parameter <i>T</i> represents the target language.	
98	
Mercury.Interpreting.InterpreterAction< T >	
Function that wraps constant values in the RHS	100
Mercury.Interpreting.InterpreterAction< T, U >	
Interpreter action with a safer return type.	103
Mercury.Interpreting.InterpreterContext< T >	
Interpreter Context class. Some languages may use it to override it and remember data when parse trees are being interpreted.	105
Mercury.Interpreting.InterpreterResult< T >	
Result of the interpretation.	107
Mercury.Exceptions.InterpretingException	
An exception thrown because of problems in the Interpreting namespace.	108
Mercury.Exceptions.InvalidActionArgumentException	
An exception representing a problem while type checking the Mercury.Interpreting.RewriteRule { <i>T</i> }.	109
Mercury.Exceptions.InvalidActionCallException	
An exception representing a problem with Mercury.Interpreting.ActionCall { <i>T</i> }	110
Mercury.Exceptions.InvalidActionException	
An exception representing a problem with Mercury.Interpreting.InterpreterAction { <i>T</i> } class.	111
Mercury.Exceptions.InvalidEdgeException	
An exception thrown when attempting to create an invalid Mercury.Parser.Edge instance.	112
Mercury.Exceptions.InvalidElementException	
An exception representing a problem with Mercury.Interpreting.IElement descendants.	113
Mercury.Exceptions.InvalidRewriteRuleException	
An exception representing a problem with Mercury.Interpreting.RewriteRule { <i>T</i> }.	114
Mercury.Exceptions.InvalidRuleException	
An exception thrown when attempting to create an invalid Mercury.Syntax.Rule instance.	114
Mercury.Exceptions.InvalidSymbolException	
An exception thrown when attempting to create an invalid Mercury.Syntax.Symbol instance.	115
Mercury.Syntax.IParser	
Creates a derivation tree from the list of input tokens	116
Mercury.Nucleus.Collections.IReadOnlyMultiDictionary< TKey, TValue >	
Represents a multi-valued read-only dictionary, a map between a key and a collection of values.	118
Mercury.Syntax.IScanner	
Scanner scans input tokens and creates additional rules for parsing	121
Mercury.Scanning.ITokenizer	
The tokenizer interface.	122
Mercury.LanguageInfo< T >	
Represents a collection of information required to analyze a language with Mercury Mercury . Mercury . Analyser { <i>T</i> , <i>TValue</i> }.	123
Mercury.Nucleus.Collections.ListMultiDictionary< TKey, TValue >	
An implementation of IMultiDictionary { <i>TKey</i> , <i>TValue</i> } using T:System.Collections.Generic.List { <i>T</i> } as an underlying collection of values.	124
Mercury.Interpreting.MemoEntry< T >	
Represents memoized value	129
Mercury.Exceptions.MercuryException	
An exception thrown by the Mercury library.	130
Mercury.Interpreting.OperationAction< T, U, V >	
Represents an operation on two values.	132
Mercury.Syntax.ParserResult	
Represents the result returned by the parser.	133
Mercury.Interpreting.RewriteRule< T >	
Interpreter rewrite rule	134

Mercury.Interpreting.RewriteRuleCollection< T >	
A read-only collection of rewrite rules	136
Mercury.Interpreting.RewriteRuleCollectionBuilder< T >	
Can be used to construct Mercury.Interpreting.RewriteRuleCollection{T}	137
Mercury.Syntax.Rule	
Represents a Context-Free Grammar rule. This is an immutable class	139
Mercury.Interpreting.RuleEntry< T >	
Describes a rule that has been used to interpret a tree.	141
Mercury.Interpreting.SemanticInterpreter< T >	
Represent the semantic analysis with interpretation. Consists of two interpreters, the first one rewrites parse trees, the second one interprets them. This class serves as a wrapper so it could be used in the Mercury.Analyser{T,TResult} class.	143
Mercury.Nucleus.Collections.SetMultiDictionary< TKey, TValue >	
An implementation of IMultiDictionary{TKey,TValue} using T:System.Collections.Generic.Hash<T>Set{T} as an underlying collection of values.	145
Mercury.Interpreting.Structure	
Represents a tree of Mercury.Interpreting.IElement instances. Must match precisely to the parse tree.	149
Mercury.Syntax.Symbol	
Represents a single symbol of the grammar. This is an immutable class.	
For a symbol name to be valid it must NOT contain control or separator characters, must not be empty and must meet these conditions:	
151	
Mercury.Exceptions.SyntaxException	
An exception thrown when something wrong happens in the Mercury.Syntax	154
Mercury.Interpreting.TextEntry	
Stores informations logged by actions or internal Evaluator.	156
Mercury.Scanning.Token	
Token , a part of an input text recognized by tokenizer	157
Mercury.Scanning.Tokenizer	
A default implementation of ITokenizer interface using Microsoft Regular Expressions.	
Splits tokens in the following fashion:	
160	
Mercury.Nucleus.Trees.Tree< T >	
A non-balanced tree implementation that can be referenced from many other trees. Because of that, there is no way to get back to the parent node	162
Mercury.Interpreting.TreeEntry< T >	
Represents an entry for tree interpretation.	166
Mercury.Nucleus.Collections.UniqueListMultiDictionary< TKey, TValue >	
An implementation of IMultiDictionary{TKey,TValue} using T:System.Collections.Generic.List{T} as underlying collection, but opposed to ListMultiDictionary{TKey,TValue} does not allow duplicate values associated with the same key	167
Mercury.Interpreting.Variable	
Represents a variable element that captures value and can be used on the RHS of the Rewrite←Rule as an argument	169
Mercury.Interpreting.VariableParameter	
Represents a named parameter that will be replaced by an actual value of a variable captured by interpreter.	171
Mercury.Interpreting.Wildcard	
An element that matches zero or more values. Can be used as a variable, but the actioncall must be of arity (0, int.MAX_VALUE) and must accept Tree or Symbol as the argument. . . .	173

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

Mercury/Mercury/ Analyser.cs	175
Mercury/Mercury/ AnalyserResult.cs	175
Mercury/Mercury/ Defaults.cs	176
Mercury/Mercury/ Extensions.cs	177
Mercury/Mercury/ LanguageInfo.cs	183
Mercury/Mercury/Exceptions/ InterpretingExceptions.cs	176
Mercury/Mercury/Exceptions/ MercuryException.cs	176
Mercury/Mercury/Exceptions/ SyntaxExceptions.cs	177
Mercury/Mercury/Interpreting/ ActionCall.cs	177
Mercury/Mercury/Interpreting/ Argument.cs	177
Mercury/Mercury/Interpreting/ BasicActions.cs	178
Mercury/Mercury/Interpreting/ Element.cs	179
Mercury/Mercury/Interpreting/ Entries.cs	179
Mercury/Mercury/Interpreting/ Evaluator.cs	180
Mercury/Mercury/Interpreting/ FormalParameter.cs	180
Mercury/Mercury/Interpreting/ Instance.cs	180
Mercury/Mercury/Interpreting/ Interpreter.cs	181
Mercury/Mercury/Interpreting/ InterpreterAction.cs	181
Mercury/Mercury/Interpreting/ InterpreterContext.cs	182
Mercury/Mercury/Interpreting/ InterpreterResult.cs	182
Mercury/Mercury/Interpreting/ RewriteRule.cs	182
Mercury/Mercury/Interpreting/ RewriteRuleCollection.cs	182
Mercury/Mercury/Nucleus/Collections/ MultiDictionary.cs	183
Mercury/Mercury/Nucleus/Trees/ Tree.cs	183
Mercury/Mercury/Nucleus/Trees/ TreeEnumerators.cs	184
Mercury/Mercury/obj/Debug/ TemporaryGeneratedFile_036C0B5B-1481-4323-8D20-8F5ADCB23D92.cs	184
Mercury/Mercury/obj/Debug/ TemporaryGeneratedFile_5937a670-0e60-4077-877b-f7221da3dda1.cs	184
Mercury/Mercury/obj/Debug/ TemporaryGeneratedFile_E7A71F73-0F8D-4B9B-B56E-8E70B10BC5D3.cs	184
Mercury/Mercury/obj/Release/ TemporaryGeneratedFile_036C0B5B-1481-4323-8D20-8F5ADCB23D92.cs	184
Mercury/Mercury/obj/Release/ TemporaryGeneratedFile_5937a670-0e60-4077-877b-f7221da3dda1.cs	184
Mercury/Mercury/obj/Release/ TemporaryGeneratedFile_E7A71F73-0F8D-4B9B-B56E-8E70B10BC5D3.cs	184
Mercury/Mercury/Properties/ AssemblyInfo.cs	184
Mercury/Mercury/Scanning/ Token.cs	184
Mercury/Mercury/Scanning/ Tokenizer.cs	185
Mercury/Mercury/Syntax/ Edge.cs	185
Mercury/Mercury/Syntax/ EdgeDerivation.cs	185

Mercury/Mercury/Syntax/ Grammar.cs	186
Mercury/Mercury/Syntax/ GrammarBuilder.cs	186
Mercury/Mercury/Syntax/ Parser.cs	186
Mercury/Mercury/Syntax/ ParserResult.cs	187
Mercury/Mercury/Syntax/ Rule.cs	187
Mercury/Mercury/Syntax/ Scanner.cs	187
Mercury/Mercury/Syntax/ Symbol.cs	188

Chapter 5

Namespace Documentation

5.1 Package Mercury

Namespaces

- package [Exceptions](#)
- package [Interpreting](#)
- package [Nucleus](#)
- package [Scanning](#)
- package [Syntax](#)

Classes

- class [Analyser< T >](#)
Wrapper class for each step of analysis
- class [Analyser< T, TResult >](#)
Wrapper class for each step of analysis.
- class [AnalyserResult< T, TResult >](#)
Holds the result of analysis.

The fields hold information according to the following table:
- class [Defaults](#)
Global class containing cool static / convenience stuff.
- class [Extensions](#)
Provides extension methods for the [Mercury](#) library.
- class [LanguageInfo< T >](#)
Represents a collection of information required to analyze a language with [Mercury](#) `Mercury.Analyser{T,TValue}`.

Enumerations

- enum [AnalyserStatus](#) {
[AnalyserStatus.Success](#), [AnalyserStatus.TokenizerFail](#), [AnalyserStatus.ScannerFail](#), [AnalyserStatus.ParserFail](#),
[AnalyserStatus.InterpreterFail](#), [AnalyserStatus.Undefined](#) }
The status of analysis. If failed, see [FailReason](#)
- enum [FailReason](#) { [FailReason.None](#), [FailReason.Exception](#), [FailReason.NoResult](#) }
Specifies the reason of analysis failure

5.1.1 Enumeration Type Documentation

5.1.1.1 enum Mercury.AnalyserStatus

The status of analysis. If failed, see FailReason

Enumerator

Success The analysis was successful and there is at least one result

TokenizerFail Failed to tokenize input

ScannerFail Failed to scan input

ParserFail Parser failed

InterpreterFail Interpreter failed

Undefined Undefined error

Definition at line 15 of file AnalyserResult.cs.

5.1.1.2 enum Mercury.FailReason

Specifies the reason of analysis failure

Enumerator

None No failure

Exception An exception has been thrown

NoResult There was no result in the stage where analysis failed

Definition at line 34 of file AnalyserResult.cs.

5.2 Package Mercury.Exceptions

Classes

- class [ErrorActionException](#)
Exception thrown by basic actions when a problem was encountered.
- class [InterpretingException](#)
An exception thrown because of problems in the [Interpreting](#) namespace.
- class [InvalidActionArgumentException](#)
An exception representing a problem while type checking the [Mercury.Interpreting.RewriteRule{T}](#).
- class [InvalidActionCallException](#)
An exception representing a problem with [Mercury.Interpreting.ActionCall{T}](#)
- class [InvalidActionException](#)
An exception representing a problem with [Mercury.Interpreting.InterpreterAction{T}](#) class.
- class [InvalidEdgeException](#)
An exception thrown when attempting to create an invalid [Mercury.Parser.Edge](#) instance.
- class [InvalidElementException](#)
An exception representing a problem with [Mercury.Interpreting.IElement](#) descendants.
- class [InvalidRewriteRuleException](#)
An exception representing a problem with [Mercury.Interpreting.RewriteRule{T}](#).
- class [InvalidRuleException](#)
An exception thrown when attempting to create an invalid [Mercury.Syntax.Rule](#) instance.

- class [InvalidSymbolException](#)
An exception thrown when attempting to create an invalid [Mercury.Syntax.Symbol](#) instance.
- class [MercuryException](#)
An exception thrown by the [Mercury](#) library.
- class [SyntaxException](#)
An exception thrown when something wrong happens in the [Mercury.Syntax](#)

5.3 Package Mercury.Interpreting

Classes

- class [ActionCall< T >](#)
This class represents the application of [Mercury.Interpreting.InterpreterAction{T}](#) on the [Mercury.Interpreting.l \$\leftrightarrow\$ FormalParameter](#) list.
- class [Alternative](#)
Represents an alternative of elements. It may match specified symbols or symboltypes (Nonterminals, ...).
- class [ArgsCountAction< T >](#)
- class [Argument](#)
Represents an argument passed to the [Mercury.Interpreting.InterpreterAction{T}](#), the actual parameter given to the [Mercury.Interpreting.Evaluator{T}](#) in place of a formal parameter.
- class [ArithmeticAction< T >](#)
Represents the arithmetic action. It is similar to the [Mercury.Interpreting.FoldingAction{T,U,V}](#) but does not require initial value. The minimal number of arguments is therefore 2. For instance, "1 2 3" would be evaluated as $f(f(1, 2), 3)$
- class [BasicActions< T >](#)
Contains basic actions for interpretation.

Type specifiers of actions are defined as follows:
- class [CaseAction< T >](#)
- class [ConcatAction< T >](#)
- class [Constant](#)
[Constant](#) element. Basically equivalent of [Mercury.Interpreting.Alternative](#).
- class [ConstantAction< T, U >](#)
Wraps a value as an action with no parameters
- class [ConstantParameter](#)
[Constant](#) parameter.
- class [ConvertAction< T, U, V >](#)
Applies the lambda function on a single value.
- interface [DataEntry](#)
Entry base (dummy) interface.
- class [EmptyContextFactory](#)
Produces contexts with no custom data
- class [EntryWriter< T >](#)
An auxiliary class that formats the [DataEntry](#) structure and writes it into a stream.
- class [EvaluateAction< T >](#)
- class [Evaluator< T >](#)
Carries out the actual evaluation of [Mercury.Interpreting.InterpreterAction{T}](#) and its arguments arguments.
- class [FailAction< T >](#)
- class [FlattenAction< T >](#)
- class [FoldingAction< T, U, V >](#)

"Folds" all its arguments into one value. For instance, for "x y z" arguments this action computes $f(f(f(s, x), y), z)$ where f is the lambda function and s is the seed parameter of the constructor

- class [GenericAction< T >](#)

Represents an action that infers its return type only after it is provided with types of its arguments.

- class **IdentityAction< T >**

- interface [IElement](#)

Element interface

- class **IfAction< T >**

- interface [IFormalParameter](#)

Represents formal parameters.

- interface [IInterpreter< T >](#)

Interface for interpreters that create objects from derivation trees according to RewriteRules.

- interface [IInterpreterContextFactory](#)

Factory of context.

- class [Instance](#)

Represents a binding between a [Mercury.Interpreting.Variable](#) and its value

- class [InstanceList](#)

Represents the list of instances

- class **InternalContext< T >**

- class [Interpreter< T >](#)

Interprets derivation trees using rewrite rules. The type parameter T represents the target language.

- class [InterpreterAction< T >](#)

Function that wraps constant values in the RHS

- class [InterpreterAction< T, U >](#)

Interpreter action with a safer return type.

- class [InterpreterContext< T >](#)

Interpreter Context class. Some languages may use it to override it and remember data when parse trees are being interpreted.

- class [InterpreterResult< T >](#)

Result of the interpretation.

- class **JoinAction< T >**

- class **LetAction< T >**

- class [MemoEntry< T >](#)

Represents memoized value

- class [OperationAction< T, U, V >](#)

Represents an operation on two values.

- class [RewriteRule< T >](#)

Interpreter rewrite rule

- class [RewriteRuleCollection< T >](#)

A read-only collection of rewrite rules

- class [RewriteRuleCollectionBuilder< T >](#)

Can be used to construct `Mercury.Interpreting.RewriteRuleCollection{T}`

- class **RewriteRuleValidator< T >**

- class [RuleEntry< T >](#)

Describes a rule that has been used to interpret a tree.

- class **SelectAction< T >**

- class [SemanticInterpreter< T >](#)

Represent the semantic analysis with interpretation. Consists of two interpreters, the first one rewrites parse trees, the second one interprets them. This class serves as a wrapper so it could be used in the `Mercury.Analyser{T,TResult}` class.

- class **SimpleAction**< T >
- class [Structure](#)
Represents a tree of [Mercury.Interpreting.IElement](#) instances. Must match precisely to the parse tree.
- class **SubTreeAction**< T >
- class [TextEntry](#)
Stores informations logged by actions or internal Evaluator.
- class **TraceAction**< T >
- class [TreeEntry](#)< T >
Represents an entry for tree interpretation.
- class **TryAction**< T >
- class **VarAction**< T >
- class [Variable](#)
Represents a variable element that captures value and can be used on the RHS of the RewriteRule as an argument
- class [VariableParameter](#)
Represents a named parameter that will be replaced by an actual value of a variable captured by interpreter.
- class [Wildcard](#)
An element that matches zero or more values. Can be used as a variable, but the actioncall must be of arity (0, `int.MAX_VALUE`) and must accept *Tree* or *Symbol* as the argument.

Enumerations

- enum [ArgumentType](#) {
[ArgumentType.None](#) = 0, [ArgumentType.Integer](#) = 1, [ArgumentType.Real](#) = 2, [ArgumentType.Boolean](#) = 4,
[ArgumentType.String](#) = 8, [ArgumentType.Symbol](#) = 16, [ArgumentType.Tree](#) = 32, [ArgumentType.TValue](#) = 64,
[ArgumentType.Primitive](#) = Integer | Real | String | Boolean, [ArgumentType.ParserEntity](#) = Symbol | Tree,
[ArgumentType.Any](#) = Primitive | ParserEntity | TValue, [ArgumentType.Null](#) = 128,
[ArgumentType.Lazy](#) = 256 | Null, [ArgumentType.Flags](#) = Null | Lazy }
Defines argument types using in the [Mercury](#) library
- enum [ElementType](#) { [ElementType.Constant](#), [ElementType.Variable](#), [ElementType.Wildcard](#), [ElementType.LStructure](#) }
Element Type enumeration
- enum [ParameterType](#) {
[ParameterType.IntegralConstant](#), [ParameterType.RealConstant](#), [ParameterType.StringConstant](#), [ParameterType.BooleanConstant](#),
[ParameterType.Variable](#), [ParameterType.ActionCall](#) }
Formal Parameter Type

Functions

- delegate T [RecursiveEval](#)< T > ([Tree](#)< [Symbol](#) > tree, InterpreterContext< T > context)
Delegate type for recursive evaluation (interpretation)

5.3.1 Enumeration Type Documentation

5.3.1.1 enum Mercury.Interpreting.ArgumentType

Defines argument types using in the [Mercury](#) library

Enumerator

- None** Invalid argument type
- Integer** Integer value

Real Real value
Boolean Boolean value
String String value
Symbol A single symbol
Tree A derivation tree
TValue A value of the type passed to the interpreter
Primitive Primitive data types
ParserEntity Grammar entity
Any Any argument
Null Accepts null value. Must be used with other types to have effect
Lazy Lazy evaluation flag. Must be used with other types to have effect
Flags All flags

Definition at line 11 of file Argument.cs.

5.3.1.2 enum Mercury.Interpreting.ElementType

Element Type enumeration

Enumerator

Constant [Constant](#) - must match precisely
Variable [Variable](#) - must match type and carries value to the RHS
Wildcard [Wildcard](#) - an arbitrary number of matches
Structure [Structure](#) - subtree (recursive) match

Definition at line 15 of file Element.cs.

5.3.1.3 enum Mercury.Interpreting.ParameterType

Formal Parameter Type

Enumerator

IntegralConstant Integral constant
RealConstant Real (double) constant
StringConstant String constant
BooleanConstant Boolean constant
Variable [Variable](#)
ActionCall Action call result

Definition at line 11 of file FormalParameter.cs.

5.3.2 Function Documentation

5.3.2.1 `delegate T Mercury.Interpreting.RecursiveEval< T > ( Tree< Symbol > tree, InterpreterContext< T > context )`

Delegate type for recursive evaluation (interpretation)

Template Parameters

<i>T</i>	Interpreter output type
----------	-------------------------

Parameters

<i>tree</i>	The tree to interpret
<i>context</i>	Interpreter context (see <code>Mercury.Interpreting.InterpreterContext{T}</code>).

Returns

The result of interpreter or `null` if failed

Type Constraints

***T* : class**

5.4 Package Mercury.Nucleus

Namespaces

- package [Collections](#)
- package [Trees](#)

5.5 Package Mercury.Nucleus.Collections

Classes

- interface [IMultiDictionary< TKey, TValue >](#)
Represents a multi-valued dictionary, in other words a map between a key and a collection of values.
- interface [IReadOnlyMultiDictionary< TKey, TValue >](#)
Represents a multi-valued read-only dictionary, a map between a key and a collection of values.
- class [ListMultiDictionary< TKey, TValue >](#)
An implementation of `IMultiDictionary{TKey,TValue}` using `T:System.Collections.Generic.List{T}` as an underlying collection of values.
- class [SetMultiDictionary< TKey, TValue >](#)
An implementation of `IMultiDictionary{TKey,TValue}` using `T:System.Collections.Generic.HashSet{T}` as an underlying collection of values.
- class [UniqueListMultiDictionary< TKey, TValue >](#)
An implementation of `IMultiDictionary{TKey,TValue}` using `T:System.Collections.Generic.List{T}` as underlying collection, but opposed to `ListMultiDictionary{TKey,TValue}` does not allow duplicate values associated with the same key

5.6 Package Mercury.Nucleus.Trees

Classes

- class [BFSTreeEnumerator< T >](#)
Breadth-First Search tree enumerator
- class [DFSTreeEnumerator< T >](#)
Depth-First Search tree enumerator
- class [Tree< T >](#)
A non-balanced tree implementation that can be referenced from many other trees. Because of that, there is no way to get back to the parent node.

5.7 Package Mercury.Scanning

Classes

- interface [ITokenizer](#)
The tokenizer interface.
- class [Token](#)
[Token](#), a part of an input text recognized by tokenizer
- class [Tokenizer](#)
A default implementation of [ITokenizer](#) interface using Microsoft Regular Expressions.

Splits tokens in the following fashion:

Enumerations

- enum [TokenType](#) { [TokenType.String](#), [TokenType.Symbol](#), [TokenType.Keyword](#), [TokenType.Number](#) }
Defines token types. The actual conditions depend on the [Mercury.Scanning.ITokenizer](#) implementation.
- enum [TokenizerOptions](#) {
[TokenizerOptions.None](#) = 0, [TokenizerOptions.CompiledMatch](#) = 1, [TokenizerOptions.IgnoreNumbers](#) = 2,
[TokenizerOptions.IgnoreCase](#) = 4,
[TokenizerOptions.IgnoreQuotes](#) = 8, [TokenizerOptions.IgnoreSigns](#) = 16, [TokenizerOptions.StrictNumbers](#) = 32 }
Options for the tokenizer

5.7.1 Enumeration Type Documentation

5.7.1.1 enum Mercury.Scanning.TokenizerOptions

Options for the tokenizer

Enumerator

None summary>No optionssummary>Compile regex pattern, which yields faster matching at the cost of slower initialization.

CompiledMatch summary>Will not detect numbers, but consider them strings instead.

IgnoreNumbers summary>Ignores case when matching patterns.

IgnoreCase summary>Treats quote characters as symbols.

IgnoreQuotes summary>Detects only positive numbers.

IgnoreSigns summary>With this option, numbers must be separated by spaces.

StrictNumbers

Definition at line 26 of file Tokenizer.cs.

5.7.1.2 enum Mercury.Scanning.TokenType

Defines token types. The actual conditions depend on the [Mercury.Scanning.ITokenizer](#) implementation.

Enumerator

String summary>String.summary>Symbol.

Symbol summary>Keyword (a reserved word).

Keyword summary>Number.

Number

Definition at line 12 of file Token.cs.

5.8 Package Mercury.Syntax

Classes

- class [BasicScanner](#)
Generates four Nonterminals @Symbol, @Keyword, @Number and @String based on the given map and types of input tokens.
- class [BasicScannerMap](#)
Contains mapping between TokenType and [Mercury.Syntax.Mapping](#) of [Mercury.Syntax.BasicScanner](#).
- class [ChartParser](#)
Thread-safe wrapper for ChartParserInternal class.
- class **ChartParserInternal**
The Bottom-Up Chart Parser implementation.
- class [CompositeScanner](#)
[Mercury.Syntax.IScanner](#) implementation that behaves like container of scanners using the Composite Design Pattern.
- class [Edge](#)
Represents an edge in the chart created by the [ChartParser](#). This is an immutable class. The [Edge](#) contains four values:
- class **EdgeDerivation**
[Edge](#) Derivation
- class **ExtendedGrammar**
Represents a Context-Free [Grammar](#) with convenience structures for [ChartParser](#).
- class **ExtendedGrammarBuilder**
- class [Grammar](#)
Represents a Context-Free [Grammar](#). This is an immutable class, to create a grammar use [Mercury.Syntax.<GrammarBuilder](#).
- class [GrammarBuilder](#)
Class that creates grammars.
- interface [IParser](#)
Creates a derivation tree from the list of input tokens
- interface [IScanner](#)
Scanner scans input tokens and creates additional rules for parsing.
- class [ParserResult](#)
Represents the result returned by the parser.
- class [Rule](#)
Represents a Context-Free [Grammar](#) rule. This is an immutable class.
- class [Symbol](#)
Represents a single symbol of the grammar. This is an immutable class.

For a symbol name to be valid it must NOT contain control or separator characters, must not be empty and must meet these conditions:

Enumerations

- enum `Mapping` {
`Mapping.None` = 0, `Mapping.AsSymbol` = 1, `Mapping.AsKeyword` = 2, `Mapping.AsNumber` = 4,
`Mapping.AsString` = 8, `Mapping.All` = 15 }
- enum `SymbolType` {
`SymbolType.None` = 0, `SymbolType.Epsilon` = 1, `SymbolType.Terminal` = 2, `SymbolType.Nonterminal` = 4,
`SymbolType.Any` = Epsilon | Terminal | Nonterminal }

Represents a type of symbol. Implemented as Flags since Symbol type inference is ambiguous.

5.8.1 Enumeration Type Documentation

5.8.1.1 enum `Mercury.Syntax.Mapping`

Enumerator

- None** Does not map to anything
- AsSymbol** Maps token to symbol (@Symbol -> token)
- AsKeyword** Maps token to keyword (@Keyword -> token)
- AsNumber** Maps token to number (@Number -> token)
- AsString** Maps token to string (@String -> token)
- All** Maps token to symbol, keyword, number and string

Definition at line 113 of file Scanner.cs.

5.8.1.2 enum `Mercury.Syntax.SymbolType`

Represents a type of symbol. Implemented as Flags since `Symbol` type inference is ambiguous.

Enumerator

- None** An invalid symbol (cannot be used in a grammar).
- Epsilon** The epsilon symbol.
- Terminal** A nonterminal symbol.
- Nonterminal** A terminal symbol.
- Any** Any symbol.

Definition at line 16 of file Symbol.cs.

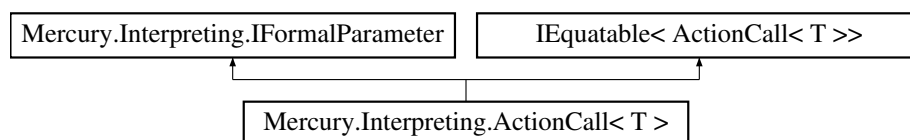
Chapter 6

Class Documentation

6.1 Mercury.Interpreting.ActionCall< T > Class Template Reference

This class represents the application of Mercury.Interpreting.InterpreterAction{T} on the [Mercury.Interpreting.IFormalParameter](#) list.

Inheritance diagram for Mercury.Interpreting.ActionCall< T >:



Public Member Functions

- [ActionCall](#) (InterpreterAction< T > action, IEnumerable< [IFormalParameter](#) > formalparams)
Initializes a new instance of the ActionCall{T} class.
- override bool [Equals](#) (object obj)
- override int [GetHashCode](#) ()
- override string [ToString](#) ()
- bool [Equals](#) ([ActionCall](#)< T > other)

Properties

- IReadOnlyList< [IFormalParameter](#) > [FormalParameters](#) [get, set]
The list of formal parameters.
- InterpreterAction< T > [Action](#) [get, set]
The action
- [ParameterType](#) Type [get, set]

6.1.1 Detailed Description

This class represents the application of Mercury.Interpreting.InterpreterAction{T} on the [Mercury.Interpreting.IFormalParameter](#) list.

Type Constraints

T : *class*

Definition at line 18 of file ActionCall.cs.

6.1.2 Constructor & Destructor Documentation

6.1.2.1 Mercury.Interpreting.ActionCall<T>.ActionCall (InterpreterAction<T> *action*, IEnumerable< IFormalParameter > *formalparams*)

Initializes a new instance of the ActionCall{T} class.

Parameters

<i>action</i>	The action.
<i>formalparams</i>	Formal parameters.

Exceptions

<i>System.ArgumentNull↵ Exception</i>	When action or formal parameters is null
<i>InvalidActionCall↵ Exception{T}</i>	Invalid number of formal parameters

Definition at line 34 of file ActionCall.cs.

6.1.3 Member Function Documentation

6.1.3.1 override bool Mercury.Interpreting.ActionCall<T>.Equals (object *obj*)

Definition at line 61 of file ActionCall.cs.

6.1.3.2 bool Mercury.Interpreting.ActionCall<T>.Equals (ActionCall<T> *other*)

Definition at line 107 of file ActionCall.cs.

6.1.3.3 override int Mercury.Interpreting.ActionCall<T>.GetHashCode ()

Definition at line 67 of file ActionCall.cs.

6.1.3.4 override string Mercury.Interpreting.ActionCall<T>.ToString ()

Definition at line 80 of file ActionCall.cs.

6.1.4 Property Documentation

6.1.4.1 InterpreterAction<T> Mercury.Interpreting.ActionCall<T>.Action [get], [set]

The action

The action

Definition at line 55 of file ActionCall.cs.

6.1.4.2 IReadOnlyList<IFormalParameter> Mercury.Interpreting.ActionCall<T>.FormalParameters [get], [set]

The list of formal parameters.

The list of formal parameters.

Definition at line 51 of file ActionCall.cs.

6.1.4.3 ParameterType Mercury.Interpreting.ActionCall< T >.Type [get], [set]

Definition at line 101 of file ActionCall.cs.

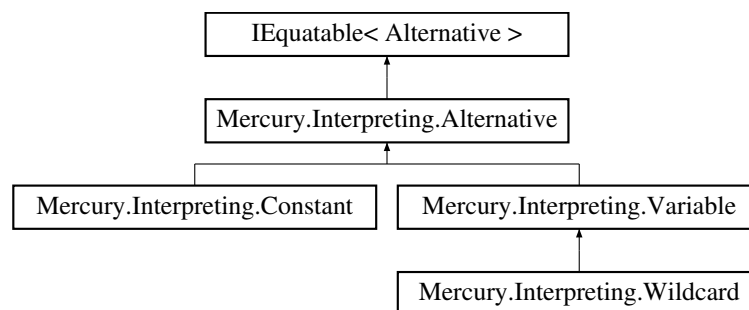
The documentation for this class was generated from the following file:

- Mercury/Mercury/Interpreting/[ActionCall.cs](#)

6.2 Mercury.Interpreting.Alternative Class Reference

Represents an alternative of elements. It may match specified symbols or symboltypes (Nonterminals, ...).

Inheritance diagram for Mercury.Interpreting.Alternative:



Public Member Functions

- **Alternative** ([SymbolType](#) sm, IEnumerable< [Symbol](#) > symbols)
Initializes a new instance of the [Alternative](#) class. If symbols contains symbols of type that is present in the sm mask, they will be omitted.
- bool **MatchesSymbol** ([Symbol](#) s)
Indicates whether this alternative matches the given symbol.
- override bool **Equals** (object obj)
- override int **GetHashCode** ()
- override string **ToString** ()
- bool **Equals** ([Alternative](#) other)

Protected Attributes

- HashSet< [Symbol](#) > **smset**

Properties

- [SymbolType](#) **SymbolMask** [get, set]
Mask of symbol types this alternative matches regardless the symbol name.
- IReadOnlyList< [Symbol](#) > **Symbols** [get, set]
List of specific symbols this alternative matches.

6.2.1 Detailed Description

Represents an alternative of elements. It may match specified symbols or symboltypes (Nonterminals, ...).

Definition at line 48 of file Element.cs.

6.2.2 Constructor & Destructor Documentation

6.2.2.1 Mercury.Interpreting.Alternative.Alternative (**SymbolType** *sm*, IEnumerable< **Symbol** > *symbols*)

Initializes a new instance of the [Alternative](#) class. If *symbols* contains symbols of type that is present in the *sm* mask, they will be omitted.

Parameters

<i>sm</i>	The Symbol mask.
<i>symbols</i>	The symbols.

Exceptions

<i>System.ArgumentNullException</i>	When symbols is null
<i>System.ArgumentException</i>	Argument 'symbols' contains null
<i>System.ArgumentException</i>	Empty alternative is not allowed

Definition at line 67 of file Element.cs.

6.2.3 Member Function Documentation

6.2.3.1 override bool Mercury.Interpreting.Alternative.Equals (object *obj*)

Definition at line 112 of file Element.cs.

6.2.3.2 bool Mercury.Interpreting.Alternative.Equals (**Alternative** *other*)

Definition at line 159 of file Element.cs.

6.2.3.3 override int Mercury.Interpreting.Alternative.GetHashCode ()

Definition at line 118 of file Element.cs.

6.2.3.4 bool Mercury.Interpreting.Alternative.MatchesSymbol (**Symbol** *s*)

Indicates whether this alternative matches the given symbol.

Parameters

<i>s</i>	The symbol.
----------	-------------

Returns

`true` if this alternative matches *s*, `false` otherwise

Definition at line 105 of file Element.cs.

6.2.3.5 override string Mercury.Interpreting.Alternative.ToString ()

Definition at line 130 of file Element.cs.

6.2.4 Member Data Documentation

6.2.4.1 HashSet<Symbol> Mercury.Interpreting.Alternative.smset [protected]

Definition at line 53 of file Element.cs.

6.2.5 Property Documentation

6.2.5.1 SymbolType Mercury.Interpreting.Alternative.SymbolMask [get], [set]

Mask of symbol types this alternative matches regardless the symbol name.

The symbol mask.

Definition at line 92 of file Element.cs.

6.2.5.2 IReadOnlyList<Symbol> Mercury.Interpreting.Alternative.Symbols [get], [set]

List of specific symbols this alternative matches.

The symbols.

Definition at line 96 of file Element.cs.

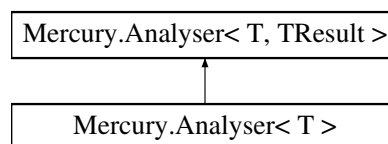
The documentation for this class was generated from the following file:

- Mercury/Mercury/Interpreting/[Element.cs](#)

6.3 Mercury.Analyser< T > Class Template Reference

Wrapper class for each step of analysis

Inheritance diagram for Mercury.Analyser< T >:



Public Member Functions

- [Analyser](#) (LanguageInfo< T > langinfo, [InterpreterContextFactory](#) contextFactory=null)
Initializes a new instance of the Analyser{T} class.
- [Analyser](#) (ITokenizer tokenizer, IScanner scanner, IParser parser, IInterpreter< T > interpreter)
Initializes a new instance of the Analyser{T} class. See Mercury.Analyser{T,TResult} for more information.

Additional Inherited Members

6.3.1 Detailed Description

Wrapper class for each step of analysis

Template Parameters

<i>T</i>	Interpreter result type
----------	-------------------------

Type Constraints

***T* : class**

Definition at line 195 of file Analyser.cs.

6.3.2 Constructor & Destructor Documentation

6.3.2.1 `Mercury.Analyser< T >.Analyser ( LanguageInfo< T > langinfo, IInterpreterContextFactory contextFactory = null )`

Initializes a new instance of the Analyser{T} class.

Parameters

<i>langinfo</i>	The language information.
<i>contextFactory</i>	The context factory. Can be <code>null</code> .

Definition at line 202 of file Analyser.cs.

6.3.2.2 `Mercury.Analyser< T >.Analyser ( ITokenizer tokenizer, IScanner scanner, IParser parser, IInterpreter< T > interpreter )`

Initializes a new instance of the Analyser{T} class. See Mercury.Analyser{T, TResult} for more information.

Parameters

<i>tokenizer</i>	The tokenizer.
<i>scanner</i>	The scanner.
<i>parser</i>	The parser.
<i>interpreter</i>	The interpreter.

Definition at line 214 of file Analyser.cs.

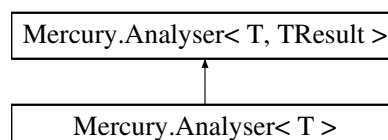
The documentation for this class was generated from the following file:

- Mercury/Mercury/[Analyser.cs](#)

6.4 Mercury.Analyser< T, TResult > Class Template Reference

Wrapper class for each step of analysis.

Inheritance diagram for Mercury.Analyser< T, TResult >:



Public Member Functions

- [Analyser](#) (LanguageInfo< T > *langinfo*, Func< T, TResult > *resultSelector*, [IInterpreterContextFactory](#) *contextFactory*=null)

Initializes a new instance of the `Analyser{T,TResult}` class from the provided language information.

- **Analyser** (**ITokenizer** tokenizer, **IScanner** scanner, **IParser** parser, **Interpreter**< T > interpreter, **Func**< T, TResult > resultSelector)

Initializes a new instance of analyzer from the class. Scanner and Interpreter can be `null` (if so, they won't be used). resultSelector can be `null` only if the interpreter is `null`.

- **AnalyserResult**< T, TResult > **Analyze** (string input)

Analyzes the specified input and returns the `Mercury.AnalyserResult{T,TResult}`.

Properties

- **ITokenizer Tokenizer** [get, set]
Gets the tokenizer.
- **IScanner Scanner** [get, set]
Gets the scanner.
- **IParser Parser** [get, set]
Gets the parser.
- **Interpreter**< T > **Interpreter** [get, set]
Gets the interpreter.

6.4.1 Detailed Description

Wrapper class for each step of analysis.

Template Parameters

<i>T</i>	Interpreter return type.
<i>TResult</i>	Result type of post-processing. May equal <i>T</i> .

Type Constraints

T : class

TResult : class

Definition at line 18 of file `Analyser.cs`.

6.4.2 Constructor & Destructor Documentation

6.4.2.1 `Mercury.Analyser< T, TResult >.Analyser ( LanguageInfo< T > langinfo, Func< T, TResult > resultSelector, InterpreterContextFactory contextFactory = null )`

Initializes a new instance of the `Analyser{T,TResult}` class from the provided language information.

Parameters

<i>langinfo</i>	The language information.
<i>resultSelector</i>	The post-processing function.
<i>contextFactory</i>	The context factory.

Definition at line 33 of file `Analyser.cs`.

6.4.2.2 `Mercury.Analyser< T, TResult >.Analyser ( ITokenizer tokenizer, IScanner scanner, IParser parser, Interpreter< T > interpreter, Func< T, TResult > resultSelector )`

Initializes a new instance of analyzer from the class. Scanner and Interpreter can be `null` (if so, they won't be used). resultSelector can be `null` only if the interpreter is `null`.

Parameters

<i>tokenizer</i>	The tokenizer.
<i>scanner</i>	The scanner.
<i>parser</i>	The parser.
<i>interpreter</i>	The interpreter.
<i>resultSelector</i>	The result selector.

Exceptions

<i>System.ArgumentNull</i> ↔ <i>Exception</i>	tokenizer, parser or resultSelector is null
--	---

Definition at line 52 of file Analyser.cs.

6.4.3 Member Function Documentation

6.4.3.1 AnalyserResult<T, TResult> Mercury.Analyser< T, TResult >.Analyze (string input)

Analyzes the specified input and returns the Mercury.AnalyserResult{T,TResult}.

Parameters

<i>input</i>	The input.
--------------	------------

Returns

The result of analysis.

Definition at line 90 of file Analyser.cs.

6.4.4 Property Documentation

6.4.4.1 IInterpreter<T> Mercury.Analyser< T, TResult >.Interpreter [get], [set]

Gets the interpreter.

Definition at line 81 of file Analyser.cs.

6.4.4.2 IParser Mercury.Analyser< T, TResult >.Parser [get], [set]

Gets the parser.

Definition at line 78 of file Analyser.cs.

6.4.4.3 IScanner Mercury.Analyser< T, TResult >.Scanner [get], [set]

Gets the scanner.

Definition at line 75 of file Analyser.cs.

6.4.4.4 ITokenizer Mercury.Analyser< T, TResult >.Tokenizer [get], [set]

Gets the tokenizer.

Definition at line 72 of file Analyser.cs.

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

- Mercury/Mercury/[Analyser.cs](#)

6.5 Mercury.AnalyserResult< T, TResult > Class Template Reference

Holds the result of analysis.

The fields hold information according to the following table:

Properties

- [AnalyserStatus Status](#) [get, set]
Status of the analysis
- [FailReason FailReason](#) [get, set]
Reason of the failure. If the value is `Exception`, `Exception` field must contain the exception that has been thrown.
- [ParserResult ParserResult](#) [get, set]
Parser result
- [IReadOnlyList< InterpreterResult< T > > InterpreterResults](#) [get, set]
The list of interpreter results for each tree in `ParserResult.Trees`.
- [IReadOnlyList< TResult > Interpretations](#) [get, set]
Final interpretations
- [Exception Exception](#) [get, set]
If the `FailReason` is `Exception`, this field contains the exception that has been thrown
- [TimeSpan ParserTime](#) [get, set]
The time it took to parse input
- [TimeSpan InterpreterTime](#) [get, set]
The time it took to parse all trees
- [TimeSpan TotalTime](#) [get, set]
Total time of analysis. Includes `ParserTime`, `InterpreterTime` and additional duration of analyser overhead

6.5.1 Detailed Description

Holds the result of analysis.

The fields hold information according to the following table:

-- Conditions --			-- Field values --		
Status	FailReason		ParserR.	Interp's	Exception
Success	None		[yes]	[yes*]	undef.
*Fail	NoValidResult		undef.	undef.	undef.
*Fail	Exception		undef.	undef.	[yes]

[yes*] - provided that `Interpreter` is not `null`

Template Parameters

<i>T</i>	Result type
----------	-------------

Type Constraints

T* : class**TResult* : class**

Definition at line 67 of file AnalyserResult.cs.

6.5.2 Property Documentation

6.5.2.1 Exception Mercury.AnalyserResult< T, TResult >.Exception [get], [set]

If the FailReason is Exception, this field contains the exception that has been thrown

Definition at line 118 of file AnalyserResult.cs.

6.5.2.2 FailReason Mercury.AnalyserResult< T, TResult >.FailReason [get], [set]

Reason of the failure. If the value is Exception, Exception field must contain the exception that has been thrown.

Definition at line 106 of file AnalyserResult.cs.

6.5.2.3 IReadOnlyList<TResult> Mercury.AnalyserResult< T, TResult >.Interpretations [get], [set]

Final interpretations

Definition at line 115 of file AnalyserResult.cs.

6.5.2.4 IReadOnlyList<InterpreterResult<T> > Mercury.AnalyserResult< T, TResult >.InterpreterResults [get], [set]

The list of interpreter results for each tree in ParserResult.Trees.

Definition at line 112 of file AnalyserResult.cs.

6.5.2.5 TimeSpan Mercury.AnalyserResult< T, TResult >.InterpreterTime [get], [set]

The time it took to parse all trees

Definition at line 124 of file AnalyserResult.cs.

6.5.2.6 ParserResult Mercury.AnalyserResult< T, TResult >.ParserResult [get], [set]

Parser result

Definition at line 109 of file AnalyserResult.cs.

6.5.2.7 TimeSpan Mercury.AnalyserResult< T, TResult >.ParserTime [get], [set]

The time it took to parse input

Definition at line 121 of file AnalyserResult.cs.

6.5.2.8 **AnalyserStatus** `Mercury.AnalyserResult< T, TResult >.Status` `[get], [set]`

Status of the analysis

Definition at line 100 of file `AnalyserResult.cs`.

6.5.2.9 **TimeSpan** `Mercury.AnalyserResult< T, TResult >.TotalTime` `[get], [set]`

Total time of analysis. Includes `ParserTime`, `InterpreterTime` and additional duration of analyser overhead

Definition at line 130 of file `AnalyserResult.cs`.

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

- `Mercury/Mercury/AnalyserResult.cs`

6.6 **Mercury.Interpreting.Argument** Class Reference

Represents an argument passed to the `Mercury.Interpreting.InterpreterAction{T}`, the actual parameter given to the `Mercury.Interpreting.Evaluator{T}` in place of a formal parameter.

Inherited by `Mercury.Interpreting.Evaluator< T >`.

Properties

- virtual `ArgumentType Type` `[get, set]`
Argument type.
- dynamic `Value` `[get]`
Argument value.

6.6.1 Detailed Description

Represents an argument passed to the `Mercury.Interpreting.InterpreterAction{T}`, the actual parameter given to the `Mercury.Interpreting.Evaluator{T}` in place of a formal parameter.

Definition at line 58 of file `Argument.cs`.

6.6.2 Property Documentation

6.6.2.1 `virtual ArgumentType Mercury.Interpreting.Argument.Type` `[get], [set]`

Argument type.

Definition at line 73 of file `Argument.cs`.

6.6.2.2 `dynamic Mercury.Interpreting.Argument.Value` `[get]`

Argument value.

Definition at line 76 of file `Argument.cs`.

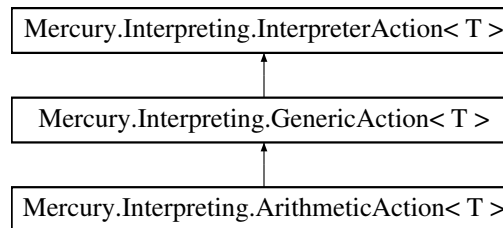
The documentation for this class was generated from the following file:

- `Mercury/Mercury/Interpreting/Argument.cs`

6.7 Mercury.Interpreting.ArithmeticAction< T > Class Template Reference

Represents the arithmetic action. It is similar to the Mercury.Interpreting.FoldingAction{T,U,V} but does not require initial value. The minimal number of arguments is therefore 2. For instance, "1 2 3" would be evaluated as $f(f(1, 2), 3)$

Inheritance diagram for Mercury.Interpreting.ArithmeticAction< T >:



Public Member Functions

- [ArithmeticAction](#) (string name, Func< dynamic, dynamic, dynamic > f)
Initializes a new instance of the ArithmeticAction{T} class.
- override [ArgumentType InferReturnType](#) ([ArgumentType](#) expected, [ArgumentType\[\]](#) actual)
Infers the actual return type from provided parameter types
- override object [Invoke](#) ([Argument\[\]](#) arguments, RecursiveEval< T > eval, InterpreterContext< T > context)
Evaluates the arguments.
It is guaranteed to have a valid number of arguments (specified by [Arity](#)) and that no argument is null UNLESS the flag [Mercury.Interpreting.ArgumentType](#) contains Lazy or Null.

Additional Inherited Members

6.7.1 Detailed Description

Represents the arithmetic action. It is similar to the Mercury.Interpreting.FoldingAction{T,U,V} but does not require initial value. The minimal number of arguments is therefore 2. For instance, "1 2 3" would be evaluated as $f(f(1, 2), 3)$

Template Parameters

<i>T</i>	
----------	--

Type Constraints

***T* : class**

Definition at line 539 of file BasicActions.cs.

6.7.2 Constructor & Destructor Documentation

6.7.2.1 Mercury.Interpreting.ArithmeticAction< T >.ArithmeticAction (string name, Func< dynamic, dynamic, dynamic > f)

Initializes a new instance of the ArithmeticAction{T} class.

Parameters

<i>name</i>	The name of the action
<i>f</i>	The lambda function

Exceptions

<i>System.ArgumentNull</i> <i>Exception</i>	When <i>f</i> is null
--	-----------------------

Definition at line 552 of file BasicActions.cs.

6.7.3 Member Function Documentation

6.7.3.1 **override** `ArgumentType Mercury.Interpreting.ArithmeticAction< T >.InferReturnType ( ArgumentType expected, ArgumentType[] actual )` [virtual]

Infers the actual return type from provided parameter types

Parameters

<i>expected</i>	Expected return type flags.
<i>actual</i>	Actual argument types.

Returns

The actual return type. Must contain exactly ONE bit set, unless it is `Integer` | `Real`

Implements [Mercury.Interpreting.GenericAction< T >](#).

Definition at line 563 of file BasicActions.cs.

6.7.3.2 **override object** `Mercury.Interpreting.ArithmeticAction< T >.Invoke ( Argument[] arguments, RecursiveEval< T > eval, InterpreterContext< T > context )` [virtual]

Evaluates the arguments.

It is guaranteed to have a valid number of arguments (specified by [Arity](#)) and that no argument is `null` UNLESS the flag [Mercury.Interpreting.ArgumentType](#) contains `Lazy` or `Null`.

Parameters

<i>arguments</i>	Arguments of the function.
<i>eval</i>	The recursive evaluation of trees.
<i>context</i>	Interpreter context.

Returns

String value of evaluation or `null` if failed.

Implements [Mercury.Interpreting.InterpreterAction< T >](#).

Definition at line 570 of file BasicActions.cs.

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

- [Mercury/Mercury/Interpreting/BasicActions.cs](#)

6.8 Mercury.Interpreting.BasicActions< T > Class Template Reference

Contains basic actions for interpretation.

Type specifiers of actions are defined as follows:

Static Public Member Functions

- static IReadOnlyList
 < InterpreterAction< T > > [GetAllActions](#) ()
Provides basic [Mercury](#) actions
- static IReadOnlyDictionary
 < string, string > [GetAliases](#) ()
Returns aliases for basic actions.

It contains the following aliases:
- static InterpreterAction< T > [CreateTracer](#) (string name, TextWriter writer)
Creates custom trace action with stream, useful for logging.

Static Public Attributes

- static readonly
 InterpreterAction< T > [ParseInteger](#)
Converts a string representation of an integer.
ParseInteger :: String -> Integer
- static readonly
 InterpreterAction< T > [ParseReal](#)
Converts a string representation of a decimal number.
ParseReal :: String -> Real
- static readonly
 InterpreterAction< T > [ParseBoolean](#)
Converts a string representation of a boolean.
ParseBoolean :: String -> Boolean
- static readonly
 InterpreterAction< T > [Round](#) = new ConvertAction<T, double, int>("Round", x => Convert.ToInt32(Math.Round(x)))
Rounds a real number.
Round :: Real -> Integer
- static readonly
 InterpreterAction< T > [Ceiling](#) = new ConvertAction<T, double, int>("Ceiling", x => Convert.ToInt32(Math.Ceiling(x)))
Returns a ceiling of a real number.
Ceiling :: Real -> Integer
- static readonly
 InterpreterAction< T > [Floor](#) = new ConvertAction<T, double, int>("Floor", x => Convert.ToInt32(Math.Floor(x)))
Returns a floor of a real number.
Floor :: Real -> Integer
- static readonly
 InterpreterAction< T > [ToReal](#) = new ConvertAction<T, int, double>("ToReal", x => Convert.ToDouble(x))
Converts an integer value to decimal.
ToReal :: Integer -> Real

- static readonly
InterpreterAction< T > **String** = new ConvertAction<T, object, string>("String", x => x.ToString())
Call ToString() method on its argument.
String :: Any -> String
- static readonly
InterpreterAction< T > **Add** = new ArithmeticAction<T>("Add", (x, y) => x + y)
Sum of the arguments.
Add[a = Integer|Real] :: a -> a -> ... -> a
- static readonly
InterpreterAction< T > **Sub** = new ArithmeticAction<T>("Sub", (x, y) => x - y)
Subtracts the arguments.
Sub[a = Integer|Real] :: a -> a -> ... -> a
- static readonly
InterpreterAction< T > **Mul** = new ArithmeticAction<T>("Mul", (x, y) => x * y)
Multiplies the arguments.
Mul[a = Integer|Real] :: a -> a -> ... -> a
- static readonly
InterpreterAction< T > **Div** = new ArithmeticAction<T>("Div", (x, y) => x / y)
Divides the values.
Div[a = Integer|Real] :: a -> a -> ... -> a
- static readonly
InterpreterAction< T > **Pow** = new ArithmeticAction<T>("Pow", (x, y) => Math.Pow(x, y))
Exponentiates the values.
Pow[a = Integer|Real] :: a -> a -> ... -> a
- static readonly
InterpreterAction< T > **FDiv** = new OperationAction<T, double, double>("FDiv", (x, y) => x / y)
Floating point division.
FDiv :: Real -> Real -> Real
- static readonly
InterpreterAction< T > **Mod** = new OperationAction<T, int, int>("Mod", (x, y) => x % y)
Modulo.
Mod :: Integer -> Integer -> Integer
- static readonly
InterpreterAction< T > **IsNull** = new ConvertAction<T, object, bool>("IsNull", x => x == null, ArgumentType.Null)
Indicates whether the object is null.
IsNull :: Any -> Boolean
- static readonly
InterpreterAction< T > **And** = new FoldingAction<T, bool, bool>("And", 2, (x, y) => x && y, true)
Logical multiplication.
And :: Boolean -> Boolean -> ... -> Boolean
- static readonly
InterpreterAction< T > **Or** = new FoldingAction<T, bool, bool>("Or", 2, (x, y) => x && y, false)
Logical addition.
Or :: Boolean -> Boolean -> ... -> Boolean
- static readonly
InterpreterAction< T > **Not** = new ConvertAction<T, bool, bool>("Not", x => !x)
Logical negation.
Not :: Boolean -> Boolean
- static readonly
InterpreterAction< T > **Xor** = new OperationAction<T, bool, bool>("Xor", (x, y) => x ^ y)
Exclusive disjunction.
Xor :: Boolean -> Boolean -> Boolean
- static readonly
InterpreterAction< T > **Eq** = new OperationAction<T, object, bool>("Eq", (x, y) => (x != null) && x.Equals(y), ArgumentType.Null)

Indicates whether provided objects are equal. If any of arguments is null, the result is always false.

Eq :: Any -> Any -> Boolean

- static readonly

InterpreterAction< T > **Neq** = new OperationAction<T, object, bool>("Neq", (x, y) => (x == null) || !x.Equals(y), ArgumentType.Null)

Complement of Eq.

Neq :: Any -> Any -> Boolean

- static readonly

InterpreterAction< T > **Gr** = new OperationAction<T, double, bool>("Gr", (x, y) => x > y)

Comparison "Greater than".

Gr :: Real -> Real -> Boolean

- static readonly

InterpreterAction< T > **Ls** = new OperationAction<T, double, bool>("Ls", (x, y) => x < y)

Comparison "Less than".

Ls :: Real -> Real -> Boolean

- static readonly

InterpreterAction< T > **Geq** = new OperationAction<T, double, bool>("Geq", (x, y) => x >= y)

Comparison "Greater than or equal to".

Geq :: Real -> Real -> Boolean

- static readonly

InterpreterAction< T > **Leq** = new OperationAction<T, double, bool>("Leq", (x, y) => x <= y)

Comparison "Less than or equal to".

Leq :: Real -> Real -> Boolean

- static readonly

InterpreterAction< T > **Empty** = new ConstantAction<T, string>("Empty", "")

Returns an empty string.

Empty :: String

- static readonly

InterpreterAction< T > **Join** = new JoinAction<T>()

Converts all arguments to strings and joins them with a given separator.

Join :: String -> [Any...] -> String

- static readonly

InterpreterAction< T > **Concat** = new ConcatAction<T>()

Converts all arguments to strings and concatenates them.

Concat :: [Any...] -> String

- static readonly

InterpreterAction< T > **Flatten** = new FlattenAction<T>()

Converts a tree into its string representation. An optional argument specifies the depth.

Flatten :: Tree -> [Integer] -> String

- static readonly

InterpreterAction< T > **Name** = new ConvertAction<T, Symbol, string>("Name", x => x.Name)

Returns the name of the given symbol.

Name :: Symbol -> String

- static readonly

InterpreterAction< T > **Symbol** = new ConvertAction<T, Tree<Symbol>, Symbol>("Symbol", x => x.Value)

Returns the root symbol of the given tree.

Symbol :: Tree -> Symbol

- static readonly

InterpreterAction< T > **Depth** = new ConvertAction<T, Tree<Symbol>, int>("Depth", x => x.Depth)

Returns the depth of the given tree.

Depth :: Tree -> Integer

- static readonly

InterpreterAction< T > **Leaves** = new ConvertAction<T, Tree<Symbol>, int>("Leaves", x => x.LeavesCount)

Returns the count of leaves of the given tree.

Leaves :: Tree -> Integer

- static readonly
InterpreterAction< T > [SubTree](#) = new SubTreeAction<T>()
Returns a child of a given tree at index specified as the second parameter. If no such child exists, the action returns null.
SubTree :: Tree -> Integer -> Tree
- static readonly
InterpreterAction< T > [Preterminal](#)
Returns a string value of a preterminal rule (of the form A -> b where b is the terminal. If the tree was not generated by a preterminal rule, returns null.
Preterminal :: Tree -> String
- static readonly
InterpreterAction< T > [If](#) = new IfAction<T>()
Condition. The semantics of (if c0 a0 c1 a1 ... e) is if (c0) then a1; else if (c1) then a1; ... else e;. All values and conditions are evaluated lazily. Moreover, types of a_ must be equal otherwise the type inference would fail.
If[a = Any] :: [Boolean,a...] -> a -> a
- static readonly
InterpreterAction< T > [Select](#) = new SelectAction<T>()
Selection. The semantics of (select i c0 c1 ... cn def) is if (0 <= i <= n) then ci; else def;. All values are evaluated lazily. Moreover, types of c_ must be equal otherwise the type inference would fail.
Select[a = Any] :: Integer -> [a...] -> a -> a
- static readonly
InterpreterAction< T > [Case](#) = new CaseAction<T>()
Case. The semantics of (case x c0 v0 c1 v1 ... def) is switch (x) { case c0: v0; break; case c1: v1; break; ... default: def; }. Moreover, types of c_ and x must be equal, as well as types of v_.
Case[v = Any, e = Any] :: v -> [v,e...] -> e -> e
- static readonly
InterpreterAction< T > [Eval](#) = new EvaluateAction<T>()
Recursively evaluates the given tree.
Eval :: Tree -> TValue
- static readonly
InterpreterAction< T > [Simple](#) = new SimpleAction<T>()
Evaluates the subtree of a given tree provided that it is the only subtree; otherwise returns null.
Simple :: Tree -> TValue
- static readonly
InterpreterAction< T > [Fail](#) = new FailAction<T>()
Returns null.
Fail :: Any
- static readonly
InterpreterAction< T > [Try](#) = new TryAction<T>()
Try action. The semantics of (try e1 e2) is try { e1; } catch (Mercury.Exceptions.ErrorActionException) { e2; }. If the second argument is not specified, it will behave like (try e1 (fail)). Moreover, the types of e1 and e2 must be equal.
Try[a = Any] :: a -> a -> a
- static readonly
InterpreterAction< T > [Error](#) = new ConvertAction<T, string, object>("Error", x => { throw new ErrorActionException(x); })
Throws an instance of Mercury.Exceptions.ErrorActionException with the specified text.
Error :: String -> Any
- static readonly
InterpreterAction< T > [Identity](#) = new IdentityAction<T>()
Returns the same value.
Identity[a = Any] :: a -> a

- static readonly
 InterpreterAction< T > **Let** = new LetAction<T>()
Saves the value in the current context.
The semantics of (Let x v0 y v1 ... act) is let (x = v0; y = v1; ...) in (act);.
The return type is same as the type of act.
`Let[a = Any] :: [String, Any...] -> a -> a`
- static readonly
 InterpreterAction< T > **Var** = new VarAction<T>()
*Retrieves the value stored by **Let**, but only in the same context. The function also performs runtime type check; if the value stored does not exist or match the type required, it will throw an exception.*
`Var :: String -> Any`
- static readonly
 InterpreterAction< T > **ArgsCount** = new ArgsCountAction<T>()
Returns the number of arguments. Useful to determine number of values captured by a [Mercury.Interpreting.Wildcard](#).
`ArgsCount :: [Any...] -> Integer`
- static readonly
 InterpreterAction< T > **Trace** = new TraceAction<T>()
Logs an event to the entry tree.
`Trace[a = Any] -> a -> String -> [Any...] -> a`

6.8.1 Detailed Description

Contains basic actions for interpretation.

Type specifiers of actions are defined as follows:

- `f :: type` defines an action `f` of type `type`.
- `f[a = X] :: type` defines an action `f` of type `type`, where `a` gets replaced by `X` and is always of the same type.

Syntax of action types is as follows:

- `a -> b -> c` is a function that takes two arguments of type `a` and `b` and returns an argument of type `c`.
- `[X]` is an optional argument of type `X`.
- `[X...]` is an arbitrary number of arguments of type `X`.
- `[X, Y...]` is an arbitrary number of arguments in the form of `X -> Y -> X -> Y` etc.
- `... arbitrary number of same arguments as previous, i.e. Integer -> ...` is at least one `Integer`.

Note that if `f :: a -> [b] -> [c] -> d`, the third argument of type `c` cannot be specified without the second argument of type `b`.

Template Parameters

<i>T</i>	Interpreter return type.
----------	--------------------------

Type Constraints

***T* : class**

Definition at line 39 of file BasicActions.cs.

6.8.2 Member Function Documentation

6.8.2.1 `static InterpreterAction<T> Mercury.Interpreting.BasicActions< T >.CreateTracer ( string name, TextWriter writer )`
`[static]`

Creates custom trace action with stream, useful for logging.

Parameters

<i>name</i>	The name of the action
<i>writer</i>	The output writer

Returns

The interpreter action

Definition at line 352 of file BasicActions.cs.

6.8.2.2 `static IReadOnlyDictionary<string,string> Mercury.Interpreting.BasicActions< T >.GetAliases ( )` `[static]`

Returns aliases for basic actions.

It contains the following aliases:

```

+ Add      - Sub      * Mul      / Div
// FDiv    % Mod      ** Pow
@ String   ++ Concat  # Var
& And      | Or       ^ Xor      ! Not
== Eq      /= Neq
> Gt       < Lt       >= Geq      <= Leq
$ Eval     id Identity

```

Definition at line 332 of file BasicActions.cs.

6.8.2.3 `static IReadOnlyList<InterpreterAction<T> > Mercury.Interpreting.BasicActions< T >.GetAllActions ( )`
`[static]`

Provides basic [Mercury](#) actions

Template Parameters

<i>T</i>	Output type parameter
----------	-----------------------

Returns

List of actions

Definition at line 304 of file BasicActions.cs.

6.8.3 Member Data Documentation

6.8.3.1 `readonly InterpreterAction<T> Mercury.Interpreting.BasicActions< T >.Add = new ArithmeticAction<T>("Add", (x, y) => x + y) [static]`

Sum of the arguments.

`Add[a = Integer|Real] :: a -> a -> ... -> a`

Definition at line 96 of file BasicActions.cs.

6.8.3.2 `readonly InterpreterAction<T> Mercury.Interpreting.BasicActions< T >.And = new FoldingAction<T, bool, bool>("And", 2, (x, y) => x && y, true) [static]`

Logical multiplication.

`And :: Boolean -> Boolean -> ... -> Boolean`

Definition at line 123 of file BasicActions.cs.

6.8.3.3 `readonly InterpreterAction<T> Mercury.Interpreting.BasicActions< T >.ArgsCount = new ArgsCountAction<T>() [static]`

Returns the number of arguments. Useful to determine number of values captured by a [Mercury.Interpreting.Wildcard](#).

`ArgsCount :: [Any...] -> Integer`

Definition at line 296 of file BasicActions.cs.

6.8.3.4 `readonly InterpreterAction<T> Mercury.Interpreting.BasicActions< T >.Case = new CaseAction<T>() [static]`

Case. The semantics of `(case x c0 v0 c1 v1 ... def)` is

```
switch (x) { case c0: v0; break; case c1: v1; break; ... default: def;
}.
```

Moreover, types of `c_` and `x` must be equal, as well as types of `v_`.

`Case[v = Any, e = Any] :: v -> [v,e...] -> e -> e`

Definition at line 238 of file BasicActions.cs.

6.8.3.5 `readonly InterpreterAction<T> Mercury.Interpreting.BasicActions< T >.Ceiling = new ConvertAction<T, double, int>("Ceiling", x => Convert.ToInt32(Math.Ceiling(x))) [static]`

Returns a ceiling of a real number.

`Ceiling :: Real -> Integer`

Definition at line 81 of file BasicActions.cs.

6.8.3.6 `readonly InterpreterAction<T> Mercury.Interpreting.BasicActions< T >.Concat = new ConcatAction<T>() [static]`

Converts all arguments to strings and concatenates them.

`Concat :: [Any...] -> String`

Definition at line 168 of file BasicActions.cs.

6.8.3.7 `readonly InterpreterAction<T> Mercury.Interpreting.BasicActions< T >.Depth = new ConvertAction<T, Tree<Symbol>, int>("Depth", x => x.Depth) [static]`

Returns the depth of the given tree.

`Depth :: Tree -> Integer`

Definition at line 186 of file BasicActions.cs.

6.8.3.8 `readonly InterpreterAction<T> Mercury.Interpreting.BasicActions< T >.Div = new ArithmeticAction<T>("Div", (x, y) => x / y) [static]`

Divides the values.

`Div[a = Integer|Real] :: a -> a -> ... -> a`

Definition at line 105 of file BasicActions.cs.

6.8.3.9 `readonly InterpreterAction<T> Mercury.Interpreting.BasicActions< T >.Empty = new ConstantAction<T, string>("Empty", "") [static]`

Returns an empty string.

`Empty :: String`

Definition at line 159 of file BasicActions.cs.

6.8.3.10 `readonly InterpreterAction<T> Mercury.Interpreting.BasicActions< T >.Eq = new OperationAction<T, object, bool>("Eq", (x, y) => (x != null) && x.Equals(y), ArgumentType.Null) [static]`

Indicates whether provided objects are equal. If any of arguments is `null`, the result is always `false`.

`Eq :: Any -> Any -> Boolean`

Definition at line 138 of file BasicActions.cs.

6.8.3.11 `readonly InterpreterAction<T> Mercury.Interpreting.BasicActions< T >.Error = new ConvertAction<T, string, object>("Error", x => { throw new ErrorActionException(x); }) [static]`

Throws an instance of [Mercury.Exceptions.ErrorActionException](#) with the specified text.

`Error :: String -> Any`

Definition at line 267 of file BasicActions.cs.

6.8.3.12 `readonly InterpreterAction<T> Mercury.Interpreting.BasicActions< T >.Eval = new EvaluateAction<T>() [static]`

Recursively evaluates the given tree.

`Eval :: Tree -> TValue`

Definition at line 241 of file BasicActions.cs.

6.8.3.13 `readonly InterpreterAction<T> Mercury.Interpreting.BasicActions< T >.Fail = new FailAction<T>() [static]`

Returns `null`.

`Fail :: Any`

Definition at line 251 of file BasicActions.cs.

6.8.3.14 `readonly InterpreterAction<T> Mercury.Interpreting.BasicActions< T >.FDiv = new OperationAction<T, double, double>("FDiv", (x, y) => x / y) [static]`

Floating point division.

```
FDiv :: Real -> Real -> Real
```

Definition at line 111 of file BasicActions.cs.

6.8.3.15 `readonly InterpreterAction<T> Mercury.Interpreting.BasicActions< T >.Flatten = new FlattenAction<T>() [static]`

Converts a tree into its string representation. An optional argument specifies the depth.

```
Flatten :: Tree -> [Integer] -> String
```

Definition at line 174 of file BasicActions.cs.

6.8.3.16 `readonly InterpreterAction<T> Mercury.Interpreting.BasicActions< T >.Floor = new ConvertAction<T, double, int>("Floor", x => Convert.ToInt32(Math.Floor(x))) [static]`

Returns a floor of a real number.

```
Floor :: Real -> Integer
```

Definition at line 84 of file BasicActions.cs.

6.8.3.17 `readonly InterpreterAction<T> Mercury.Interpreting.BasicActions< T >.Geq = new OperationAction<T, double, bool>("Geq", (x, y) => x >= y) [static]`

Comparison "Greater than or equal to".

```
Geq :: Real -> Real -> Boolean
```

Definition at line 150 of file BasicActions.cs.

6.8.3.18 `readonly InterpreterAction<T> Mercury.Interpreting.BasicActions< T >.Gr = new OperationAction<T, double, bool>("Gr", (x, y) => x > y) [static]`

Comparison "Greater than".

```
Gr :: Real -> Real -> Boolean
```

Definition at line 144 of file BasicActions.cs.

6.8.3.19 `readonly InterpreterAction<T> Mercury.Interpreting.BasicActions< T >.Identity = new IdentityAction<T>() [static]`

Returns the same value.

```
Identity[a = Any] :: a -> a
```

Definition at line 273 of file BasicActions.cs.

6.8.3.20 `readonly InterpreterAction<T> Mercury.Interpreting.BasicActions< T >.If = new IfAction<T>() [static]`

Condition. The semantics of (if c0 a0 c1 a1 ... e) is if (c0) then a1; else if (c1) then a1; ... else e;. All values and conditions are evaluated lazily.

Moreover, types of a_ must be equal otherwise the type inference would fail.

```
If[a = Any] :: [Boolean, a...] -> a -> a
```

Definition at line 221 of file BasicActions.cs.

6.8.3.21 `readonly InterpreterAction<T> Mercury.Interpreting.BasicActions< T >.IsNull = new ConvertAction<T, object, bool>("IsNull", x => x == null, ArgumentType.Null) [static]`

Indicates whether the object is `null`.

`IsNull :: Any -> Boolean`

Definition at line 120 of file `BasicActions.cs`.

6.8.3.22 `readonly InterpreterAction<T> Mercury.Interpreting.BasicActions< T >.Join = new JoinAction<T>() [static]`

Converts all arguments to strings and joins them with a given separator.

`Join :: String -> [Any...] -> String`

Definition at line 165 of file `BasicActions.cs`.

6.8.3.23 `readonly InterpreterAction<T> Mercury.Interpreting.BasicActions< T >.Leaves = new ConvertAction<T, Tree<Symbol>, int>("Leaves", x => x.LeavesCount) [static]`

Returns the count of leaves of the given tree.

`Leaves :: Tree -> Integer`

Definition at line 189 of file `BasicActions.cs`.

6.8.3.24 `readonly InterpreterAction<T> Mercury.Interpreting.BasicActions< T >.Leq = new OperationAction<T, double, bool>("Leq", (x, y) => x <= y) [static]`

Comparison "Less than or equal to".

`Leq :: Real -> Real -> Boolean`

Definition at line 153 of file `BasicActions.cs`.

6.8.3.25 `readonly InterpreterAction<T> Mercury.Interpreting.BasicActions< T >.Let = new LetAction<T>() [static]`

Saves the value in the *current* context.

The semantics of `(Let x v0 y v1 ... act)` is `let (x = v0; y = v1; ...) in (act);`.

The return type is same as the type of `act`.

`Let[a = Any] :: [String, Any...] -> a -> a`

Definition at line 282 of file `BasicActions.cs`.

6.8.3.26 `readonly InterpreterAction<T> Mercury.Interpreting.BasicActions< T >.Ls = new OperationAction<T, double, bool>("Ls", (x, y) => x < y) [static]`

Comparison "Less than".

`Ls :: Real -> Real -> Boolean`

Definition at line 147 of file `BasicActions.cs`.

6.8.3.27 `readonly InterpreterAction<T> Mercury.Interpreting.BasicActions< T >.Mod = new OperationAction<T, int, int>("Mod", (x, y) => x % y) [static]`

Modulo.

`Mod :: Integer -> Integer -> Integer`

Definition at line 114 of file `BasicActions.cs`.

6.8.3.28 `readonly InterpreterAction<T> Mercury.Interpreting.BasicActions< T >.Mul = new ArithmeticAction<T>("Mul", (x, y) => x * y) [static]`

Multiplies the arguments.

`Mul[a = Integer|Real] :: a -> a -> ... -> a`

Definition at line 102 of file BasicActions.cs.

6.8.3.29 `readonly InterpreterAction<T> Mercury.Interpreting.BasicActions< T >.Name = new ConvertAction<T, Symbol, string>("Name", x => x.Name) [static]`

Returns the name of the given symbol.

`Name :: Symbol -> String`

Definition at line 180 of file BasicActions.cs.

6.8.3.30 `readonly InterpreterAction<T> Mercury.Interpreting.BasicActions< T >.Neq = new OperationAction<T, object, bool>("Neq", (x, y) => (x == null) || !x.Equals(y), ArgumentType.Null) [static]`

Complement of [Eq](#).

`Neq :: Any -> Any -> Boolean`

Definition at line 141 of file BasicActions.cs.

6.8.3.31 `readonly InterpreterAction<T> Mercury.Interpreting.BasicActions< T >.Not = new ConvertAction<T, bool, bool>("Not", x => !x) [static]`

Logical negation.

`Not :: Boolean -> Boolean`

Definition at line 129 of file BasicActions.cs.

6.8.3.32 `readonly InterpreterAction<T> Mercury.Interpreting.BasicActions< T >.Or = new FoldingAction<T, bool, bool>("Or", 2, (x, y) => x && y, false) [static]`

Logical addition.

`Or :: Boolean -> Boolean -> ... -> Boolean`

Definition at line 126 of file BasicActions.cs.

6.8.3.33 `readonly InterpreterAction<T> Mercury.Interpreting.BasicActions< T >.ParseBoolean [static]`

Initial value:

```
= new ConvertAction<T, string, bool>("ParseBoolean",
    x => { bool b; return bool.TryParse(x, out b) ? b : default(bool); })
```

Converts a string representation of a boolean.

`ParseBoolean :: String -> Boolean`

Definition at line 71 of file BasicActions.cs.

6.8.3.34 `readonly InterpreterAction<T> Mercury.Interpreting.BasicActions< T >.ParseInteger [static]`

Initial value:

```
= new ConvertAction<T, string, int>("ParseInteger",
    x => { int i; return int.TryParse(x, NumberStyles.Integer, cinfo, out i) ? i : default(int);
    })
```

Converts a string representation of an integer.

ParseInteger :: String -> Integer

Definition at line 55 of file BasicActions.cs.

6.8.3.35 readonly InterpreterAction<T> Mercury.Interpreting.BasicActions<T>.ParseReal [static]

Initial value:

```
= new ConvertAction<T, string, double>("ParseReal",
    x => { double d; return double.TryParse(x, NumberStyles.Float, cinfo, out d) ? d : default(
    double); })
```

Converts a string representation of a decimal number.

ParseReal :: String -> Real

Definition at line 63 of file BasicActions.cs.

6.8.3.36 readonly InterpreterAction<T> Mercury.Interpreting.BasicActions<T>.Pow = new ArithmeticAction<T>("Pow", (x, y) => Math.Pow(x, y)) [static]

Exponentiates the values.

Pow[a = Integer|Real] :: a -> a -> ... -> a

Definition at line 108 of file BasicActions.cs.

6.8.3.37 readonly InterpreterAction<T> Mercury.Interpreting.BasicActions<T>.Preterminal [static]

Initial value:

```
= new ConvertAction<T, Tree<Symbol>, string>("Preterminal", x =>
    {
        return ((x.Children.Count != 1)
            || ((x[0].Value.Type & (SymbolType.Terminal |
            SymbolType.Epsilon)) == SymbolType.None))
            ? null : x[0].Value.Name;
    })
```

Returns a string value of a preterminal rule (of the form A -> b where b is the terminal. If the tree was not generated by a preterminal rule, returns null.

Preterminal :: Tree -> String

Definition at line 204 of file BasicActions.cs.

6.8.3.38 readonly InterpreterAction<T> Mercury.Interpreting.BasicActions<T>.Round = new ConvertAction<T, double, int>("Round", x => Convert.ToInt32(Math.Round(x))) [static]

Rounds a real number.

Round :: Real -> Integer

Definition at line 78 of file BasicActions.cs.

6.8.3.39 readonly InterpreterAction<T> Mercury.Interpreting.BasicActions<T>.Select = new SelectAction<T>() [static]

Selection. The semantics of (select i c0 c1 ... cn def) is if (0 <= i <= n) then ci; else def;. All values are evaluated lazily.

Moreover, types of `c_` must be equal otherwise the type inference would fail.

```
Select[a = Any] :: Integer -> [a...] -> a -> a
```

Definition at line 230 of file BasicActions.cs.

6.8.3.40 `readonly InterpreterAction<T> Mercury.Interpreting.BasicActions< T >.Simple = new SimpleAction<T>()`
[static]

Evaluates the subtree of a given tree provided that it is the only subtree; otherwise returns `null`.

```
Simple :: Tree -> TValue
```

Definition at line 248 of file BasicActions.cs.

6.8.3.41 `readonly InterpreterAction<T> Mercury.Interpreting.BasicActions< T >.String = new ConvertAction<T, object, string>("String", x => x.ToString())` [static]

Call `ToString()` method on its argument.

```
String :: Any -> String
```

Definition at line 90 of file BasicActions.cs.

6.8.3.42 `readonly InterpreterAction<T> Mercury.Interpreting.BasicActions< T >.Sub = new ArithmeticAction<T>("Sub", (x, y) => x - y)` [static]

Subtracts the arguments.

```
Sub[a = Integer|Real] :: a -> a -> ... -> a
```

Definition at line 99 of file BasicActions.cs.

6.8.3.43 `readonly InterpreterAction<T> Mercury.Interpreting.BasicActions< T >.SubTree = new SubTreeAction<T>()`
[static]

Returns a child of a given tree at index specified as the second parameter. If no such child exists, the action returns `null`.

```
SubTree :: Tree -> Integer -> Tree
```

Definition at line 196 of file BasicActions.cs.

6.8.3.44 `readonly InterpreterAction<T> Mercury.Interpreting.BasicActions< T >.Symbol = new ConvertAction<T, Tree<Symbol>, Symbol>("Symbol", x => x.Value)` [static]

Returns the root symbol of the given tree.

```
Symbol :: Tree -> Symbol
```

Definition at line 183 of file BasicActions.cs.

6.8.3.45 `readonly InterpreterAction<T> Mercury.Interpreting.BasicActions< T >.ToReal = new ConvertAction<T, int, double>("ToReal", x => Convert.ToDouble(x))` [static]

Converts an integer value to decimal.

```
ToReal :: Integer -> Real
```

Definition at line 87 of file BasicActions.cs.

6.8.3.46 `readonly InterpreterAction<T> Mercury.Interpreting.BasicActions< T >.Trace = new TraceAction<T>()`
`[static]`

Logs an event to the entry tree.

`Trace[a = Any] -> a -> String -> [Any...] -> a`

Definition at line 299 of file BasicActions.cs.

6.8.3.47 `readonly InterpreterAction<T> Mercury.Interpreting.BasicActions< T >.Try = new TryAction<T>()` `[static]`

Try action. The semantics of `(try e1 e2) is try { e1; } catch (Mercury.Exceptions.Error←
ActionException) { e2; }.`

If the second argument is not specified, it will behave like `(try e1 (fail))`. Moreover, the types of `e1` and `e2` must be equal.

`Try[a = Any] :: a -> a -> a`

Definition at line 260 of file BasicActions.cs.

6.8.3.48 `readonly InterpreterAction<T> Mercury.Interpreting.BasicActions< T >.Var = new VarAction<T>()` `[static]`

Retrieves the value stored by [Let](#), but only in the same context. The function also performs runtime type check; if the value stored does not exist or match the type required, it will throw an exception.

`Var :: String -> Any`

Definition at line 290 of file BasicActions.cs.

6.8.3.49 `readonly InterpreterAction<T> Mercury.Interpreting.BasicActions< T >.Xor = new OperationAction<T, bool,
 bool>("Xor", (x, y) => x ^ y)` `[static]`

Exclusive disjunction.

`Xor :: Boolean -> Boolean -> Boolean`

Definition at line 132 of file BasicActions.cs.

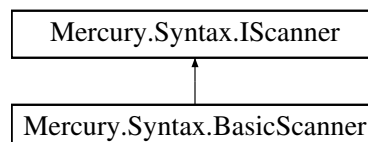
The documentation for this class was generated from the following file:

- [Mercury/Mercury/Interpreting/BasicActions.cs](#)

6.9 Mercury.Syntax.BasicScanner Class Reference

Generates four Nonterminals `@Symbol`, `@Keyword`, `@Number` and `@String` based on the given map and types of input tokens.

Inheritance diagram for Mercury.Syntax.BasicScanner:



Public Member Functions

- [BasicScanner](#) ([BasicScannerMap](#) map)

Initializes a new instance of the [BasicScanner](#) class.

- `IEnumerable< Rule > Scan (IEnumerable< Token > tokens)`
Scans input tokens and creates additional rules for parsing.

6.9.1 Detailed Description

Generates four Nonterminals `@Symbol`, `@Keyword`, `@Number` and `@String` based on the given map and types of input tokens.

Definition at line 229 of file Scanner.cs.

6.9.2 Constructor & Destructor Documentation

6.9.2.1 Mercury.Syntax.BasicScanner.BasicScanner (BasicScannerMap map)

Initializes a new instance of the `BasicScanner` class.

Parameters

<i>map</i>	The map of types
------------	------------------

Exceptions

<i>System.ArgumentNull← Exception</i>	When map is null
---	------------------

Definition at line 245 of file Scanner.cs.

6.9.3 Member Function Documentation

6.9.3.1 IEnumerable<Rule> Mercury.Syntax.BasicScanner.Scan (IEnumerable< Token > tokens)

Scans input tokens and creates additional rules for parsing.

Parameters

<i>tokens</i>	Input tokens.
---------------	---------------

Returns

Enumeration of additional rules.

Implements `Mercury.Syntax.IScanner`.

Definition at line 256 of file Scanner.cs.

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

- Mercury/Mercury/Syntax/[Scanner.cs](#)

6.10 Mercury.Syntax.BasicScannerMap Class Reference

Contains mapping between TokenType and `Mercury.Syntax.Mapping` of `Mercury.Syntax.BasicScanner`.

Public Member Functions

- `BasicScannerMap (IDictionary< TokenType, Mapping > map)`

Creates a new instance of [Mercury.Syntax.BasicScannerMap](#).

- [Mapping GetMapping](#) ([TokenType](#) ttype)

Returns the mapping for a given [TokenType](#).

Static Public Member Functions

- static [BasicScannerMap SingleMap](#) ([Mapping](#) custom)

Creates a mapping for all [TokenTypes](#) to custom [Mapping](#)

Properties

- static [BasicScannerMap Identity](#) [get]

The Identity map ([Symbol](#) to [Symbol](#), [Number](#) as [Number](#) etc.)

- static [BasicScannerMap NumberOrString](#) [get]

Maps [Number](#) to [Number](#) and other [TokenTypes](#) to [String](#)

- static [BasicScannerMap All](#) [get]

Maps any token type to everything

- [Mapping this\[TokenType ttype\]](#) [get]

6.10.1 Detailed Description

Contains mapping between [TokenType](#) and [Mercury.Syntax.Mapping](#) of [Mercury.Syntax.BasicScanner](#).

Definition at line 132 of file [Scanner.cs](#).

6.10.2 Constructor & Destructor Documentation

6.10.2.1 [Mercury.Syntax.BasicScannerMap.BasicScannerMap](#) ([IDictionary](#)< [TokenType](#), [Mapping](#) > map)

Creates a new instance of [Mercury.Syntax.BasicScannerMap](#).

Parameters

<i>map</i>	The map between TokenType and Mapping
------------	---

Exceptions

<i>System.ArgumentNull</i> <i>Exception</i>	When map is null
--	------------------

Definition at line 145 of file [Scanner.cs](#).

6.10.3 Member Function Documentation

6.10.3.1 [Mapping Mercury.Syntax.BasicScannerMap.GetMapping](#) ([TokenType](#) ttype)

Returns the mapping for a given [TokenType](#).

Parameters

<i>ttype</i>	The token type
--------------	----------------

Returns

[Mapping](#)

Definition at line 209 of file [Scanner.cs](#).

6.10.3.2 static BasicScannerMap Mercury.Syntax.BasicScannerMap.SingleMap (Mapping *custom*) [static]

Creates a mapping for all TokenType to custom Mapping

Parameters

<i>custom</i>	The custom mapping
---------------	--------------------

Returns

Scanner map

Definition at line 188 of file Scanner.cs.

6.10.4 Property Documentation

6.10.4.1 BasicScannerMap Mercury.Syntax.BasicScannerMap.All [static], [get]

Maps any token type to everything

Definition at line 176 of file Scanner.cs.

6.10.4.2 BasicScannerMap Mercury.Syntax.BasicScannerMap.Identity [static], [get]

The Identity map ([Symbol](#) to [Symbol](#), Number as Number etc.)

Definition at line 158 of file Scanner.cs.

6.10.4.3 BasicScannerMap Mercury.Syntax.BasicScannerMap.NumberOrString [static], [get]

Maps Number to Number and other TokenType to String

Definition at line 167 of file Scanner.cs.

6.10.4.4 Mapping Mercury.Syntax.BasicScannerMap.this[TokenType ttype] [get]

Definition at line 200 of file Scanner.cs.

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

- Mercury/Mercury/Syntax/[Scanner.cs](#)

6.11 Mercury.Nucleus.Trees.BFSTreeEnumerator< T > Class Template Reference

Breadth-First Search tree enumerator

Inheritance diagram for Mercury.Nucleus.Trees.BFSTreeEnumerator< T >:



Public Member Functions

- [BFSTreeEnumerator](#) (Tree< T > root)
Creates a new instance of the `BFSTreeEnumerator{T}` class.
- void [Dispose](#) ()
- bool [MoveNext](#) ()
- void [Reset](#) ()

Properties

- T [Current](#) [get]
Gets the element in the collection at the current position of the enumerator.

6.11.1 Detailed Description

Breadth-First Search tree enumerator

Template Parameters

<i>T</i>	Node value type
----------	-----------------

Definition at line 90 of file `TreeEnumerators.cs`.

6.11.2 Constructor & Destructor Documentation

6.11.2.1 `Mercury.Nucleus.Trees.BFSTreeEnumerator< T >.BFSTreeEnumerator ( Tree< T > root )`

Creates a new instance of the `BFSTreeEnumerator{T}` class.

Parameters

<i>root</i>	The root element
-------------	------------------

Definition at line 105 of file `TreeEnumerators.cs`.

6.11.3 Member Function Documentation

6.11.3.1 `void Mercury.Nucleus.Trees.BFSTreeEnumerator< T >.Dispose ( )`

Definition at line 130 of file `TreeEnumerators.cs`.

6.11.3.2 `bool Mercury.Nucleus.Trees.BFSTreeEnumerator< T >.MoveNext ( )`

Definition at line 136 of file `TreeEnumerators.cs`.

6.11.3.3 `void Mercury.Nucleus.Trees.BFSTreeEnumerator< T >.Reset ( )`

Definition at line 150 of file `TreeEnumerators.cs`.

6.11.4 Property Documentation

6.11.4.1 `T Mercury.Nucleus.Trees.BFSTreeEnumerator< T >.Current [get]`

Gets the element in the collection at the current position of the enumerator.

Returns

The element in the collection at the current position of the enumerator.

Definition at line 118 of file TreeEnumerators.cs.

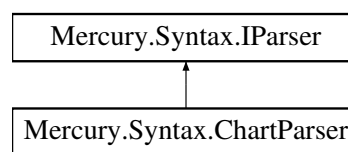
The documentation for this class was generated from the following file:

- Mercury/Mercury/Nucleus/Trees/[TreeEnumerators.cs](#)

6.12 Mercury.Syntax.ChartParser Class Reference

Thread-safe wrapper for ChartParserInternal class.

Inheritance diagram for Mercury.Syntax.ChartParser:

**Public Member Functions**

- [ChartParser](#) ([Grammar](#) grammar)
Chart Parser constructor
- [ParserResult Parse](#) (IList< [Token](#) > tokens)
Creates a trees (in [Mercury.Syntax.ParserResult](#)) from the tokens
- [ParserResult Parse](#) (IList< [Token](#) > tokens, IList< [Rule](#) > scanresult)
Creates a trees (in [Mercury.Syntax.ParserResult](#)) from the tokens with the support of [Mercury.Syntax.IScanner](#) generated rules.

Properties

- [Grammar Grammar](#) [get, set]
[Grammar](#) that this parser recognizes.

6.12.1 Detailed Description

Thread-safe wrapper for ChartParserInternal class.

Definition at line 298 of file Parser.cs.

6.12.2 Constructor & Destructor Documentation

6.12.2.1 Mercury.Syntax.ChartParser.ChartParser ([Grammar grammar](#))

Chart Parser constructor

Parameters

<i>grammar</i>	A Context-Free Grammar
----------------	--

Definition at line 308 of file Parser.cs.

6.12.3 Member Function Documentation

6.12.3.1 `ParserResult Mercury.Syntax.ChartParser.Parse ( IList< Token > tokens )`

Creates a trees (in [Mercury.Syntax.ParserResult](#)) from the tokens

Parameters

<i>tokens</i>	The tokens.
---------------	-------------

Returns

Parser result containing trees.

Implements [Mercury.Syntax.IParser](#).

Definition at line 323 of file Parser.cs.

6.12.3.2 `ParserResult Mercury.Syntax.ChartParser.Parse ( IList< Token > tokens, IList< Rule > scanresult )`

Creates a trees (in [Mercury.Syntax.ParserResult](#)) from the tokens with the support of [Mercury.Syntax.IScanner](#) generated rules.

Parameters

<i>tokens</i>	The tokens.
<i>scanresult</i>	The rules from scanner

Returns

Implements [Mercury.Syntax.IParser](#).

Definition at line 326 of file Parser.cs.

6.12.4 Property Documentation

6.12.4.1 `Grammar Mercury.Syntax.ChartParser.Grammar [get], [set]`

[Grammar](#) that this parser recognizes.

Definition at line 317 of file Parser.cs.

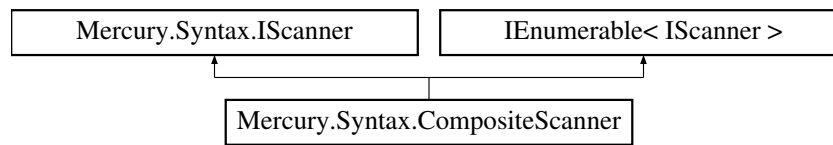
The documentation for this class was generated from the following file:

- [Mercury/Mercury/Syntax/Parser.cs](#)

6.13 Mercury.Syntax.CompositeScanner Class Reference

[Mercury.Syntax.IScanner](#) implementation that behaves like container of scanners using the Composite Design Pattern.

Inheritance diagram for Mercury.Syntax.CompositeScanner:



Public Member Functions

- [CompositeScanner](#) ()
Creates an empty Composite scanner.
- [CompositeScanner](#) (IEnumerable< [IScanner](#) > scanners)
Creates a scanner initialized by source scanners.
- [IScanner Add](#) ([IScanner](#) scanner)
Adds the specified scanner to the container. If the scanner is already present, this method will have no effect.
- [IScanner Remove](#) ([IScanner](#) scanner)
Removes the scanner from the container.
- IEnumerable< [Rule](#) > [Scan](#) (IEnumerable< [Token](#) > tokens)
Scans input tokens and creates additional rules for parsing.
- IEnumerator< [IScanner](#) > [GetEnumerator](#) ()

6.13.1 Detailed Description

[Mercury.Syntax.IScanner](#) implementation that behaves like container of scanners using the Composite Design Pattern.

When `Scan` method is called, it internally calls all scanners and concatenates their outputs.

Definition at line 42 of file `Scanner.cs`.

6.13.2 Constructor & Destructor Documentation

6.13.2.1 [Mercury.Syntax.CompositeScanner.CompositeScanner](#) ()

Creates an empty Composite scanner.

Definition at line 53 of file `Scanner.cs`.

6.13.2.2 [Mercury.Syntax.CompositeScanner.CompositeScanner](#) (IEnumerable< [IScanner](#) > scanners)

Creates a scanner initialized by source scanners.

Parameters

<i>scanners</i>	The source scanners
-----------------	---------------------

Definition at line 60 of file `Scanner.cs`.

6.13.3 Member Function Documentation

6.13.3.1 [IScanner Mercury.Syntax.CompositeScanner.Add](#) ([IScanner scanner](#))

Adds the specified scanner to the container. If the scanner is already present, this method will have no effect.

Parameters

<i>scanner</i>	The scanner.
----------------	--------------

Returns

This instance to support chaining.

Definition at line 72 of file Scanner.cs.

6.13.3.2 `IEnumerator<IScanner> Mercury.Syntax.CompositeScanner.GetEnumerator ( )`

Definition at line 100 of file Scanner.cs.

6.13.3.3 `IScanner Mercury.Syntax.CompositeScanner.Remove ( IScanner scanner )`

Removes the scanner from the container.

Parameters

<i>scanner</i>	The scanner.
----------------	--------------

Returns

This instance

Definition at line 83 of file Scanner.cs.

6.13.3.4 `IEnumerable<Rule> Mercury.Syntax.CompositeScanner.Scan ( IEnumerable<Token > tokens )`

Scans input tokens and creates additional rules for parsing.

Parameters

<i>tokens</i>	Input tokens.
---------------	---------------

Returns

Enumeration of additional rules.

Implements [Mercury.Syntax.IScanner](#).

Definition at line 93 of file Scanner.cs.

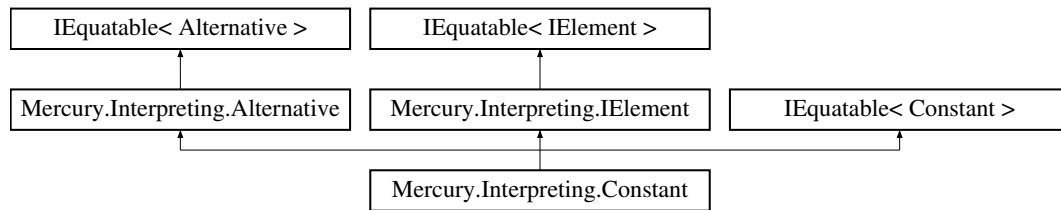
The documentation for this class was generated from the following file:

- [Mercury/Mercury/Syntax/Scanner.cs](#)

6.14 Mercury.Interpreting.Constant Class Reference

[Constant](#) element. Basically equivalent of [Mercury.Interpreting.Alternative](#).

Inheritance diagram for Mercury.Interpreting.Constant:



Public Member Functions

- [Constant](#) ([Symbol](#) symbol)
Creates a [Constant](#) made of one symbol.
- [Constant](#) ([IEnumerable](#)< [Symbol](#) > symbols)
[Constant](#) that matches any of the specified symbols.
- [Constant](#) ([SymbolType](#) sm, [IEnumerable](#)< [Symbol](#) > symbols)
Initializes a new instance of the [Constant](#) class. See [Mercury.Interpreting.Alternative](#)'s constructor for more details.
- override [bool](#) [Equals](#) ([object](#) obj)
- override [int](#) [GetHashCode](#) ()
- override [string](#) [ToString](#) ()
- [bool](#) [Equals](#) ([IElement](#) other)
- [bool](#) [Equals](#) ([Constant](#) other)

Properties

- [ElementType](#) [Type](#) [get, set]

Additional Inherited Members

6.14.1 Detailed Description

[Constant](#) element. Basically equivalent of [Mercury.Interpreting.Alternative](#).

Definition at line 178 of file Element.cs.

6.14.2 Constructor & Destructor Documentation

6.14.2.1 Mercury.Interpreting.Constant.Constant ([Symbol](#) symbol)

Creates a [Constant](#) made of one symbol.

Parameters

symbol	The symbol.
------------------------	-------------

Definition at line 186 of file Element.cs.

6.14.2.2 Mercury.Interpreting.Constant.Constant ([IEnumerable](#)< [Symbol](#) > symbols)

[Constant](#) that matches any of the specified symbols.

Parameters

<i>symbols</i>	The sequence symbols.
----------------	-----------------------

Definition at line 192 of file Element.cs.

6.14.2.3 Mercury.Interpreting.Constant.Constant (**SymbolType** *sm*, IEnumerable< **Symbol** > *symbols*)

Initializes a new instance of the [Constant](#) class. See [Mercury.Interpreting.Alternative](#)'s constructor for more details.

Parameters

<i>sm</i>	The symbol mask.
<i>symbols</i>	The symbols.

Definition at line 203 of file Element.cs.

6.14.3 Member Function Documentation

6.14.3.1 override bool Mercury.Interpreting.Constant.Equals (object *obj*)

Definition at line 211 of file Element.cs.

6.14.3.2 bool Mercury.Interpreting.Constant.Equals (**IElement** *other*)

Definition at line 235 of file Element.cs.

6.14.3.3 bool Mercury.Interpreting.Constant.Equals (**Constant** *other*)

Definition at line 242 of file Element.cs.

6.14.3.4 override int Mercury.Interpreting.Constant.GetHashCode ()

Definition at line 217 of file Element.cs.

6.14.3.5 override string Mercury.Interpreting.Constant.ToString ()

Definition at line 220 of file Element.cs.

6.14.4 Property Documentation

6.14.4.1 **ElementType** Mercury.Interpreting.Constant.Type [get], [set]

Definition at line 229 of file Element.cs.

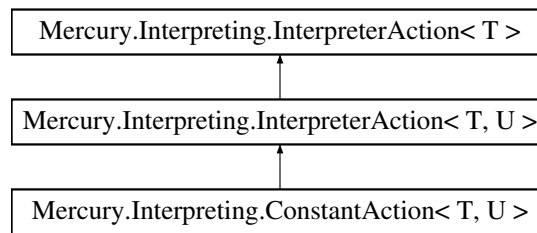
The documentation for this class was generated from the following file:

- Mercury/Mercury/Interpreting/[Element.cs](#)

6.15 Mercury.Interpreting.ConstantAction< T, U > Class Template Reference

Wraps a value as an action with no parameters

Inheritance diagram for Mercury.Interpreting.ConstantAction< T, U >:



Public Member Functions

- [ConstantAction](#) (string name, U value)
Initializes a new instance of the ConstantAction{T, U} class.
- override U [Evaluate](#) ([Argument](#)[]) arguments, RecursiveEval< T > eval, InterpreterContext< T > context)
Evaluates the action.

Additional Inherited Members

6.15.1 Detailed Description

Wraps a value as an action with no parameters

Template Parameters

<i>T</i>	Interpreter return type
<i>U</i>	Value type

Type Constraints

***T* : class**

Definition at line 500 of file BasicActions.cs.

6.15.2 Constructor & Destructor Documentation

6.15.2.1 Mercury.Interpreting.ConstantAction< T, U >.ConstantAction (string name, U value)

Initializes a new instance of the ConstantAction{T, U} class.

Parameters

<i>name</i>	The name of the action
<i>value</i>	The value

Definition at line 512 of file BasicActions.cs.

6.15.3 Member Function Documentation

6.15.3.1 override U Mercury.Interpreting.ConstantAction< T, U >.Evaluate ([Argument](#)[] arguments, RecursiveEval< T > eval, InterpreterContext< T > context) [virtual]

Evaluates the action.

Parameters

<i>arguments</i>	Arguments provided by the interpreter.
<i>eval</i>	The recursive evaluation delegate.
<i>context</i>	The interpreter context.

Returns

Implements [Mercury.Interpreting.InterpreterAction< T, U >](#).

Definition at line 520 of file BasicActions.cs.

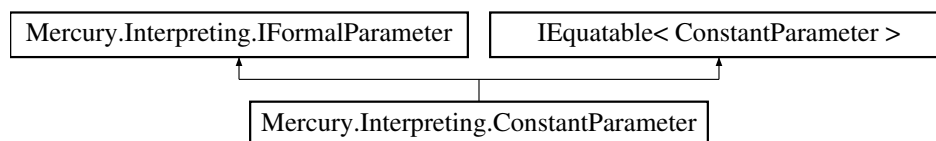
The documentation for this class was generated from the following file:

- Mercury/Mercury/Interpreting/[BasicActions.cs](#)

6.16 Mercury.Interpreting.ConstantParameter Class Reference

[Constant](#) parameter.

Inheritance diagram for Mercury.Interpreting.ConstantParameter:



Public Member Functions

- [ConstantParameter](#) (object value, [ParameterType](#) type)
Creates a new [ConstantParameter](#) instance.
- override bool [Equals](#) (object obj)
- override int [GetHashCode](#) ()
- override string [ToString](#) ()
- bool [Equals](#) ([ConstantParameter](#) other)

Static Public Member Functions

- static [ConstantParameter](#) [IntegralParameter](#) (int value)
Creates a new integral constant parameter.
- static [ConstantParameter](#) [RealParameter](#) (double value)
Creates a new real constant parameter.
- static [ConstantParameter](#) [BooleanParameter](#) (bool value)
Creates a new boolean constant parameter
- static [ConstantParameter](#) [StringParameter](#) (string value)
Creates a new string constant parameter.
- static [ConstantParameter](#) [AutoCreate](#) (string value)
Tries to parse the string value and infers its type.

Properties

- object [Value](#) [get, set]
Value of this constant
- [ParameterType Type](#) [get, set]

6.16.1 Detailed Description

[Constant](#) parameter.

Definition at line 43 of file FormalParameter.cs.

6.16.2 Constructor & Destructor Documentation

6.16.2.1 Mercury.Interpreting.ConstantParameter.ConstantParameter (object *value*, [ParameterType type](#))

Creates a new [ConstantParameter](#) instance.

Parameters

<i>value</i>	The value of this constant.
<i>type</i>	Parameter type.

Exceptions

<i>System.ArgumentNull</i> ↔ <i>Exception</i>	When value is null.
--	---------------------

Definition at line 53 of file FormalParameter.cs.

6.16.3 Member Function Documentation

6.16.3.1 static [ConstantParameter](#) Mercury.Interpreting.ConstantParameter.AutoCreate (string *value*) [static]

Tries to parse the string value and infers its type.

Parameters

<i>value</i>	String constant value.
--------------	------------------------

Returns

A constant parameter.

Definition at line 98 of file FormalParameter.cs.

6.16.3.2 static [ConstantParameter](#) Mercury.Interpreting.ConstantParameter.BooleanParameter (bool *value*) [static]

Creates a new boolean constant parameter

Parameters

<i>value</i>	The value.
--------------	------------

Returns

The constant parameter.

Definition at line 84 of file FormalParameter.cs.

6.16.3.3 override bool Mercury.Interpreting.ConstantParameter.Equals (object *obj*)

Definition at line 117 of file FormalParameter.cs.

6.16.3.4 bool Mercury.Interpreting.ConstantParameter.Equals (ConstantParameter *other*)

Definition at line 141 of file FormalParameter.cs.

6.16.3.5 override int Mercury.Interpreting.ConstantParameter.GetHashCode ()

Definition at line 123 of file FormalParameter.cs.

6.16.3.6 static ConstantParameter Mercury.Interpreting.ConstantParameter.IntegralParameter (int *value*) [static]

Creates a new integral constant parameter.

Parameters

<i>value</i>	The value.
--------------	------------

Returns

The constant parameter.

Definition at line 72 of file FormalParameter.cs.

6.16.3.7 static ConstantParameter Mercury.Interpreting.ConstantParameter.RealParameter (double *value*) [static]

Creates a new real constant parameter.

Parameters

<i>value</i>	The value.
--------------	------------

Returns

The constant parameter.

Definition at line 78 of file FormalParameter.cs.

6.16.3.8 static ConstantParameter Mercury.Interpreting.ConstantParameter.StringParameter (string *value*) [static]

Creates a new string constant parameter.

Parameters

<i>value</i>	The value.
--------------	------------

Returns

The constant parameter.

Definition at line 90 of file FormalParameter.cs.

6.16.3.9 override string Mercury.Interpreting.ConstantParameter.ToString ()

Definition at line 126 of file FormalParameter.cs.

6.16.4 Property Documentation

6.16.4.1 ParameterType Mercury.Interpreting.ConstantParameter.Type [get], [set]

Definition at line 135 of file FormalParameter.cs.

6.16.4.2 object Mercury.Interpreting.ConstantParameter.Value [get], [set]

Value of this constant

Definition at line 64 of file FormalParameter.cs.

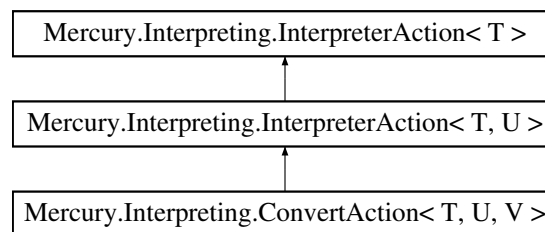
The documentation for this class was generated from the following file:

- Mercury/Mercury/Interpreting/[FormalParameter.cs](#)

6.17 Mercury.Interpreting.ConvertAction< T, U, V > Class Template Reference

Applies the lambda function on a single value.

Inheritance diagram for Mercury.Interpreting.ConvertAction< T, U, V >:



Public Member Functions

- [ConvertAction](#) (string name, Func< U, V > f, [ArgumentType](#) flags=ArgumentType.None)
Initializes a new instance of the ConvertAction{T, U, V} class.
- override V [Evaluate](#) ([Argument](#)[] arguments, RecursiveEval< T > eval, InterpreterContext< T > context)
Evaluates the action.

Additional Inherited Members

6.17.1 Detailed Description

Applies the lambda function on a single value.

Template Parameters

<i>T</i>	Interpreter return type
<i>U</i>	Input type
<i>V</i>	Output type

Type Constraints

***T* : class**

Definition at line 461 of file BasicActions.cs.

6.17.2 Constructor & Destructor Documentation

6.17.2.1 `Mercury.Interpreting.ConvertAction< T, U, V >.ConvertAction ( string name, Func< U, V > f, ArgumentType flags = ArgumentType.None )`

Initializes a new instance of the `ConvertAction{T, U, V}` class.

Parameters

<i>name</i>	The name of the action
<i>f</i>	The lambda function
<i>flags</i>	Additional flags

Exceptions

<code>System.ArgumentNull← Exception</code>	When <i>f</i> is null
---	-----------------------

Definition at line 475 of file `BasicActions.cs`.

6.17.3 Member Function Documentation

6.17.3.1 `override V Mercury.Interpreting.ConvertAction< T, U, V >.Evaluate ( Argument[] arguments, RecursiveEval< T > eval, InterpreterContext< T > context ) [virtual]`

Evaluates the action.

Parameters

<i>arguments</i>	Arguments provided by the interpreter.
<i>eval</i>	The recursive evaluation delegate.
<i>context</i>	The interpreter context.

Returns

Implements [Mercury.Interpreting.InterpreterAction< T, U >](#).

Definition at line 487 of file `BasicActions.cs`.

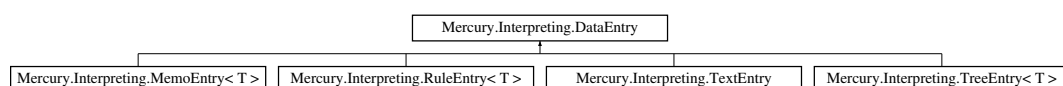
The documentation for this class was generated from the following file:

- `Mercury/Mercury/Interpreting/BasicActions.cs`

6.18 Mercury.Interpreting.DataEntry Interface Reference

Entry base (dummy) interface.

Inheritance diagram for `Mercury.Interpreting.DataEntry`:



6.18.1 Detailed Description

Entry base (dummy) interface.

Definition at line 17 of file Entries.cs.

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

- Mercury/Mercury/Interpreting/[Entries.cs](#)

6.19 Mercury.Defaults Class Reference

Global class containing cool static / convenience stuff.

Static Public Member Functions

- static [ArgumentType](#) [ParamToArgType](#) ([ParameterType](#) ptype)
Translates formal [Mercury.Interpreting.ParameterType](#) into [Mercury.Interpreting.ArgumentType](#). If the input is [Mercury.Interpreting.ParameterType.ActionCall](#), this function will yield [Mercury.Interpreting.ArgumentType.None](#) as it cannot resolve return types of the [Mercury.Interpreting.ActionCall](#). In that case use `ReturnType` field of the [Mercury.Interpreting.InterpreterAction{T}](#) class wrapped in the [Mercury.Interpreting.ActionCall](#).
- static [ArgumentType](#) [TypeToArgType](#)< T, U > ()
Translates a .NET type into [Mercury.Interpreting.ArgumentType](#). If T equals U, T:System.Object or dynamic this function will return [Mercury.Interpreting.ArgumentType.Any](#). If T is one of int, double, bool, string, Symbol or Tree<Symbol>, this function will yield [ArgumentType.TValue](#) | (equivalent of T). Otherwise it will yield equivalent [Mercury.Interpreting.ArgumentType](#) with no other types set.

Properties

- static string [Epsilon](#) [get]
Default epsilon symbol name
- static string [Root](#) [get]
Default root symbol name

6.19.1 Detailed Description

Global class containing cool static / convenience stuff.

Definition at line 15 of file Defaults.cs.

6.19.2 Member Function Documentation

6.19.2.1 static [ArgumentType](#) [Mercury.Defaults.ParamToArgType](#) ([ParameterType](#) ptype) [static]

Translates formal [Mercury.Interpreting.ParameterType](#) into [Mercury.Interpreting.ArgumentType](#). If the input is [Mercury.Interpreting.ParameterType.ActionCall](#), this function will yield [Mercury.Interpreting.ArgumentType.None](#) as it cannot resolve return types of the [Mercury.Interpreting.ActionCall](#). In that case use `ReturnType` field of the [Mercury.Interpreting.InterpreterAction{T}](#) class wrapped in the [Mercury.Interpreting.ActionCall](#).

Parameters

<i>ptype</i>	Type of the formal parameter
--------------	------------------------------

Returns

Equivalent argument type

Definition at line 66 of file Defaults.cs.

6.19.2.2 static `ArgumentType` `Mercury.Defaults.TypeToArgType< T, U > ( )` [static]

Translates a .NET type into [Mercury.Interpreting.ArgumentType](#). If T equals U , T :`System.Object` or `dynamic` this function will return `Mercury.Interpreting.ArgumentType.Any`. If T is one of `int`, `double`, `bool`, `string`, `Symbol` or `Tree<Symbol>`, this function will yield `ArgumentType.TValue` | (equivalent of T). Otherwise it will yield equivalent [Mercury.Interpreting.ArgumentType](#) with no other types set.

Template Parameters

T	Interpreter return type
U	

Returns

Definition at line 83 of file `Defaults.cs`.

6.19.3 Property Documentation

6.19.3.1 `string` `Mercury.Defaults.Epsilon` [static], [get]

Default epsilon symbol name

Definition at line 50 of file `Defaults.cs`.

6.19.3.2 `string` `Mercury.Defaults.Root` [static], [get]

Default root symbol name

Definition at line 53 of file `Defaults.cs`.

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

- `Mercury/Mercury/Defaults.cs`

6.20 `Mercury.Nucleus.Trees.DFSTreeEnumerator< T >` Class Template Reference

Depth-First Search tree enumerator

Inheritance diagram for `Mercury.Nucleus.Trees.DFSTreeEnumerator< T >`:



Public Member Functions

- [DFSTreeEnumerator](#) (`Tree< T >` root)
Creates a new instance of the `DFSTreeEnumerator{T}` class.
- void [Dispose](#) ()
- bool [MoveNext](#) ()
- void [Reset](#) ()

Properties

- **T Current** [get]

Gets the element in the collection at the current position of the enumerator.

6.20.1 Detailed Description

Depth-First Search tree enumerator

Template Parameters

<i>T</i>	Node value type
----------	-----------------

Definition at line 13 of file TreeEnumerators.cs.

6.20.2 Constructor & Destructor Documentation

6.20.2.1 Mercury.Nucleus.Trees.DFSTreeEnumerator< T >.DFSTreeEnumerator (Tree< T > root)

Creates a new instance of the DFSTreeEnumerator{T} class.

Parameters

<i>root</i>	The root node
-------------	---------------

Definition at line 28 of file TreeEnumerators.cs.

6.20.3 Member Function Documentation

6.20.3.1 void Mercury.Nucleus.Trees.DFSTreeEnumerator< T >.Dispose ()

Definition at line 53 of file TreeEnumerators.cs.

6.20.3.2 bool Mercury.Nucleus.Trees.DFSTreeEnumerator< T >.MoveNext ()

Definition at line 59 of file TreeEnumerators.cs.

6.20.3.3 void Mercury.Nucleus.Trees.DFSTreeEnumerator< T >.Reset ()

Definition at line 74 of file TreeEnumerators.cs.

6.20.4 Property Documentation

6.20.4.1 T Mercury.Nucleus.Trees.DFSTreeEnumerator< T >.Current [get]

Gets the element in the collection at the current position of the enumerator.

Returns

The element in the collection at the current position of the enumerator.

Definition at line 41 of file TreeEnumerators.cs.

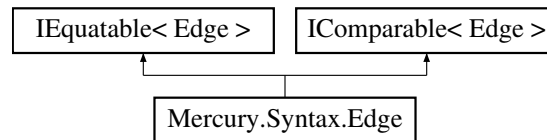
The documentation for this class was generated from the following file:

- Mercury/Mercury/Nucleus/Trees/[TreeEnumerators.cs](#)

6.21 Mercury.Syntax.Edge Class Reference

Represents an edge in the chart created by the [ChartParser](#). This is an immutable class. The [Edge](#) contains four values:

Inheritance diagram for Mercury.Syntax.Edge:



Public Member Functions

- [Edge](#) ([Rule](#) rule, int dot, int left, int right)
Creates a new edge. If the constructor does not throw any exception, the edge is guaranteed to be valid.
- bool [IsValid](#) ()
Determines whether this edge is valid.
- override bool [Equals](#) (object obj)
- override int [GetHashCode](#) ()
- override string [ToString](#) ()
- bool [Equals](#) ([Edge](#) other)
- int [CompareTo](#) ([Edge](#) other)
Compares the edge with another edge. Defines a lexicographical order as follows:

Properties

- [Rule Rule](#) [get, set]
The source rule.
- int [Dot](#) [get, set]
Position of the dot.
- int [Left](#) [get, set]
Left node number.
- int [Right](#) [get, set]
Right node number.
- bool [IsActive](#) [get, set]
Determines whether the edge is active.
- bool [IsPassive](#) [get, set]
*Complement of *IsActive* (convenience property).*
- [Symbol NextSymbol](#) [get, set]
*Returns the next symbol after the dot. The edge must be active, otherwise is *null*.*

6.21.1 Detailed Description

Represents an edge in the chart created by the [ChartParser](#). This is an immutable class. The [Edge](#) contains four values:

- `source rule`
- rule that possibly created a part of sentence
- `dot`
- position in the rule
- `left`
- number of left node in the chart
- `right`
- number of right node in the chart

`Edge` is usually written in the form $(l, r) A \rightarrow \alpha . \beta$ which means that this edge "covers" tokens l to r in the input (α) and is yet to cover β .

`Edge` is said to be active iff β is not empty (that is, $\text{dot} < n$). Otherwise the rule is said to be passive.

For an edge to be valid the following conditions must hold:

1. source rule is valid,
2.
 - a) $\text{dot} == 0$ if the source rule is an epsilon rule
 - b) $0 \leq \text{dot} \leq n$ otherwise (n is the count of RHS symbols)
3. $\text{left} \leq \text{right}$

Definition at line 37 of file `Edge.cs`.

6.21.2 Constructor & Destructor Documentation

6.21.2.1 Mercury.Syntax.Edge.Edge (Rule rule, int dot, int left, int right)

Creates a new edge. If the constructor does not throw any exception, the edge is guaranteed to be valid.

Parameters

<i>rule</i>	The source rule.
<i>dot</i>	Position of the dot.
<i>left</i>	Left node number.
<i>right</i>	Right node number.

Exceptions

<i>System.ArgumentNull↔ Exception</i>	When the source rule is null.
<i>Mercury.Exceptions↔ InvalidEdgeException</i>	When the edge is not valid.

Definition at line 55 of file Edge.cs.

6.21.3 Member Function Documentation

6.21.3.1 `int Mercury.Syntax.Edge.CompareTo ( Edge other )`

Compares the edge with another edge. Defines a lexicographical order as follows:

1. compare left node numbers
2. compare right node numbers
3. lexicographically compare source rules
4. compare dot positions

Parameters

<i>other</i>	Another edge
--------------	--------------

Returns

A value that indicates the relative order of the edges. See `Comparable`.

Definition at line 179 of file Edge.cs.

6.21.3.2 `override bool Mercury.Syntax.Edge.Equals ( object obj )`

Definition at line 124 of file Edge.cs.

6.21.3.3 `bool Mercury.Syntax.Edge.Equals ( Edge other )`

Definition at line 155 of file Edge.cs.

6.21.3.4 `override int Mercury.Syntax.Edge.GetHashCode ( )`

Definition at line 130 of file Edge.cs.

6.21.3.5 `bool Mercury.Syntax.Edge.IsValid ( )`

Determines whether this edge is valid.

Returns

`true` if the edge is valid, `false` otherwise.

Definition at line 117 of file Edge.cs.

6.21.3.6 `override string Mercury.Syntax.Edge.ToString ( )`

Definition at line 133 of file Edge.cs.

6.21.4 Property Documentation

6.21.4.1 `int Mercury.Syntax.Edge.Dot` `[get]`, `[set]`

Position of the dot.

Definition at line 76 of file `Edge.cs`.

6.21.4.2 `bool Mercury.Syntax.Edge.IsActive` `[get]`, `[set]`

Determines whether the edge is active.

`true` if the edge is active, `false` otherwise

Definition at line 91 of file `Edge.cs`.

6.21.4.3 `bool Mercury.Syntax.Edge.IsPassive` `[get]`, `[set]`

Complement of `IsActive` (convenience property).

`true` if the rule is not active `false` otherwise

Definition at line 100 of file `Edge.cs`.

6.21.4.4 `int Mercury.Syntax.Edge.Left` `[get]`, `[set]`

Left node number.

Definition at line 79 of file `Edge.cs`.

6.21.4.5 `Symbol Mercury.Syntax.Edge.NextSymbol` `[get]`, `[set]`

Returns the next symbol after the dot. The edge must be active, otherwise is `null`.

Definition at line 106 of file `Edge.cs`.

6.21.4.6 `int Mercury.Syntax.Edge.Right` `[get]`, `[set]`

Right node number.

Definition at line 82 of file `Edge.cs`.

6.21.4.7 `Rule Mercury.Syntax.Edge.Rule` `[get]`, `[set]`

The source rule.

Definition at line 73 of file `Edge.cs`.

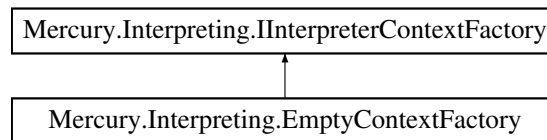
The documentation for this class was generated from the following file:

- `Mercury/Mercury/Syntax/Edge.cs`

6.22 Mercury.Interpreting.EmptyContextFactory Class Reference

Produces contexts with no custom data

Inheritance diagram for `Mercury.Interpreting.EmptyContextFactory`:



Public Member Functions

- `InterpreterContext< T > CreateContext< T > ()`
Creates a new context.

6.22.1 Detailed Description

Produces contexts with no custom data

Definition at line 117 of file `InterpreterContext.cs`.

6.22.2 Member Function Documentation

6.22.2.1 `InterpreterContext<T> Mercury.Interpreting.EmptyContextFactory.CreateContext< T > ( )`

Creates a new context.

Template Parameters

<i>T</i>	Interpreter output type.
----------	--------------------------

Returns

A new context.

Implements [Mercury.Interpreting.IInterpreterContextFactory](#).

Type Constraints

***T* : class**

Definition at line 124 of file `InterpreterContext.cs`.

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

- `Mercury/Mercury/Interpreting/InterpreterContext.cs`

6.23 Mercury.Interpreting.EntryWriter< T > Class Template Reference

An auxiliary class that formats the [DataEntry](#) structure and writes it into a stream.

Static Public Member Functions

- static void `WriteEntries` (`TreeEntry< T > root`, `TextWriter o`)
Writes the entries into the stream.

6.23.1 Detailed Description

An auxiliary class that formats the [DataEntry](#) structure and writes it into a stream.

Template Parameters

<i>T</i>	Interpreter return type
----------	-------------------------

Type Constraints

***T* : class**

Definition at line 146 of file Entries.cs.

6.23.2 Member Function Documentation

6.23.2.1 static void Mercury.Interpreting.EntryWriter< T >.WriteEntries (TreeEntry< T > *root*, TextWriter *o*) [static]

Writes the entries into the stream.

Parameters

<i>root</i>	The root.
<i>o</i>	The o.

Definition at line 236 of file Entries.cs.

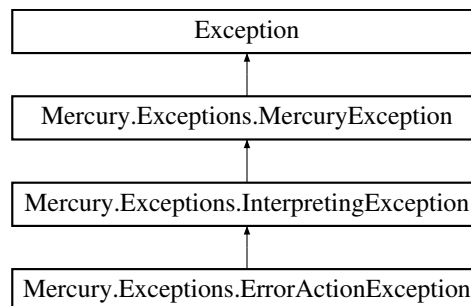
The documentation for this class was generated from the following file:

- Mercury/Mercury/Interpreting/[Entries.cs](#)

6.24 Mercury.Exceptions.ErrorActionException Class Reference

Exception thrown by basic actions when a problem was encountered.

Inheritance diagram for Mercury.Exceptions.ErrorActionException:



Public Member Functions

- [ErrorActionException](#) (string message)
Initializes a new instance of the [ErrorActionException](#) class.
- [ErrorActionException](#) (string message, params object[] args)
Initializes a new instance of the [ErrorActionException](#) class.

Additional Inherited Members

6.24.1 Detailed Description

Exception thrown by basic actions when a problem was encountered.

Definition at line 178 of file InterpretingExceptions.cs.

6.24.2 Constructor & Destructor Documentation

6.24.2.1 Mercury.Exceptions.ErrorActionException.ErrorActionException (string message)

Initializes a new instance of the [ErrorActionException](#) class.

Parameters

<i>message</i>	The message that describes the error.
----------------	---------------------------------------

Definition at line 183 of file InterpretingExceptions.cs.

6.24.2.2 Mercury.Exceptions.ErrorActionException.ErrorActionException (string message, params object[] args)

Initializes a new instance of the [ErrorActionException](#) class.

Parameters

<i>message</i>	The message that describes the error.
<i>args</i>	Objects to format.

Definition at line 190 of file InterpretingExceptions.cs.

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

- Mercury/Mercury/Exceptions/[InterpretingExceptions.cs](#)

6.25 Mercury.Extensions Class Reference

Provides extension methods for the [Mercury](#) library.

Static Public Member Functions

- static bool [Power2](#) (int x)
Checks whether the given number is a power of two
- static int [IntegralPow](#) (int a, uint exp)
Computes an integral power of given number
- static string [UniqueCharSeq](#) ()
Creates a unique sequence of [A-Za-z] characters.
- static string [RandomString](#) (int length, string init="")
Creates a random [A-Za-z] string.
- static string [RandomString](#) (string pattern)
Creates a random string according to the pattern. The pattern can contain any characters that will remain unchanged, except the following which will be replaced:
 - . - a random [A-Za-z] character
 - % - a random digit
 - * - a unique sequence of [A-Za-z0-9] chars (useful when generating random names)
- static IList< [Token](#) > [Tokenize](#) (this string src)
Uses defaultTokenizer (Mercury.Nucleus.Tokens.Tokenizer) to tokenize string
- static string [Unescape](#) (this string src)
Removes '\ ' from the string more efficiently than src. Where (x => x != '\')

- static IDictionary< TKey, TValue > [ToMultiDictionary](#)< TSource, TKey, TValue > (this IEnumerable< TSource > source, Func< TSource, TKey > keySelector, Func< TSource, TValue > valueSelector)
Creates a multidictionary from the sequence of values
- static IDictionary< TKey, TSource > [ToMultiDictionary](#)< TSource, TKey > (this IEnumerable< TSource > source, Func< TSource, TKey > keySelector)
Creates a MultiDictionary from the sequence of values
- static T [ElementAtOrLast](#)< T > (this IReadOnlyList< T > source, int index)
Returns the element specified by the index or the last element if the index is greater than number of elements

6.25.1 Detailed Description

Provides extension methods for the [Mercury](#) library.

Definition at line 12 of file Extensions.cs.

6.25.2 Member Function Documentation

6.25.2.1 static T [Mercury.Extensions.ElementAtOrLast](#)< T > (this IReadOnlyList< T > *source*, int *index*) [static]

Returns the element specified by the index or the last element if the index is greater than number of elements

Template Parameters

<i>T</i>	Type of elements
----------	------------------

Parameters

<i>source</i>	The source list
<i>index</i>	The index

Returns

Element

Definition at line 233 of file Extensions.cs.

6.25.2.2 static int [Mercury.Extensions.IntegralPow](#) (int *a*, uint *exp*) [static]

Computes an integral power of given number

Parameters

<i>a</i>	Integer to exponentiate.
<i>exp</i>	The exponent.

Returns

a to the power of *exp* .

Adapted from <http://stackoverflow.com/a/383596>

Definition at line 90 of file Extensions.cs.

6.25.2.3 static bool [Mercury.Extensions.Power2](#) (int *x*) [static]

Checks whether the given number is a power of two

Parameters

<i>x</i>	The number
----------	------------

Returns

`true` iff the number is power of 2

Definition at line 82 of file Extensions.cs.

6.25.2.4 `static string Mercury.Extensions.RandomString ( int length, string init = " " ) [static]`

Creates a random `[A-Za-z]` string.

Parameters

<i>length</i>	The length of the string.
<i>init</i>	Initial value. Not included in the <i>length</i> .

Returns

A string of random `[A-Za-z]` characters.

Definition at line 115 of file Extensions.cs.

6.25.2.5 `static string Mercury.Extensions.RandomString ( string pattern ) [static]`

Creates a random string according to the pattern. The pattern can contain any characters that will remain unchanged, except the following which will be replaced:

`.` - a random `[A-Za-z]` character

`%` - a random digit

`*` - a unique sequence of `[A-Za-z0-9]` chars (useful when generating random names)

Parameters

<i>pattern</i>	The pattern
----------------	-------------

Returns

Definition at line 135 of file Extensions.cs.

6.25.2.6 `static IList<Token> Mercury.Extensions.Tokenize ( this string src ) [static]`

Uses defaultTokenizer (Mercury.Nucleus.Tokens.Tokenizer) to tokenize string

Parameters

<i>src</i>	this string
------------	-------------

Returns

token values

Definition at line 161 of file Extensions.cs.

6.25.2.7 `static IDictionary<TKey, TSource> Mercury.Extensions.ToMultiDictionary< TSource, TKey > (this
IEnumerable< TSource > source, Func< TSource, TKey > keySelector) [static]`

Creates a MultiDictionary from the sequence of values

Template Parameters

<i>TSource</i>	The type of the source value
<i>TKey</i>	The type of the key

Parameters

<i>source</i>	The source sequence
<i>keySelector</i>	The key selector function

Returns

Definition at line 216 of file Extensions.cs.

6.25.2.8 `static IDictionary<TKey, TValue> Mercury.Extensions.ToMultiDictionary< TSource, TKey, TValue > ( this IEnumerable< TSource > source, Func< TSource, TKey > keySelector, Func< TSource, TValue > valueSelector )` `[static]`

Creates a multidictionary from the sequence of values

Template Parameters

<i>TSource</i>	The type of the source values
<i>TKey</i>	The type of the key
<i>TValue</i>	The type of the values

Parameters

<i>source</i>	The source enumeration
<i>keySelector</i>	The key selector function
<i>valueSelector</i>	The value selector function

Returns

The MultiDictionary

Definition at line 197 of file Extensions.cs.

6.25.2.9 `static string Mercury.Extensions.Unescape ( this string src )` `[static]`

Removes ``` from the string more efficiently than `src.Where(x => x != `)`

Parameters

<i>src</i>	The source string
------------	-------------------

Returns

The string without ``` characters

Definition at line 169 of file Extensions.cs.

6.25.2.10 `static string Mercury.Extensions.UniqueCharSeq ( )` `[static]`

Creates a unique sequence of `[A-Za-z]` characters.

Returns

A unique equence.

Definition at line 108 of file Extensions.cs.

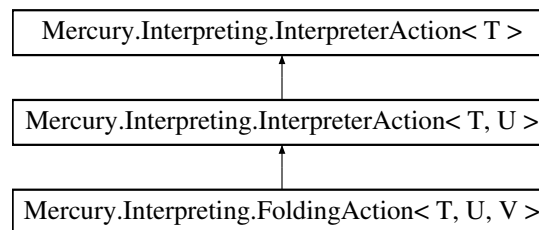
The documentation for this class was generated from the following file:

- Mercury/Mercury/[Extensions.cs](#)

6.26 Mercury.Interpreting.FoldingAction< T, U, V > Class Template Reference

"Folds" all its arguments into one value. For instance, for "x y z" arguments this action computes $f(f(f(s, x), y), z)$ where f is the lamda function and s is the seed parameter of the constructor

Inheritance diagram for Mercury.Interpreting.FoldingAction< T, U, V >:

**Public Member Functions**

- [FoldingAction](#) (string name, int minargs, Func< V, U, V > f, V seed, [ArgumentType](#) flags=ArgumentType.↔ None)
Initializes a new instance of the FoldingAction{T, U, V} class.
- override V [Evaluate](#) ([Argument](#)[] arguments, RecursiveEval< T > eval, InterpreterContext< T > context)
Evaluates the action.

Additional Inherited Members**6.26.1 Detailed Description**

"Folds" all its arguments into one value. For instance, for "x y z" arguments this action computes $f(f(f(s, x), y), z)$ where f is the lamda function and s is the seed parameter of the constructor

Template Parameters

<i>T</i>	Interpreter return type
<i>U</i>	Type of values
<i>V</i>	Output type

Type Constraints

***T* : class**

Definition at line 415 of file BasicActions.cs.

6.26.2 Constructor & Destructor Documentation

6.26.2.1 `Mercury.Interpreting.FoldingAction< T, U, V >.FoldingAction ( string name, int minargs, Func< V, U, V > f, V seed, ArgumentType flags = ArgumentType.None )`

Initializes a new instance of the `FoldingAction{T, U, V}` class.

Parameters

<i>name</i>	The name of the action
<i>minargs</i>	Minimal number of arguments
<i>f</i>	The lambda function
<i>seed</i>	Initial value
<i>flags</i>	The flags.

Exceptions

<i>System.ArgumentNull</i> <i>Exception</i>	<i>f</i>
--	----------

Definition at line 432 of file BasicActions.cs.

6.26.3 Member Function Documentation

6.26.3.1 `override V Mercury.Interpreting.FoldingAction< T, U, V >.Evaluate ( Argument[] arguments, RecursiveEval< T > eval, InterpreterContext< T > context ) [virtual]`

Evaluates the action.

Parameters

<i>arguments</i>	Arguments provided by the interpreter.
<i>eval</i>	The recursive evaluation delegate.
<i>context</i>	The interpreter context.

Returns

Implements [Mercury.Interpreting.InterpreterAction< T, U >](#).

Definition at line 446 of file BasicActions.cs.

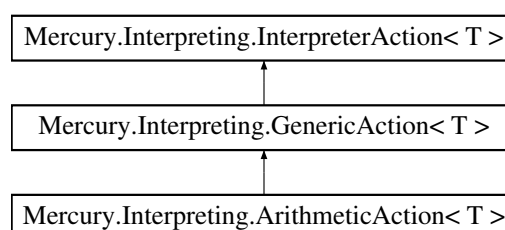
The documentation for this class was generated from the following file:

- [Mercury/Mercury/Interpreting/BasicActions.cs](#)

6.27 Mercury.Interpreting.GenericAction< T > Class Template Reference

Represents an action that infers its return type only after it is provided with types of its arguments.

Inheritance diagram for Mercury.Interpreting.GenericAction< T >:



Public Member Functions

- [GenericAction](#) (string name, Tuple< int, int > arity, [ArgumentType](#)[] argtypes, bool allowswild=true)

Initializes a `GenericAction{T}` instance.

- abstract [ArgumentType](#) [InferReturnType](#) ([ArgumentType](#) expected, [ArgumentType\[\]](#) actual)
Infers the actual return type from provided parameter types

Additional Inherited Members

6.27.1 Detailed Description

Represents an action that infers its return type only after it is provided with types of its arguments.

Template Parameters

<i>T</i>	Interpreter output type.
----------	--------------------------

Type Constraints

***T* : class**

Definition at line 214 of file `InterpreterAction.cs`.

6.27.2 Constructor & Destructor Documentation

6.27.2.1 `Mercury.Interpreting.GenericAction< T >.GenericAction ( string name, Tuple< int, int > arity, ArgumentType[] argtypes, bool allowswild = true )`

Initializes a `GenericAction{T}` instance.

Parameters

<i>name</i>	The action name.
<i>arity</i>	The action arity.
<i>argtypes</i>	The argument types mask
<i>allowswild</i>	Indicates whether this action can accept wildcards.

Definition at line 226 of file `InterpreterAction.cs`.

6.27.3 Member Function Documentation

6.27.3.1 `abstract ArgumentType Mercury.Interpreting.GenericAction< T >.InferReturnType ( ArgumentType expected, ArgumentType[] actual ) [pure virtual]`

Infers the actual return type from provided parameter types

Parameters

<i>expected</i>	Expected return type flags.
<i>actual</i>	Actual argument types.

Returns

The actual return type. Must contain exactly ONE bit set, unless it is `Integer` | `Real`

Implemented in [Mercury.Interpreting.ArithmeticAction< T >](#).

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

- [Mercury/Mercury/Interpreting/InterpreterAction.cs](#)

6.28 Mercury.Syntax.Grammar Class Reference

Represents a Context-Free [Grammar](#). This is an immutable class, to create a grammar use [Mercury.Syntax.←GrammarBuilder](#).

Inherited by [Mercury.Syntax.ExtendedGrammar](#).

Properties

- [IReadOnlyList<Symbol> Nonterminals](#) [get, set]
List of nonterminal symbols.
- [IReadOnlyList<Symbol> Terminals](#) [get, set]
List of terminal symbols.
- [IReadOnlyList<Rule> Rules](#) [get, set]
List of rules.
- [Symbol Root](#) [get, set]
The root symbol.
- [Symbol Epsilon](#) [get, set]
The epsilon symbol.

6.28.1 Detailed Description

Represents a Context-Free [Grammar](#). This is an immutable class, to create a grammar use [Mercury.Syntax.←GrammarBuilder](#).

Definition at line 18 of file Grammar.cs.

6.28.2 Property Documentation

6.28.2.1 [Symbol Mercury.Syntax.Grammar.Epsilon](#) [get], [set]

The epsilon symbol.

Epsilon

Definition at line 37 of file Grammar.cs.

6.28.2.2 [IReadOnlyList<Symbol> Mercury.Syntax.Grammar.Nonterminals](#) [get], [set]

List of nonterminal symbols.

Definition at line 23 of file Grammar.cs.

6.28.2.3 [Symbol Mercury.Syntax.Grammar.Root](#) [get], [set]

The root symbol.

Root

Definition at line 33 of file Grammar.cs.

6.28.2.4 [IReadOnlyList<Rule> Mercury.Syntax.Grammar.Rules](#) [get], [set]

List of rules.

Definition at line 29 of file Grammar.cs.

6.28.2.5 IReadOnlyList<Symbol> Mercury.Syntax.Grammar.Terminals [get], [set]

List of terminal symbols.

Definition at line 26 of file Grammar.cs.

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

- Mercury/Mercury/Syntax/[Grammar.cs](#)

6.29 Mercury.Syntax.GrammarBuilder Class Reference

Class that creates grammars.

Inherited by Mercury.Syntax.ExtendedGrammarBuilder.

Public Member Functions

- [GrammarBuilder](#) ()
Default constructor. Uses Root and Epsilon symbols that are specified in the configuration.
- [GrammarBuilder](#) ([Symbol](#) root, [Symbol](#) epsilon)
Creates a [GrammarBuilder](#) with custom Root and Epsilon symbols.
- [GrammarBuilder AddSymbol](#) ([Symbol](#) newSymbol)
Adds a new symbol to the grammar. The destination set is determined by the symbol type. Epsilon or symbols that already are in the grammar are ignored.
- virtual [GrammarBuilder AddRule](#) ([Rule](#) rule)
Adds a new rule to the grammar. If the rule contains symbols that are not in the grammar, they will be added. Rules that are already in the grammar will be ignored.
- [GrammarBuilder AddRules](#) (IEnumerable< [Rule](#) > rules)
Adds the collection of rules to the builder.
- [GrammarBuilder Include](#) ([GrammarBuilder](#) other)
Includes the other builder.
- [Grammar GetGrammar](#) ()
Returns
A grammar from the builder.

Protected Attributes

- HashSet< [Symbol](#) > nonterminals = new HashSet<[Symbol](#)>()
- HashSet< [Symbol](#) > terminals = new HashSet<[Symbol](#)>()
- HashSet< [Rule](#) > rules = new HashSet<[Rule](#)>()

Properties

- [Symbol Root](#) [get, set]
Root symbol.
- [Symbol Epsilon](#) [get, set]
Epsilon symbol.

6.29.1 Detailed Description

Class that creates grammars.

Definition at line 13 of file GrammarBuilder.cs.

6.29.2 Constructor & Destructor Documentation

6.29.2.1 Mercury.Syntax.GrammarBuilder.GrammarBuilder ()

Default constructor. Uses Root and Epsilon symbols that are specified in the configuration.

Definition at line 27 of file GrammarBuilder.cs.

6.29.2.2 Mercury.Syntax.GrammarBuilder.GrammarBuilder (Symbol root, Symbol epsilon)

Creates a [GrammarBuilder](#) with custom Root and Epsilon symbols.

Parameters

<i>root</i>	A root symbol, must be a nonterminal.
<i>epsilon</i>	Epsilon symbol.

Definition at line 36 of file GrammarBuilder.cs.

6.29.3 Member Function Documentation

6.29.3.1 virtual GrammarBuilder Mercury.Syntax.GrammarBuilder.AddRule (Rule rule) [virtual]

Adds a new rule to the grammar. If the rule contains symbols that are not in the grammar, they will be added. Rules that are already in the grammar will be ignored.

Parameters

<i>rule</i>	The rule to be added
-------------	----------------------

Returns

This builder to support chaining.

Definition at line 93 of file GrammarBuilder.cs.

6.29.3.2 GrammarBuilder Mercury.Syntax.GrammarBuilder.AddRules (IEnumerable< Rule > rules)

Adds the collection of rules to the builder.

Parameters

<i>rules</i>	The rules.
--------------	------------

Returns

This builder to support chaining.

Definition at line 111 of file GrammarBuilder.cs.

6.29.3.3 GrammarBuilder Mercury.Syntax.GrammarBuilder.AddSymbol (Symbol newSymbol)

Adds a new symbol to the grammar. The destination set is determined by the symbol type. Epsilon or symbols that already are in the grammar are ignored.

Parameters

<i>newSymbol</i>	The new symbol.
------------------	-----------------

Returns

This builder to support chaining.

Definition at line 67 of file GrammarBuilder.cs.

6.29.3.4 Grammar Mercury.Syntax.GrammarBuilder.GetGrammar ()**Returns**

A grammar from the builder.

Definition at line 121 of file GrammarBuilder.cs.

6.29.3.5 GrammarBuilder Mercury.Syntax.GrammarBuilder.Include (GrammarBuilder other)

Includes the other builder.

Parameters

<i>other</i>	The other builder.
--------------	--------------------

Returns

This builder to support chaining.

Definition at line 117 of file GrammarBuilder.cs.

6.29.4 Member Data Documentation**6.29.4.1 HashSet<Symbol> Mercury.Syntax.GrammarBuilder.nonterminals = new HashSet<Symbol>() [protected]**

Definition at line 17 of file GrammarBuilder.cs.

6.29.4.2 HashSet<Rule> Mercury.Syntax.GrammarBuilder.rules = new HashSet<Rule>() [protected]

Definition at line 19 of file GrammarBuilder.cs.

6.29.4.3 HashSet<Symbol> Mercury.Syntax.GrammarBuilder.terminals = new HashSet<Symbol>() [protected]

Definition at line 18 of file GrammarBuilder.cs.

6.29.5 Property Documentation**6.29.5.1 Symbol Mercury.Syntax.GrammarBuilder.Epsilon [get], [set]**

Epsilon symbol.

Definition at line 56 of file GrammarBuilder.cs.

6.29.5.2 Symbol Mercury.Syntax.GrammarBuilder.Root [get], [set]

Root symbol.

Definition at line 53 of file GrammarBuilder.cs.

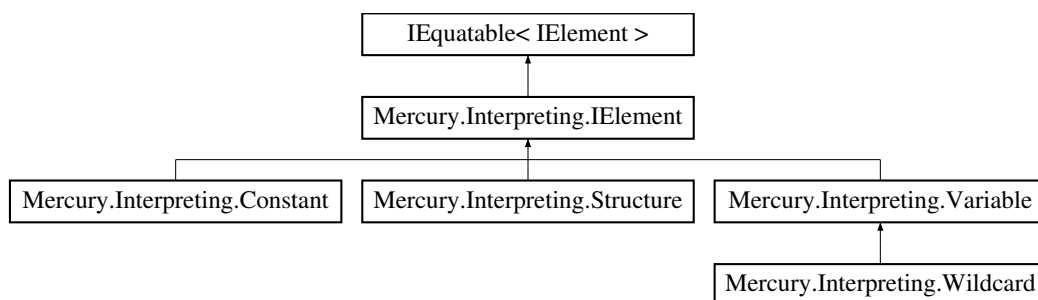
The documentation for this class was generated from the following file:

- Mercury/Mercury/Syntax/[GrammarBuilder.cs](#)

6.30 Mercury.Interpreting.IElement Interface Reference

Element interface

Inheritance diagram for Mercury.Interpreting.IElement:



Properties

- [ElementType Type](#) [get]
Element type

6.30.1 Detailed Description

Element interface

Definition at line 32 of file Element.cs.

6.30.2 Property Documentation

6.30.2.1 ElementType Mercury.Interpreting.IElement.Type [get]

Element type

Definition at line 37 of file Element.cs.

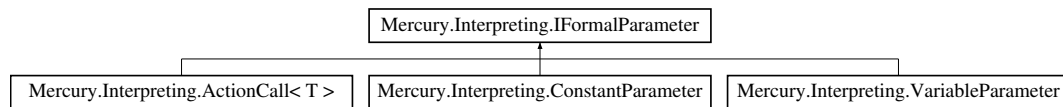
The documentation for this interface was generated from the following file:

- Mercury/Mercury/Interpreting/[Element.cs](#)

6.31 Mercury.Interpreting.IFormalParameter Interface Reference

Represents formal parameters.

Inheritance diagram for Mercury.Interpreting.IFormalParameter:



Properties

- [ParameterType Type](#) [get]
Formal parameter type.

6.31.1 Detailed Description

Represents formal parameters.

Definition at line 32 of file FormalParameter.cs.

6.31.2 Property Documentation

6.31.2.1 ParameterType Mercury.Interpreting.IFormalParameter.Type [get]

Formal parameter type.

Definition at line 35 of file FormalParameter.cs.

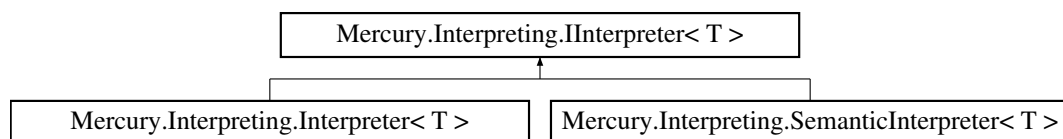
The documentation for this interface was generated from the following file:

- Mercury/Mercury/Interpreting/[FormalParameter.cs](#)

6.32 Mercury.Interpreting.IInterpreter< T > Interface Template Reference

Interface for interpreters that create objects from derivation trees according to RewriteRules.

Inheritance diagram for Mercury.Interpreting.IInterpreter< T >:



Public Member Functions

- InterpreterResult< T > [Interpret](#) ([Tree](#)< [Symbol](#) > tree)
Uses rewrite rules to create a result value from the tree
- IReadOnlyList
< InterpreterResult< T > > [Interpret](#) (IReadOnlyList< [Tree](#)< [Symbol](#) >> trees)
Interprets the list of trees. This is to support memoization and/or parallelisation.

Properties

- RewriteRuleCollection< T > [RewriteRules](#) [get]
The collection of rewrite rules used by the interpreter

6.32.1 Detailed Description

Interface for interpreters that create objects from derivation trees according to RewriteRules.

Template Parameters

<i>T</i>	Interpreter output type.
----------	--------------------------

Type Constraints

***T* : class**

Definition at line 20 of file Interpreter.cs.

6.32.2 Member Function Documentation**6.32.2.1 InterpreterResult<T> Mercury.Interpreting.IInterpreter< T >.Interpret (Tree< Symbol > tree)**

Uses rewrite rules to create a result value from the tree

Parameters

<i>tree</i>	The derivation tree.
-------------	----------------------

Returns

Interpretation of the tree or `null` if interpretation failed.Implemented in [Mercury.Interpreting.SemanticInterpreter< T >](#), and [Mercury.Interpreting.IInterpreter< T >](#).**6.32.2.2 IReadOnlyList<InterpreterResult<T> > Mercury.Interpreting.IInterpreter< T >.Interpret (IReadOnlyList< Tree< Symbol >> trees)**

Interprets the list of trees. This is to support memoization and/or parallelisation.

Parameters

<i>trees</i>	The trees.
--------------	------------

Returns

List of results

Implemented in [Mercury.Interpreting.SemanticInterpreter< T >](#), and [Mercury.Interpreting.IInterpreter< T >](#).**6.32.3 Property Documentation****6.32.3.1 RewriteRuleCollection<T> Mercury.Interpreting.IInterpreter< T >.RewriteRules [get]**

The collection of rewrite rules used by the interpreter

Definition at line 23 of file Interpreter.cs.

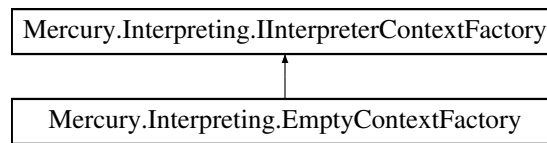
The documentation for this interface was generated from the following file:

- Mercury/Mercury/Interpreting/[Interpreter.cs](#)

6.33 Mercury.Interpreting.IInterpreterContextFactory Interface Reference

Factory of context.

Inheritance diagram for Mercury.Interpreting.IInterpreterContextFactory:



Public Member Functions

- `InterpreterContext< T > CreateContext< T > ()`
Creates a new context.

6.33.1 Detailed Description

Factory of context.

Definition at line 89 of file InterpreterContext.cs.

6.33.2 Member Function Documentation

6.33.2.1 `InterpreterContext<T> Mercury.Interpreting.IInterpreterContextFactory.CreateContext< T > ( )`

Creates a new context.

Template Parameters

<i>T</i>	Interpreter output type.
----------	--------------------------

Returns

A new context.

Implemented in [Mercury.Interpreting.EmptyContextFactory](#).

Type Constraints

***T* : class**

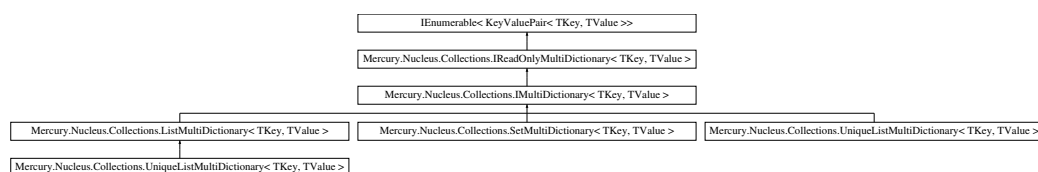
The documentation for this interface was generated from the following file:

- [Mercury/Mercury/Interpreting/InterpreterContext.cs](#)

6.34 Mercury.Nucleus.Collections.IMultiDictionary< TKey, TValue > Interface Template Reference

Represents a multi-valued dictionary, in other words a map between a key and a collection of values.

Inheritance diagram for Mercury.Nucleus.Collections.IMultiDictionary< TKey, TValue >:



Public Member Functions

- bool [Add](#) (TKey key, TValue value)
Adds the value to the dictionary. The actual behaviour and handling of duplicities is specified by an implementation.
- bool [Add](#) (KeyValuePair< TKey, TValue > tuple)
Adds the value to the dictionary. The actual behaviour and handling of duplicities is specified by an implementation.
- bool [Add](#) (TKey key, IEnumerable< TValue > values)
*Adds the collection of values to the dictionary. If the key is already present, NO value is added and *false* will be returned.*
- void [Clear](#) ()
Removes the content of collection.

Additional Inherited Members

6.34.1 Detailed Description

Represents a multi-valued dictionary, in other words a map between a key and a collection of values.

Template Parameters

<i>TKey</i>	The type of the key
<i>TValue</i>	The type of values

Definition at line 64 of file MultiDictionary.cs.

6.34.2 Member Function Documentation

6.34.2.1 bool Mercury.Nucleus.Collections.IMultiDictionary< TKey, TValue >.Add (TKey key, TValue value)

Adds the value to the dictionary. The actual behaviour and handling of duplicities is specified by an implementation.

Parameters

<i>key</i>	The key.
<i>value</i>	The value.

Returns

`true` if value added successfully, `false` otherwise (depends on the implementation).

Exceptions

<i>ArgumentNullException</i>	The key is null.
------------------------------	------------------

Implemented in [Mercury.Nucleus.Collections.UniqueListMultiDictionary< TKey, TValue >](#), [Mercury.Nucleus.Collections.ListMultiDictionary< TKey, TValue >](#), and [Mercury.Nucleus.Collections.SetMultiDictionary< TKey, TValue >](#).

6.34.2.2 bool Mercury.Nucleus.Collections.IMultiDictionary< TKey, TValue >.Add (KeyValuePair< TKey, TValue > tuple)

Adds the value to the dictionary. The actual behaviour and handling of duplicities is specified by an implementation.

Parameters

<i>tuple</i>	The tuple of key and value.
--------------	-----------------------------

Returns

`true` if value added successfully, `false` otherwise (depends on the implementation).

Exceptions

<i>ArgumentNullException</i>	The tuple is null.
------------------------------	--------------------

Implemented in [Mercury.Nucleus.Collections.ListMultiDictionary< TKey, TValue >](#), and [Mercury.Nucleus.Collections.SetMultiDictionary< TKey, TValue >](#).

6.34.2.3 `bool Mercury.Nucleus.Collections.IMultiDictionary< TKey, TValue >.Add ( TKey key, IEnumerable< TValue > values )`

Adds the collection of values to the dictionary. If the *key* is already present, NO value is added and `false` will be returned.

Parameters

<i>key</i>	The key.
<i>values</i>	The collection of values.

Returns

`true` if *key* is not present yet and at least one value was successfully added; `false` otherwise.

Exceptions

<i>ArgumentNullException</i>	The key is null.
------------------------------	------------------

Implemented in [Mercury.Nucleus.Collections.ListMultiDictionary< TKey, TValue >](#), and [Mercury.Nucleus.Collections.SetMultiDictionary< TKey, TValue >](#).

6.34.2.4 `void Mercury.Nucleus.Collections.IMultiDictionary< TKey, TValue >.Clear ( )`

Removes the content of collection.

Implemented in [Mercury.Nucleus.Collections.UniqueListMultiDictionary< TKey, TValue >](#), [Mercury.Nucleus.Collections.ListMultiDictionary< TKey, TValue >](#), and [Mercury.Nucleus.Collections.SetMultiDictionary< TKey, TValue >](#).

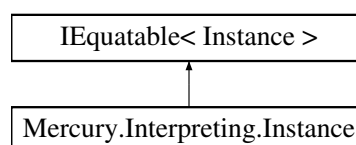
The documentation for this interface was generated from the following file:

- [Mercury/Mercury/Nucleus/Collections/MultiDictionary.cs](#)

6.35 Mercury.Interpreting.Instance Class Reference

Represents a binding between a [Mercury.Interpreting.Variable](#) and its value

Inheritance diagram for Mercury.Interpreting.Instance:



Public Member Functions

- override bool [Equals](#) (object obj)
- override int [GetHashCode](#) ()
- bool [Equals](#) ([Instance](#) other)

Properties

- [Variable Var](#) [get, set]
The variable
- object [Value](#) [get, set]
Bound value

6.35.1 Detailed Description

Represents a binding between a [Mercury.Interpreting.Variable](#) and its value

Definition at line 17 of file Instance.cs.

6.35.2 Member Function Documentation

6.35.2.1 override bool Mercury.Interpreting.Instance.Equals (object obj)

Definition at line 53 of file Instance.cs.

6.35.2.2 bool Mercury.Interpreting.Instance.Equals (Instance other)

Definition at line 68 of file Instance.cs.

6.35.2.3 override int Mercury.Interpreting.Instance.GetHashCode ()

Definition at line 59 of file Instance.cs.

6.35.3 Property Documentation

6.35.3.1 object Mercury.Interpreting.Instance.Value [get], [set]

Bound value

Definition at line 47 of file Instance.cs.

6.35.3.2 Variable Mercury.Interpreting.Instance.Var [get], [set]

The variable

Definition at line 44 of file Instance.cs.

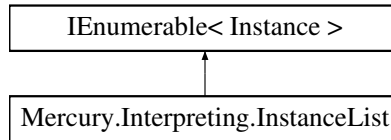
The documentation for this class was generated from the following file:

- Mercury/Mercury/Interpreting/[Instance.cs](#)

6.36 Mercury.Interpreting.InstanceList Class Reference

Represents the list of instances

Inheritance diagram for Mercury.Interpreting.InstanceList:



Public Member Functions

- `IEnumerator< Instance > GetEnumerator ()`

Properties

- `int Count` [get]
Number of elements in the list
- `Instance this[string name]` [get]
Finds the [Instance](#) with the specified name.

6.36.1 Detailed Description

Represents the list of instances

Definition at line 94 of file Instance.cs.

6.36.2 Member Function Documentation

6.36.2.1 `IEnumerator<Instance> Mercury.Interpreting.InstanceList.GetEnumerator ( )`

Definition at line 164 of file Instance.cs.

6.36.3 Property Documentation

6.36.3.1 `int Mercury.Interpreting.InstanceList.Count` [get]

Number of elements in the list

Definition at line 112 of file Instance.cs.

6.36.3.2 `Instance Mercury.Interpreting.InstanceList.this[string name]` [get]

Finds the [Instance](#) with the specified name.

The [Instance](#).

Parameters

<i>name</i>	The name of variable.
-------------	-----------------------

Returns

The instance.

Definition at line 120 of file Instance.cs.

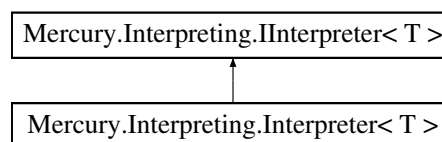
The documentation for this class was generated from the following file:

- Mercury/Mercury/Interpreting/[Instance.cs](#)

6.37 Mercury.Interpreting.Interpreter< T > Class Template Reference

Interprets derivation trees using rewrite rules. The type parameter *T* represents the target language.

Inheritance diagram for Mercury.Interpreting.Interpreter< T >:



Public Member Functions

- [Interpreter](#) (RewriteRuleCollection< T > rules, [IInterpreterContextFactory](#) factory, bool logging=true, int boostlim=4)
Creates a new [Interpreter](#) instance.
- InterpreterResult< T > [Interpret](#) (Tree< [Symbol](#) > tree)
Interprets the tree using rewrite rules.
- IReadOnlyList< InterpreterResult< T > > [Interpret](#) (IReadOnlyList< Tree< [Symbol](#) >> trees)
Interprets the specified trees. If the number of trees is at least `boostlim` specified in the constructor, the method will use memoization and parallelism.

Properties

- RewriteRuleCollection< T > [RewriteRules](#) [get, set]
The collection of rewrite rules.

6.37.1 Detailed Description

Interprets derivation trees using rewrite rules. The type parameter *T* represents the target language.

Achtung: if *T* equals `dynamic`, the type check will assume all possible values on arguments and actions that take (or return for that matter) `TValue`.

The method `Interpret` parametrized with lists uses parallelism and memory caching. If number of interpreted trees is at least `boostlim`, the method will use `System.Runtime.Caching.MemoryCache` and `System.Threading.Tasks.Parallel` components to speed up the analysis. The value `boostlim` is 4 by default.

Type Constraints

***T* : class**

Definition at line 57 of file `Interpreter.cs`.

6.37.2 Constructor & Destructor Documentation

6.37.2.1 `Mercury.Interpreting.Interpreter< T >.Interpreter ( RewriteRuleCollection< T > rules, IInterpreterContextFactory factory, bool logging = true, int boostlim = 4 )`

Creates a new [Interpreter](#) instance.

Parameters

<i>rules</i>	A collection of rewrite rules.
<i>factory</i>	The context factory or <code>null</code> to use the default factory.
<i>logging</i>	If <code>false</code> , the interpreter will throw away logging structure to save memory
<i>boostlim</i>	Specifies the amount of trees required for the method <code>Interpret</code> on list of trees to use memoisation and parallelism. The default value is 4.

Exceptions

<i>System.ArgumentNullException</i>	When rules reference is null.
-------------------------------------	-------------------------------

Definition at line 367 of file `Interpreter.cs`.

6.37.3 Member Function Documentation

6.37.3.1 `InterpreterResult<T> Mercury.Interpreting.Interpreter< T >.Interpret ( Tree< Symbol > tree )`

Interprets the tree using rewrite rules.

Parameters

<i>tree</i>	The tree to interpret.
-------------	------------------------

Returns

Result of the interpretation.

Implements [Mercury.Interpreting.IInterpreter< T >](#).

Definition at line 391 of file `Interpreter.cs`.

6.37.3.2 `IReadOnlyList<InterpreterResult<T> > Mercury.Interpreting.Interpreter< T >.Interpret ( IReadOnlyList< Tree< Symbol > > trees )`

Interprets the specified trees. If the number of trees is at least `boostlim` specified in the constructor, the method will use memoization and parallelism.

Parameters

<i>trees</i>	The list of trees.
--------------	--------------------

Returns

List of results.

Implements [Mercury.Interpreting.IInterpreter< T >](#).

Definition at line 401 of file Interpreter.cs.

6.37.4 Property Documentation

6.37.4.1 RewriteRuleCollection<T> Mercury.Interpreting.Interpreter< T >.RewriteRules [get], [set]

The collection of rewrite rules.

Definition at line 382 of file Interpreter.cs.

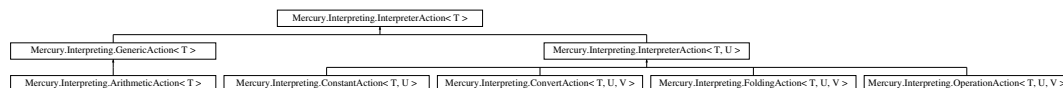
The documentation for this class was generated from the following file:

- Mercury/Mercury/Interpreting/[Interpreter.cs](#)

6.38 Mercury.Interpreting.InterpreterAction< T > Class Template Reference

Function that wraps constant values in the RHS

Inheritance diagram for Mercury.Interpreting.InterpreterAction< T >:



Public Member Functions

- [InterpreterAction](#) (string name, Tuple< int, int > arity, [ArgumentType](#)[] argtypes, bool allowswild=true)

Initializes a new instance of the InterpreterAction{T} class. The target application can inherit from this class, but rather one of Mercury.Interpreting.InterpreterAction{T,U} or Mercury.Interpreting.GenericAction{T} is recommended to derive.

Conditions:

- abstract object [Invoke](#) ([Argument](#)[] arguments, RecursiveEval< T > eval, InterpreterContext< T > context)

Evaluates the arguments.

It is guaranteed to have a valid number of arguments (specified by [Arity](#)) and that no argument is null UNLESS the flag Mercury.Interpreting.ArgumentType contains Lazy or Null.

Protected Member Functions

- void [CheckDefinition](#) ()

Properties

- string [Name](#) [get, set]
Action name.
- Tuple< int, int > [Arity](#) [get, set]
Minimum and maximum number of arguments.
- [ArgumentType](#) [ReturnType](#) [get, set]
Action return type.
- [ArgumentType](#)[] [ArgumentTypes](#) [get, set]
Allowed types of arguments. If more arguments specified (within range of Arity), the last type in this array will be used. It can be empty if and only if arity is (0,0).
- bool [AllowsWildcard](#) [get, set]
Indicates whether a [Mercury.Interpreting.Wildcard](#) can be an argument of this action.

6.38.1 Detailed Description

Function that wraps constant values in the RHS

Template Parameters

<i>T</i>	Interpreter output type
----------	-------------------------

Parameters

<i>args</i>	The arguments
<i>eval</i>	The recursive eval function

Returns

Function that wraps multiple action calls in the RHS into one

Template Parameters

<i>T</i>	
----------	--

Parameters

<i>args</i>	The arguments.
-------------	----------------

Returns

Represents Actions taken in order to interpret trees.

Type Constraints

***T* : class**

Definition at line 46 of file InterpreterAction.cs.

6.38.2 Constructor & Destructor Documentation

6.38.2.1 Mercury.Interpreting.InterpreterAction< T >.InterpreterAction (string *name*, Tuple< int, int > *arity*, ArgumentType[] *argtypes*, bool *allowswild* = true)

Initializes a new instance of the InterpreterAction{T} class. The target application can inherit from this class, but rather one of Mercury.Interpreting.InterpreterAction{T,U} or Mercury.Interpreting.GenericAction{T} is recommended

to derive.

Conditions:

1. *name*
 - must not be null, empty nor consist of whitespaces only
2. *arity*
 - *Item1* is less than *Item2*
 - non-negative components
3. *types*
 - non-empty (unless arity is (0, 0))

Parameters

<i>name</i>	The name of the action.
<i>arity</i>	The arity, minimum and maximum number of allowed arguments.
<i>argtypes</i>	The types of arguments. If specified less than required, the last one will be repeated for unmatched arguments.
<i>allowswild</i>	if set to <code>true</code> AND the maximum number of arguments equals <code>System.Int32.MAX_VALUE</code> , the action will support wildcards.

Exceptions

<i>System.ArgumentNull</i> ↳ <i>Exception</i>	When <i>name</i> , <i>arity</i> or <i>argtypes</i> are <code>null</code> .
--	--

Definition at line 94 of file `InterpreterAction.cs`.

6.38.3 Member Function Documentation

6.38.3.1 `void Mercury.Interpreting.InterpreterAction< T >.CheckDefinition ( )` [protected]

Definition at line 51 of file `InterpreterAction.cs`.

6.38.3.2 `abstract object Mercury.Interpreting.InterpreterAction< T >.Invoke ( Argument[] arguments, RecursiveEval< T > eval, InterpreterContext< T > context )` [pure virtual]

Evaluates the arguments.

It is guaranteed to have a valid number of arguments (specified by [Arity](#)) and that no argument is `null` UNLESS the flag [Mercury.Interpreting.ArgumentType](#) contains `Lazy` or `Null`.

Parameters

<i>arguments</i>	Arguments of the function.
<i>eval</i>	The recursive evaluation of trees.
<i>context</i>	Interpreter context.

Returns

String value of evaluation or `null` if failed.

Implemented in [Mercury.Interpreting.ArithmeticAction< T >](#), and [Mercury.Interpreting.InterpreterAction< T, U >](#).

6.38.4 Property Documentation

6.38.4.1 `bool Mercury.Interpreting.InterpreterAction< T >.AllowsWildcard` `[get]`, `[set]`

Indicates whether a [Mercury.Interpreting.Wildcard](#) can be an argument of this action.

Definition at line 132 of file InterpreterAction.cs.

6.38.4.2 `ArgumentType [] Mercury.Interpreting.InterpreterAction< T >.ArgumentTypes` `[get]`, `[set]`

Allowed types of arguments. If more arguments specified (within range of Arity), the last type in this array will be used. It can be empty if and only if arity is (0,0).

Definition at line 126 of file InterpreterAction.cs.

6.38.4.3 `Tuple<int, int> Mercury.Interpreting.InterpreterAction< T >.Arity` `[get]`, `[set]`

Minimum and maximum number of arguments.

Definition at line 116 of file InterpreterAction.cs.

6.38.4.4 `string Mercury.Interpreting.InterpreterAction< T >.Name` `[get]`, `[set]`

Action name.

Definition at line 113 of file InterpreterAction.cs.

6.38.4.5 `ArgumentType Mercury.Interpreting.InterpreterAction< T >.ReturnType` `[get]`, `[set]`

Action return type.

Definition at line 119 of file InterpreterAction.cs.

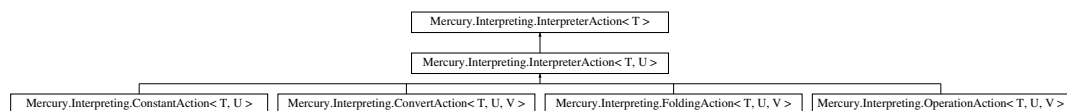
The documentation for this class was generated from the following file:

- [Mercury/Mercury/Interpreting/InterpreterAction.cs](#)

6.39 Mercury.Interpreting.InterpreterAction< T, U > Class Template Reference

Interpreter action with a safer return type.

Inheritance diagram for Mercury.Interpreting.InterpreterAction< T, U >:



Public Member Functions

- [InterpreterAction](#) (string name, Tuple< int, int > arity, [ArgumentType](#)[] argtypes, bool allowswild=true)
Initializes a new instance of the InterpreterAction{T,U} class. See Mercury.Interpreting.InterpreterAction{T} for better explanation.
- abstract U [Evaluate](#) ([Argument](#)[] arguments, RecursiveEval< T > eval, InterpreterContext< T > context)
Evaluates the action.

- override object [Invoke](#) ([Argument](#)[] arguments, RecursiveEval< T > eval, InterpreterContext< T > context)
Evaluates the arguments.
It is guaranteed to have a valid number of arguments (specified by [Arity](#)) and that no argument is `null` UNLESS the flag [Mercury.Interpreting.ArgumentType](#) contains `Lazy` or `Null`.

Additional Inherited Members

6.39.1 Detailed Description

Interpreter action with a safer return type.

Template Parameters

<i>T</i>	Interpreter output type.
<i>U</i>	Action return type.

Type Constraints

***T* : class**

Definition at line 157 of file InterpreterAction.cs.

6.39.2 Constructor & Destructor Documentation

6.39.2.1 [Mercury.Interpreting.InterpreterAction](#)< T, U >.InterpreterAction (string *name*, Tuple< int, int > *arity*, [ArgumentType](#)[] *argtypes*, bool *allowswild* =true)

Initializes a new instance of the [InterpreterAction](#){T,U} class. See [Mercury.Interpreting.InterpreterAction](#){T} for better explanation.

Parameters

<i>name</i>	The name of action.
<i>arity</i>	The arity of action.
<i>argtypes</i>	Argument types.
<i>allowswild</i>	Indicates whether the action can accept wildcards.

Definition at line 170 of file InterpreterAction.cs.

6.39.3 Member Function Documentation

6.39.3.1 abstract U [Mercury.Interpreting.InterpreterAction](#)< T, U >.Evaluate ([Argument](#)[] *arguments*, RecursiveEval< T > *eval*, InterpreterContext< T > *context*) [pure virtual]

Evaluates the action.

Parameters

<i>arguments</i>	Arguments provided by the interpreter.
<i>eval</i>	The recursive evaluation delegate.
<i>context</i>	The interpreter context.

Returns

Implemented in [Mercury.Interpreting.ConstantAction](#)< T, U >, [Mercury.Interpreting.ConvertAction](#)< T, U, V >, [Mercury.Interpreting.FoldingAction](#)< T, U, V >, and [Mercury.Interpreting.OperationAction](#)< T, U, V >.

6.39.3.2 override object Mercury.Interpreting.InterpreterAction< T, U >.Invoke ([Argument\[\] arguments](#), RecursiveEval< T > *eval*, InterpreterContext< T > *context*) [virtual]

Evaluates the arguments.

It is guaranteed to have a valid number of arguments (specified by [Arity](#)) and that no argument is `null` UNLESS the flag [Mercury.Interpreting.ArgumentType](#) contains `Lazy` or `Null`.

Parameters

<i>arguments</i>	Arguments of the function.
<i>eval</i>	The recursive evaluation of trees.
<i>context</i>	Interpreter context.

Returns

String value of evaluation or `null` if failed.

Implements [Mercury.Interpreting.InterpreterAction< T >](#).

Definition at line 197 of file InterpreterAction.cs.

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

- Mercury/Mercury/Interpreting/[InterpreterAction.cs](#)

6.40 Mercury.Interpreting.InterpreterContext< T > Class Template Reference

Interpreter Context class. Some languages may use it to override it and remember data when parse trees are being interpreted.

Inherited by Mercury.Interpreting.InternalContext< T >.

Public Member Functions

- [InterpreterContext](#) ()
Initializes a new instance of the InterpreterContext{T} class.
- abstract [InterpreterContext](#)< T > [DeepClone](#) ()
Creates a deep copy of the object
- abstract void [CopyFrom](#) ([InterpreterContext](#)< T > other)
Copies data from another context
- void [LogEntry](#) (string text, params object[] args)
Logs an event into the history.

Properties

- TreeEntry< T > [RootEntry](#) [get]
Records history (trees, rules, events etc.) that occurred while interpreting a tree. Mercury.Interpreting.EntryWriter{T} can be used to format the data into a text writer.

6.40.1 Detailed Description

Interpreter Context class. Some languages may use it to override it and remember data when parse trees are being interpreted.

Type Constraints

***T* : class**

Definition at line 18 of file InterpreterContext.cs.

6.40.2 Constructor & Destructor Documentation**6.40.2.1 Mercury.Interpreting.InterpreterContext< T >.InterpreterContext ()**

Initializes a new instance of the InterpreterContext{T} class.

Definition at line 55 of file InterpreterContext.cs.

6.40.3 Member Function Documentation**6.40.3.1 abstract void Mercury.Interpreting.InterpreterContext< T >.CopyFrom (InterpreterContext< T > *other*)**
[pure virtual]

Copies data from another context

6.40.3.2 abstract InterpreterContext<T> Mercury.Interpreting.InterpreterContext< T >.DeepClone () [pure virtual]

Creates a deep copy of the object

Returns

The context.

6.40.3.3 void Mercury.Interpreting.InterpreterContext< T >.LogEntry (string *text*, params object[] *args*)

Logs an event into the history.

Parameters

<i>text</i>	Event message.
<i>args</i>	Format arguments.

Definition at line 80 of file InterpreterContext.cs.

6.40.4 Property Documentation**6.40.4.1 TreeEntry<T> Mercury.Interpreting.InterpreterContext< T >.RootEntry** [get]

Records history (trees, rules, events etc.) that occurred while interpreting a tree. Mercury.Interpreting.EntryWriter{T} can be used to format the data into a text writer.

Definition at line 66 of file InterpreterContext.cs.

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

- Mercury/Mercury/Interpreting/[InterpreterContext.cs](#)

6.41 Mercury.Interpreting.InterpreterResult< T > Class Template Reference

Result of the interpretation.

Properties

- bool **Success** [get]
Gets a value indicating whether the interpretation was successful. If the value is false, *Exception* and *Result* fields may be null.
- Tree< Symbol > **Input** [get, set]
A tree that was interpreted.
- T **Result** [get, set]
Result interpretation or null
- Exception **Exception** [get, set]
If an exception was caught during interpretation, it is stored in this field.
- InterpreterContext< T > **Context** [get, set]
Copy of context used for interpretation. Provides *RootEntry* that serves as the interpreter log (see Mercury.↔
Interpreting.InterpreterContext{T}).

6.41.1 Detailed Description

Result of the interpretation.

Template Parameters

<i>T</i>	Interpreter return type.
----------	--------------------------

Type Constraints

***T* : class**

Definition at line 17 of file InterpreterResult.cs.

6.41.2 Property Documentation

6.41.2.1 InterpreterContext<T> Mercury.Interpreting.InterpreterResult< T >.Context [get], [set]

Copy of context used for interpretation. Provides *RootEntry* that serves as the interpreter log (see Mercury.↔
Interpreting.InterpreterContext{T}).

Definition at line 44 of file InterpreterResult.cs.

6.41.2.2 Exception Mercury.Interpreting.InterpreterResult< T >.Exception [get], [set]

If an exception was caught during interpretation, it is stored in this field.

Definition at line 37 of file InterpreterResult.cs.

6.41.2.3 Tree<Symbol> Mercury.Interpreting.InterpreterResult< T >.Input [get], [set]

A tree that was interpreted.

Definition at line 28 of file InterpreterResult.cs.

6.41.2.4 `T Mercury.Interpreting.InterpreterResult< T >.Result` [get], [set]

Result interpretation or `null`

Definition at line 31 of file `InterpreterResult.cs`.

6.41.2.5 `bool Mercury.Interpreting.InterpreterResult< T >.Success` [get]

Gets a value indicating whether the interpretation was successful. If the value is `false`, `Exception` and `Result` fields may be `null`.

Definition at line 25 of file `InterpreterResult.cs`.

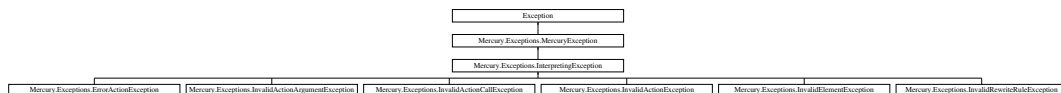
The documentation for this class was generated from the following file:

- `Mercury/Mercury/Interpreting/InterpreterResult.cs`

6.42 Mercury.Exceptions.InterpretingException Class Reference

An exception thrown because of problems in the [Interpreting](#) namespace.

Inheritance diagram for `Mercury.Exceptions.InterpretingException`:



Public Member Functions

- [InterpretingException](#) ()
Initializes a new instance of the [InterpretingException](#) class.
- [InterpretingException](#) (string message)
Initializes a new instance of the [InterpretingException](#) class with a specified message.
- [InterpretingException](#) (string message, [Exception](#) inner)
Initializes a new instance of the [InterpretingException](#) class with a specified message and an exception that caused this problem.

Protected Member Functions

- [InterpretingException](#) (SerializationInfo info, StreamingContext context)
Initializes a new instance of the [InterpretingException](#) class with serialization data.

6.42.1 Detailed Description

An exception thrown because of problems in the [Interpreting](#) namespace.

Definition at line 11 of file `InterpretingExceptions.cs`.

6.42.2 Constructor & Destructor Documentation

6.42.2.1 `Mercury.Exceptions.InterpretingException.InterpretingException ( )`

Initializes a new instance of the [InterpretingException](#) class.

Definition at line 17 of file `InterpretingExceptions.cs`.

6.42.2.2 Mercury.Exceptions.InterpretingException.InterpretingException (string *message*)

Initializes a new instance of the [InterpretingException](#) class with a specified message.

Parameters

<i>message</i>	The message that describes the error.
----------------	---------------------------------------

Definition at line 25 of file InterpretingExceptions.cs.

6.42.2.3 Mercury.Exceptions.InterpretingException.InterpretingException (string *message*, Exception *inner*)

Initializes a new instance of the [InterpretingException](#) class with a specified message and an exception that caused this problem.

Parameters

<i>message</i>	The message that describes the error.
<i>inner</i>	The exception that caused the problem.

Definition at line 35 of file InterpretingExceptions.cs.

6.42.2.4 Mercury.Exceptions.InterpretingException.InterpretingException (SerializationInfo *info*, StreamingContext *context*)
[protected]

Initializes a new instance of the [InterpretingException](#) class with serialization data.

Parameters

<i>info</i>	The T:System.Runtime.Serialization.SerializationInfo that holds the serialized object data about the exception being thrown.
<i>context</i>	The T:System.Runtime.Serialization.StreamingContext that contains contextual information about the source or destination.

Definition at line 51 of file InterpretingExceptions.cs.

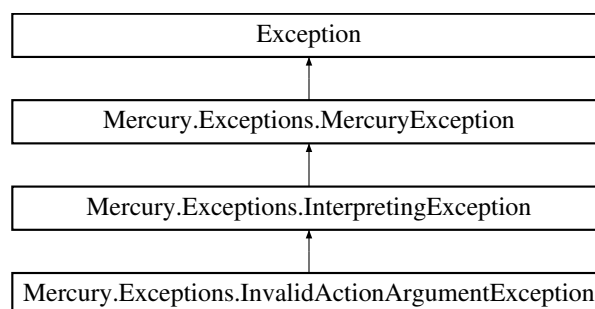
The documentation for this class was generated from the following file:

- [Mercury/Mercury/Exceptions/InterpretingExceptions.cs](#)

6.43 Mercury.Exceptions.InvalidActionArgumentException Class Reference

An exception representing a problem while type checking the Mercury.Interpreting.RewriteRule{T}.

Inheritance diagram for Mercury.Exceptions.InvalidActionArgumentException:



Public Member Functions

- [InvalidActionArgumentException](#) (string rule, string call, int argpos, string reason, params object[] args)
Initializes a new instance of the [InvalidActionArgumentException](#) class.

Additional Inherited Members

6.43.1 Detailed Description

An exception representing a problem while type checking the `Mercury.Interpreting.RewriteRule{T}`.

Definition at line 132 of file `InterpretingExceptions.cs`.

6.43.2 Constructor & Destructor Documentation

- 6.43.2.1 `Mercury.Exceptions.InvalidActionArgumentException.InvalidActionArgumentException ( string rule, string call, int argpos, string reason, params object[] args )`

Initializes a new instance of the [InvalidActionArgumentException](#) class.

Parameters

<i>rule</i>	The rule containing the action.
<i>call</i>	The action call.
<i>argpos</i>	Position of the argument causing the problem.
<i>reason</i>	The reason of the problem.
<i>args</i>	Objects to format.

Definition at line 143 of file `InterpretingExceptions.cs`.

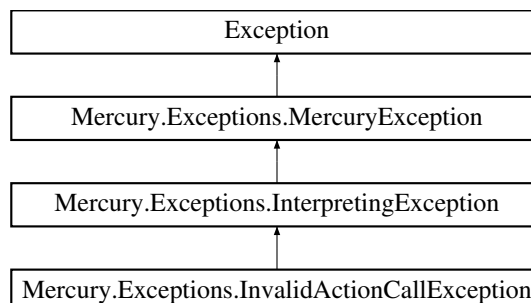
The documentation for this class was generated from the following file:

- `Mercury/Mercury/Exceptions/InterpretingExceptions.cs`

6.44 Mercury.Exceptions.InvalidActionCallException Class Reference

An exception representing a problem with `Mercury.Interpreting.ActionCall{T}`

Inheritance diagram for `Mercury.Exceptions.InvalidActionCallException`:



Public Member Functions

- [InvalidActionCallException](#) (string call, string reason, params object[] args)
Initializes a new instance of the [InvalidActionCallException](#) class.

Additional Inherited Members

6.44.1 Detailed Description

An exception representing a problem with Mercury.Interpreting.ActionCall{T}

Definition at line 64 of file InterpretingExceptions.cs.

6.44.2 Constructor & Destructor Documentation

6.44.2.1 Mercury.Exceptions.InvalidActionCallException.InvalidActionCallException (string *call*, string *reason*, params object[] *args*)

Initializes a new instance of the [InvalidActionCallException](#) class.

Parameters

<i>call</i>	The action call.
<i>reason</i>	The reason of the problem.
<i>args</i>	Objects to format.

Definition at line 73 of file InterpretingExceptions.cs.

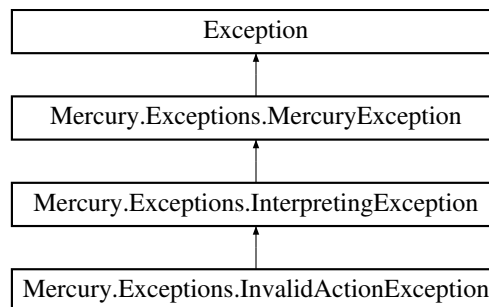
The documentation for this class was generated from the following file:

- Mercury/Mercury/Exceptions/[InterpretingExceptions.cs](#)

6.45 Mercury.Exceptions.InvalidActionException Class Reference

An exception representing a problem with Mercury.Interpreting.InterpreterAction{T} class.

Inheritance diagram for Mercury.Exceptions.InvalidActionException:



Public Member Functions

- [InvalidActionException](#) (string action, string reason, params object[] args)
Initializes a new instance of the [InvalidActionException](#) class.

Additional Inherited Members

6.45.1 Detailed Description

An exception representing a problem with Mercury.Interpreting.InterpreterAction{T} class.

Definition at line 109 of file InterpretingExceptions.cs.

6.45.2 Constructor & Destructor Documentation

6.45.2.1 Mercury.Exceptions.InvalidActionException.InvalidActionException (string *action*, string *reason*, params object[] *args*)

Initializes a new instance of the [InvalidActionException](#) class.

Parameters

<i>action</i>	The action.
<i>reason</i>	The reason of the problem.
<i>args</i>	Object to format.

Definition at line 118 of file InterpretingExceptions.cs.

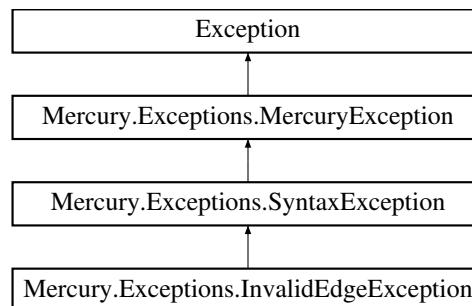
The documentation for this class was generated from the following file:

- Mercury/Mercury/Exceptions/[InterpretingExceptions.cs](#)

6.46 Mercury.Exceptions.InvalidEdgeException Class Reference

An exception thrown when attempting to create an invalid Mercury.Parser.Edge instance.

Inheritance diagram for Mercury.Exceptions.InvalidEdgeException:



Public Member Functions

- [InvalidEdgeException](#) (Edge *edge*)
Initializes a new instance of the [InvalidEdgeException](#) class.

Additional Inherited Members

6.46.1 Detailed Description

An exception thrown when attempting to create an invalid Mercury.Parser.Edge instance.

Definition at line 105 of file SyntaxExceptions.cs.

6.46.2 Constructor & Destructor Documentation

6.46.2.1 Mercury.Exceptions.InvalidEdgeException.InvalidEdgeException (Edge *edge*)

Initializes a new instance of the [InvalidEdgeException](#) class.

Parameters

<i>edge</i>	The edge.
-------------	-----------

Definition at line 112 of file SyntaxExceptions.cs.

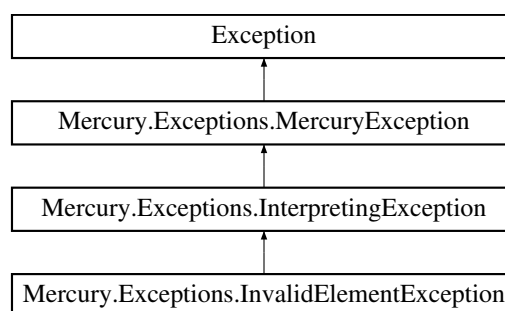
The documentation for this class was generated from the following file:

- Mercury/Mercury/Exceptions/[SyntaxExceptions.cs](#)

6.47 Mercury.Exceptions.InvalidElementException Class Reference

An exception representing a problem with [Mercury.Interpreting.IElement](#) descendants.

Inheritance diagram for Mercury.Exceptions.InvalidElementException:



Public Member Functions

- [InvalidElementException](#) ([IElement](#) element, string reason=null)
Initializes a new instance of the [InvalidElementException](#) class.

Additional Inherited Members

6.47.1 Detailed Description

An exception representing a problem with [Mercury.Interpreting.IElement](#) descendants.

Definition at line 87 of file InterpretingExceptions.cs.

6.47.2 Constructor & Destructor Documentation

6.47.2.1 Mercury.Exceptions.InvalidElementException.InvalidElementException ([IElement](#) *element*, string *reason* = null)

Initializes a new instance of the [InvalidElementException](#) class.

Parameters

<i>element</i>	The element
<i>reason</i>	The reason of the problem.

Definition at line 95 of file InterpretingExceptions.cs.

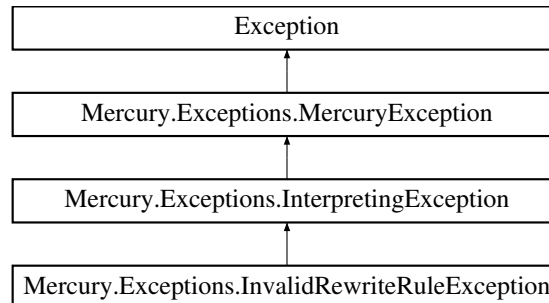
The documentation for this class was generated from the following file:

- Mercury/Mercury/Exceptions/[InterpretingExceptions.cs](#)

6.48 Mercury.Exceptions.InvalidRewriteRuleException Class Reference

An exception representing a problem with `Mercury.Interpreting.RewriteRule{T}`.

Inheritance diagram for `Mercury.Exceptions.InvalidRewriteRuleException`:



Public Member Functions

- [InvalidRewriteRuleException](#) (string rule, string reason, params object[] args)
Initializes a new instance of the [InvalidRewriteRuleException](#) class.

Additional Inherited Members

6.48.1 Detailed Description

An exception representing a problem with `Mercury.Interpreting.RewriteRule{T}`.

Definition at line 158 of file `InterpretingExceptions.cs`.

6.48.2 Constructor & Destructor Documentation

6.48.2.1 `Mercury.Exceptions.InvalidRewriteRuleException.InvalidRewriteRuleException ( string rule, string reason, params object[] args )`

Initializes a new instance of the [InvalidRewriteRuleException](#) class.

Parameters

<i>rule</i>	The rewrite rule
<i>reason</i>	The reason of the problem
<i>args</i>	Objects to format

Definition at line 167 of file `InterpretingExceptions.cs`.

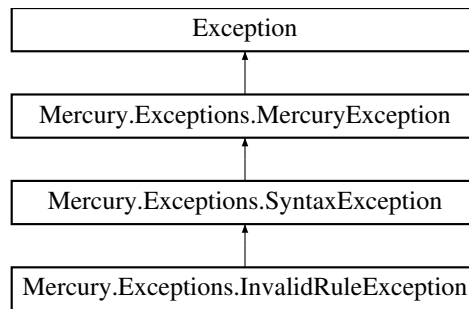
The documentation for this class was generated from the following file:

- `Mercury/Mercury/Exceptions/InterpretingExceptions.cs`

6.49 Mercury.Exceptions.InvalidRuleException Class Reference

An exception thrown when attempting to create an invalid [Mercury.Syntax.Rule](#) instance.

Inheritance diagram for `Mercury.Exceptions.InvalidRuleException`:



Public Member Functions

- [InvalidRuleException](#) ([Rule](#) rule, string reason=null)

Initializes a new instance of the [InvalidRuleException](#) class.

Additional Inherited Members

6.49.1 Detailed Description

An exception thrown when attempting to create an invalid [Mercury.Syntax.Rule](#) instance.

Definition at line 83 of file SyntaxExceptions.cs.

6.49.2 Constructor & Destructor Documentation

6.49.2.1 Mercury.Exceptions.InvalidRuleException.InvalidRuleException ([Rule](#) rule, string reason = null)

Initializes a new instance of the [InvalidRuleException](#) class.

Parameters

<i>rule</i>	The rule
<i>reason</i>	The reason of the problem.

Definition at line 91 of file SyntaxExceptions.cs.

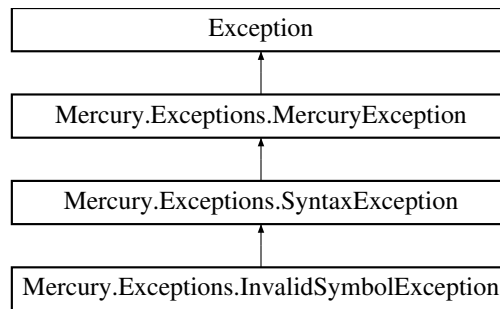
The documentation for this class was generated from the following file:

- Mercury/Mercury/Exceptions/[SyntaxExceptions.cs](#)

6.50 Mercury.Exceptions.InvalidSymbolException Class Reference

An exception thrown when attempting to create an invalid [Mercury.Syntax.Symbol](#) instance.

Inheritance diagram for Mercury.Exceptions.InvalidSymbolException:



Public Member Functions

- [InvalidSymbolException](#) ([Symbol](#) symbol)
Initializes a new instance of the [InvalidSymbolException](#) class.

Additional Inherited Members

6.50.1 Detailed Description

An exception thrown when attempting to create an invalid [Mercury.Syntax.Symbol](#) instance.

Definition at line 63 of file SyntaxExceptions.cs.

6.50.2 Constructor & Destructor Documentation

6.50.2.1 [Mercury.Exceptions.InvalidSymbolException.InvalidSymbolException](#) ([Symbol](#) symbol)

Initializes a new instance of the [InvalidSymbolException](#) class.

Parameters

<i>symbol</i>	The invalid symbol.
---------------	---------------------

Definition at line 70 of file SyntaxExceptions.cs.

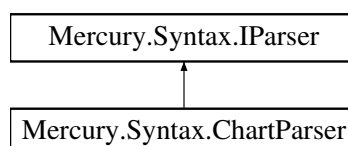
The documentation for this class was generated from the following file:

- [Mercury/Mercury/Exceptions/SyntaxExceptions.cs](#)

6.51 Mercury.Syntax.IParser Interface Reference

Creates a derivation tree from the list of input tokens

Inheritance diagram for [Mercury.Syntax.IParser](#):



Public Member Functions

- [ParserResult](#) [Parse](#) (IList< [Token](#) > tokens)

Creates a trees (in [Mercury.Syntax.ParserResult](#)) from the tokens

- [ParserResult Parse](#) (IList< [Token](#) > tokens, IList< [Rule](#) > scanresult)

Creates a trees (in [Mercury.Syntax.ParserResult](#)) from the tokens with the support of [Mercury.Syntax.IScanner](#) generated rules.

Properties

- [Grammar Grammar](#) [get]

The [Grammar](#) used to build the tree.

6.51.1 Detailed Description

Creates a derivation tree from the list of input tokens

Definition at line 15 of file Parser.cs.

6.51.2 Member Function Documentation

6.51.2.1 [ParserResult Mercury.Syntax.IParser.Parse](#) (IList< [Token](#) > *tokens*)

Creates a trees (in [Mercury.Syntax.ParserResult](#)) from the tokens

Parameters

<i>tokens</i>	The tokens.
---------------	-------------

Returns

Parser result containing trees.

Implemented in [Mercury.Syntax.ChartParser](#).

6.51.2.2 [ParserResult Mercury.Syntax.IParser.Parse](#) (IList< [Token](#) > *tokens*, IList< [Rule](#) > *scanresult*)

Creates a trees (in [Mercury.Syntax.ParserResult](#)) from the tokens with the support of [Mercury.Syntax.IScanner](#) generated rules.

Parameters

<i>tokens</i>	The tokens.
<i>scanresult</i>	The rules from scanner

Returns

Implemented in [Mercury.Syntax.ChartParser](#).

6.51.3 Property Documentation

6.51.3.1 [Grammar Mercury.Syntax.IParser.Grammar](#) [get]

The [Grammar](#) used to build the tree.

Definition at line 18 of file Parser.cs.

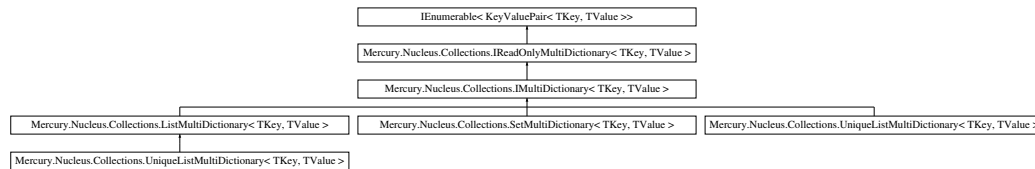
The documentation for this interface was generated from the following file:

- [Mercury/Mercury/Syntax/Parser.cs](#)

6.52 Mercury.Nucleus.Collections.IReadOnlyMultiDictionary< TKey, TValue > Interface Template Reference

Represents a multi-valued read-only dictionary, a map between a key and a collection of values.

Inheritance diagram for Mercury.Nucleus.Collections.IReadOnlyMultiDictionary< TKey, TValue >:



Public Member Functions

- bool [Contains](#) (TKey key, TValue value)
Indicates whether this collection contains the specified key and value.
- bool [ContainsKey](#) (TKey key)
Indicates whether this collection contains the specified key.
- bool [TryGetValue](#) (TKey key, out IReadOnlyList< TValue > collection)
Gets the value associated with the specified key.

Properties

- ICollection< TKey > [Keys](#) [get]
The collection of keys.
- IReadOnlyList< TValue > [this\[TKey key\]](#) [get]
Gets the collection of values associated with the specified key.

6.52.1 Detailed Description

Represents a multi-valued read-only dictionary, a map between a key and a collection of values.

Template Parameters

<i>TKey</i>	The type of the key.
<i>TValue</i>	The type of values.

Definition at line 18 of file MultiDictionary.cs.

6.52.2 Member Function Documentation

6.52.2.1 bool Mercury.Nucleus.Collections.IReadOnlyMultiDictionary< TKey, TValue >.Contains (TKey key, TValue value)

Indicates whether this collection contains the specified key and value.

Parameters

<i>key</i>	The key.
<i>value</i>	The value.

Returns

`true` if this collection contains the given key and value.

Implemented in [Mercury.Nucleus.Collections.UniqueListMultiDictionary< TKey, TValue >](#), [Mercury.Nucleus.Collections.ListMultiDictionary< TKey, TValue >](#), and [Mercury.Nucleus.Collections.SetMultiDictionary< TKey, TValue >](#).

6.52.2.2 `bool Mercury.Nucleus.Collections.IReadOnlyMultiDictionary< TKey, TValue >.ContainsKey ( TKey key )`

Indicates whether this collection contains the specified key.

Parameters

<i>key</i>	The key.
------------	----------

Returns

`true` if the collection contains the key, `false` otherwise.

Exceptions

<i>ArgumentNullException</i>	The key is null.
------------------------------	------------------

Implemented in [Mercury.Nucleus.Collections.ListMultiDictionary< TKey, TValue >](#), and [Mercury.Nucleus.Collections.SetMultiDictionary< TKey, TValue >](#).

6.52.2.3 `bool Mercury.Nucleus.Collections.IReadOnlyMultiDictionary< TKey, TValue >.TryGetValue ( TKey key, out IReadOnlyList< TValue > collection )`

Gets the value associated with the specified key.

Parameters

<i>key</i>	The key.
<i>collection</i>	When this method returns <code>true</code> , this fuek contains the values associated with the given key, otherwise it is set to <code>null</code> .

Returns

`true` if the *key* was found, `false` otherwise

Implemented in [Mercury.Nucleus.Collections.ListMultiDictionary< TKey, TValue >](#), and [Mercury.Nucleus.Collections.SetMultiDictionary< TKey, TValue >](#).

6.52.3 Property Documentation**6.52.3.1** `ICollection<TKey> Mercury.Nucleus.Collections.IReadOnlyMultiDictionary< TKey, TValue >.Keys` [get]

The collection of keys.

Definition at line 45 of file MultiDictionary.cs.

6.52.3.2 `ICollection<TValue> Mercury.Nucleus.Collections.IReadOnlyMultiDictionary< TKey, TValue >.this[TKey key]`
[get]

Gets the collection of values associated with the specified key.

Parameters

<i>key</i>	The key.
------------	----------

Returns

The collection of values associated with the *key* .

Exceptions

<i>KeyNotFoundException</i>	The key was not found
-----------------------------	-----------------------

Definition at line 51 of file MultiDictionary.cs.

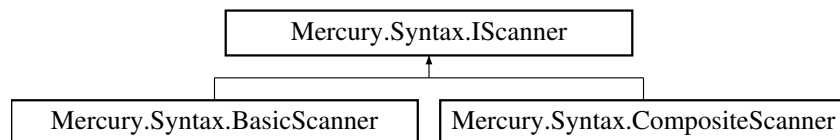
The documentation for this interface was generated from the following file:

- Mercury/Mercury/Nucleus/Collections/[MultiDictionary.cs](#)

6.53 Mercury.Syntax.IScanner Interface Reference

Scanner scans input tokens and creates additional rules for parsing.

Inheritance diagram for Mercury.Syntax.IScanner:



Public Member Functions

- `IEnumerable< Rule > Scan (IEnumerable< Token > tokens)`
Scans input tokens and creates additional rules for parsing.

6.53.1 Detailed Description

Scanner scans input tokens and creates additional rules for parsing.

LHS of these rules should (but is not required to) begin with ' @ ' character to make them different from grammar rules.

Definition at line 20 of file Scanner.cs.

6.53.2 Member Function Documentation

6.53.2.1 `IEnumerable<Rule> Mercury.Syntax.IScanner.Scan ( IEnumerable<Token> tokens )`

Scans input tokens and creates additional rules for parsing.

Parameters

<i>tokens</i>	Input tokens.
---------------	---------------

Returns

Enumeration of additional rules.

Implemented in [Mercury.Syntax.BasicScanner](#), and [Mercury.Syntax.CompositeScanner](#).

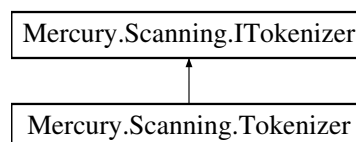
The documentation for this interface was generated from the following file:

- [Mercury/Mercury/Syntax/Scanner.cs](#)

6.54 Mercury.Scanning.ITokenizer Interface Reference

The tokenizer interface.

Inheritance diagram for Mercury.Scanning.ITokenizer:

**Public Member Functions**

- `IList< Token > Tokenize` (string input)
Splits source string into tokens.

6.54.1 Detailed Description

The tokenizer interface.

Definition at line 14 of file `Tokenizer.cs`.

6.54.2 Member Function Documentation

6.54.2.1 `IList<Token> Mercury.Scanning.ITokenizer.Tokenize ( string input )`

Splits source string into tokens.

Parameters

<i>input</i>	Source string.
--------------	----------------

Returns

List of tokens.

Implemented in [Mercury.Scanning.Tokenizer](#).

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

- [Mercury/Mercury/Scanning/Tokenizer.cs](#)

6.55 Mercury.LanguageInfo< T > Class Template Reference

Represents a collection of information required to analyze a language with [Mercury](#) Mercury.Analyser{T,TValue}.

Public Member Functions

- [LanguageInfo](#) (string name, [ITokenizer](#) tokenizer, [IScanner](#) scanner, [Grammar](#) grammar, RewriteRuleCollection< T > rrules)

Initializes a new instance of the LanguageInfo{T} class.

Properties

- [ITokenizer](#) [Tokenizer](#) [get, set]
Gets the tokenizer.
- [IScanner](#) [Scanner](#) [get, set]
Gets the scanner.
- [Grammar](#) [Grammar](#) [get, set]
Gets the grammar.
- RewriteRuleCollection< T > [RewriteRules](#) [get, set]
Gets the rewrite rules.
- string [Name](#) [get, set]
Gets the name of the language.

6.55.1 Detailed Description

Represents a collection of information required to analyze a language with [Mercury](#) Mercury.Analyser{T,TValue}.

Template Parameters

<i>T</i>	Interpretation result type
----------	----------------------------

Type Constraints

***T* : class**

Definition at line 13 of file LanguageInfo.cs.

6.55.2 Constructor & Destructor Documentation

6.55.2.1 Mercury.LanguageInfo< T >.LanguageInfo (string name, ITokenizer tokenizer, IScanner scanner, Grammar grammar, RewriteRuleCollection< T > rrules)

Initializes a new instance of the LanguageInfo{T} class.

Parameters

<i>name</i>	The name of the language.
<i>tokenizer</i>	The tokenizer.
<i>scanner</i>	The scanner.

<i>grammar</i>	The grammar.
<i>rwrules</i>	The rewrite rules.

Exceptions

<i>System.ArgumentException</i>	Name is not valid
---------------------------------	-------------------

Definition at line 27 of file LanguageInfo.cs.

6.55.3 Property Documentation**6.55.3.1 Grammar** `Mercury.LanguageInfo< T >.Grammar` [get], [set]

Gets the grammar.

Definition at line 49 of file LanguageInfo.cs.

6.55.3.2 string `Mercury.LanguageInfo< T >.Name` [get], [set]

Gets the name of the language.

Definition at line 55 of file LanguageInfo.cs.

6.55.3.3 RewriteRuleCollection<T> `Mercury.LanguageInfo< T >.RewriteRules` [get], [set]

Gets the rewrite rules.

Definition at line 52 of file LanguageInfo.cs.

6.55.3.4 IScanner `Mercury.LanguageInfo< T >.Scanner` [get], [set]

Gets the scanner.

Definition at line 46 of file LanguageInfo.cs.

6.55.3.5 ITokenizer `Mercury.LanguageInfo< T >.Tokenizer` [get], [set]

Gets the tokenizer.

Definition at line 43 of file LanguageInfo.cs.

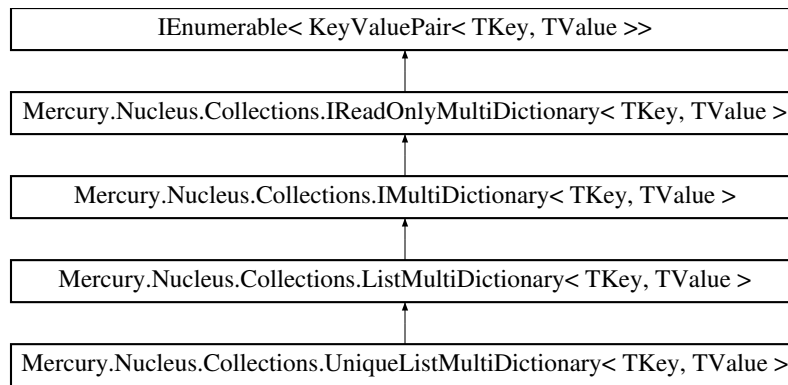
The documentation for this class was generated from the following file:

- Mercury/Mercury/[LanguageInfo.cs](#)

6.56 Mercury.Nucleus.Collections.ListMultiDictionary< TKey, TValue > Class Template Reference

An implementation of `IMultiDictionary{TKey,TValue}` using `T:System.Collections.Generic.List{T}` as an underlying collection of values.

Inheritance diagram for `Mercury.Nucleus.Collections.ListMultiDictionary< TKey, TValue >`:



Public Member Functions

- [ListMultiDictionary](#) ()
Default constructor, creates an empty dictionary
- [ListMultiDictionary](#) (IMultiDictionary< TKey, TValue > source)
Creates a multidictionary initialized by values from another multidictionary.
- TValue [GetValue](#) (TKey key, int ix)
Gets the value from the underlying list.
- int [GetValuesCount](#) (TKey key)
The size of underlying list.
- virtual bool [Add](#) (TKey key, TValue value)
Adds the value to the dictionary. The actual behaviour and handling of duplicities is specified by an implementation.
- virtual bool [Add](#) (KeyValuePair< TKey, TValue > pair)
Adds the value to the dictionary. The actual behaviour and handling of duplicities is specified by an implementation.
- virtual bool [Add](#) (TKey key, IEnumerable< TValue > values)
Adds the collection of values to the dictionary. If the key is already present, NO value is added and false will be returned.
- virtual bool [Contains](#) (TKey key, TValue value)
Indicates whether this collection contains the specified key and value.
- bool [ContainsKey](#) (TKey key)
Indicates whether this collection contains the specified key.
- virtual void [Clear](#) ()
Removes the content of collection.
- bool [TryGetValue](#) (TKey key, out IReadOnlyList< TValue > collection)
Gets the value associated with the specified key.
- IEnumerator< KeyValuePair< TKey, TValue > > [GetEnumerator](#) ()

Properties

- ICollection< TKey > [Keys](#) [get]
- IReadOnlyList< TValue > [this\[TKey key\]](#) [get]

6.56.1 Detailed Description

An implementation of IMultiDictionary{TKey,TValue} using T:System.Collections.Generic.List{T} as an underlying collection of values.

This implementation of IMultiDictionary{TKey,TValue} DOES allow duplicit (key,value) pairs.

Template Parameters

<i>TKey</i>	The type of the key.
<i>TValue</i>	The type of the value.

Definition at line 223 of file MultiDictionary.cs.

6.56.2 Constructor & Destructor Documentation

6.56.2.1 Mercury.Nucleus.Collections.ListMultiDictionary< TKey, TValue >.ListMultiDictionary ()

Default constructor, creates an empty dictionary

Definition at line 234 of file MultiDictionary.cs.

6.56.2.2 Mercury.Nucleus.Collections.ListMultiDictionary< TKey, TValue >.ListMultiDictionary (IMultiDictionary< TKey, TValue > source)

Creates a multidictionary initialized by values from another multidictionary.

Parameters

<i>source</i>	Source multidictionary
---------------	------------------------

Definition at line 242 of file MultiDictionary.cs.

6.56.3 Member Function Documentation

6.56.3.1 virtual bool Mercury.Nucleus.Collections.ListMultiDictionary< TKey, TValue >.Add (TKey key, TValue value) [virtual]

Adds the value to the dictionary. The actual behaviour and handling of duplicities is specified by an implementation.

Parameters

<i>key</i>	The key.
<i>value</i>	The value.

Returns

`true` if value added successfully, `false` otherwise (depends on the implementation).

Exceptions

<i>ArgumentNullException</i>	The key is null.
------------------------------	------------------

Implements [Mercury.Nucleus.Collections.IMultiDictionary< TKey, TValue >](#).

Reimplemented in [Mercury.Nucleus.Collections.UniqueListMultiDictionary< TKey, TValue >](#).

Definition at line 264 of file MultiDictionary.cs.

6.56.3.2 virtual bool Mercury.Nucleus.Collections.ListMultiDictionary< TKey, TValue >.Add (KeyValuePair< TKey, TValue > tuple) [virtual]

Adds the value to the dictionary. The actual behaviour and handling of duplicities is specified by an implementation.

Parameters

<i>tuple</i>	The tuple of key and value.
--------------	-----------------------------

Returns

`true` if value added successfully, `false` otherwise (depends on the implementation).

Exceptions

<i>ArgumentNullException</i>	The tuple is null.
------------------------------	--------------------

Implements [Mercury.Nucleus.Collections.IMultiDictionary< TKey, TValue >](#).

Definition at line 275 of file MultiDictionary.cs.

6.56.3.3 `virtual bool Mercury.Nucleus.Collections.ListMultiDictionary< TKey, TValue >.Add ( TKey key, IEnumerable< TValue > values ) [virtual]`

Adds the collection of values to the dictionary. If the *key* is already present, NO value is added and `false` will be returned.

Parameters

<i>key</i>	The key.
<i>values</i>	The collection of values.

Returns

`true` if *key* is not present yet and at least one value was successfully added; `false` otherwise.

Exceptions

<i>ArgumentNullException</i>	The key is null.
------------------------------	------------------

Implements [Mercury.Nucleus.Collections.IMultiDictionary< TKey, TValue >](#).

Definition at line 278 of file MultiDictionary.cs.

6.56.3.4 `virtual void Mercury.Nucleus.Collections.ListMultiDictionary< TKey, TValue >.Clear ( ) [virtual]`

Removes the content of collection.

Implements [Mercury.Nucleus.Collections.IMultiDictionary< TKey, TValue >](#).

Reimplemented in [Mercury.Nucleus.Collections.UniqueListMultiDictionary< TKey, TValue >](#).

Definition at line 298 of file MultiDictionary.cs.

6.56.3.5 `virtual bool Mercury.Nucleus.Collections.ListMultiDictionary< TKey, TValue >.Contains ( TKey key, TValue value ) [virtual]`

Indicates whether this collection contains the specified key and value.

Parameters

<i>key</i>	The key.
------------	----------

<i>value</i>	The value.
--------------	------------

Returns

`true` if this collection contains the given key and value.

Implements [Mercury.Nucleus.Collections.IReadOnlyMultiDictionary< TKey, TValue >](#).

Reimplemented in [Mercury.Nucleus.Collections.UniqueListMultiDictionary< TKey, TValue >](#).

Definition at line 286 of file MultiDictionary.cs.

6.56.3.6 `bool Mercury.Nucleus.Collections.ListMultiDictionary< TKey, TValue >.ContainsKey ( TKey key )`

Indicates whether this collection contains the specified key.

Parameters

<i>key</i>	The key.
------------	----------

Returns

`true` if the collection contains the key, `false` otherwise.

Exceptions

<i>ArgumentNullException</i>	The key is null.
------------------------------	------------------

Implements [Mercury.Nucleus.Collections.IReadOnlyMultiDictionary< TKey, TValue >](#).

Definition at line 295 of file MultiDictionary.cs.

6.56.3.7 `IEnumerator<KeyValuePair<TKey, TValue> > Mercury.Nucleus.Collections.ListMultiDictionary< TKey, TValue >.GetEnumerator ( )`

Definition at line 320 of file MultiDictionary.cs.

6.56.3.8 `TValue Mercury.Nucleus.Collections.ListMultiDictionary< TKey, TValue >.GetValue ( TKey key, int ix )`

Gets the value from the underlying list.

Parameters

<i>key</i>	The key of the list.
<i>ix</i>	The index of the value.

Returns

The element at position `ix`.

Definition at line 251 of file MultiDictionary.cs.

6.56.3.9 `int Mercury.Nucleus.Collections.ListMultiDictionary< TKey, TValue >.GetValuesCount ( TKey key )`

The size of underlying list.

Parameters

<i>key</i>	The key of the list.
------------	----------------------

Returns

Count of elements of the list.

Definition at line 257 of file MultiDictionary.cs.

6.56.3.10 `bool Mercury.Nucleus.Collections.ListMultiDictionary< TKey, TValue >.TryGetValue ( TKey key, out IReadOnlyList< TValue > collection )`

Gets the value associated with the specified key.

Parameters

<i>key</i>	The key.
<i>collection</i>	When this method returns <code>true</code> , this fuek contains the values associated with the given key, otherwise it is set to <code>null</code> .

Returns

`true` if the *key* was found, `false` otherwise

Implements [Mercury.Nucleus.Collections.IReadOnlyMultiDictionary< TKey, TValue >](#).

Definition at line 301 of file MultiDictionary.cs.

6.56.4 Property Documentation

6.56.4.1 `ICollection<TKey> Mercury.Nucleus.Collections.ListMultiDictionary< TKey, TValue >.Keys` [get]

Definition at line 311 of file MultiDictionary.cs.

6.56.4.2 `IReadOnlyList<TValue> Mercury.Nucleus.Collections.ListMultiDictionary< TKey, TValue >.this[TKey key]` [get]

Definition at line 314 of file MultiDictionary.cs.

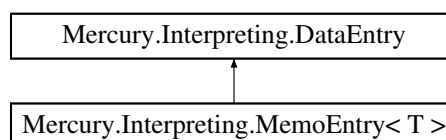
The documentation for this class was generated from the following file:

- [Mercury/Mercury/Nucleus/Collections/MultiDictionary.cs](#)

6.57 Mercury.Interpreting.MemoEntry< T > Class Template Reference

Represents memoized value

Inheritance diagram for Mercury.Interpreting.MemoEntry< T >:



Properties

- [T MemoizedValue](#) [get, set]
Gets the memoized value.

6.57.1 Detailed Description

Represents memoized value

Template Parameters

T	Interpreter return type
-------------------	-------------------------

Definition at line 82 of file Entries.cs.

6.57.2 Property Documentation

6.57.2.1 [T Mercury.Interpreting.MemoEntry< T >.MemoizedValue](#) [get], [set]

Gets the memoized value.

Definition at line 85 of file Entries.cs.

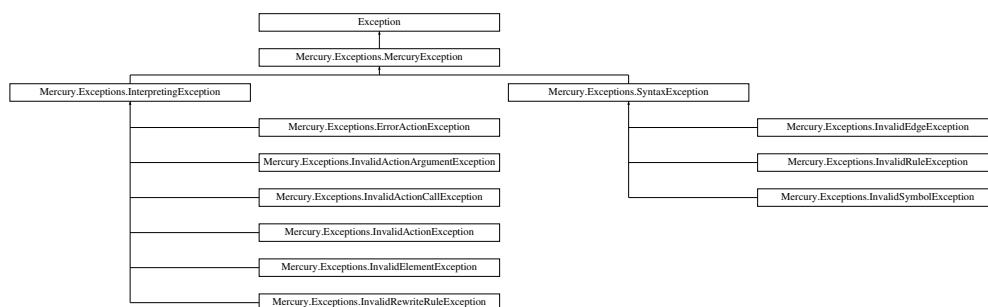
The documentation for this class was generated from the following file:

- [Mercury/Mercury/Interpreting/Entries.cs](#)

6.58 Mercury.Exceptions.MercuryException Class Reference

An exception thrown by the [Mercury](#) library.

Inheritance diagram for Mercury.Exceptions.MercuryException:



Public Member Functions

- [MercuryException](#) ()
Initializes a new instance of the [MercuryException](#) class.
- [MercuryException](#) (string message)
Initializes a new instance of the [MercuryException](#) class with a specified error message.
- [MercuryException](#) (string message, [Exception](#) inner)
Initializes a new instance of the [MercuryException](#) class with a specified error message and an exception that caused the problem.

Protected Member Functions

- [MercuryException](#) (SerializationInfo info, StreamingContext context)
Initializes a new instance of the [MercuryException](#) class with serialized data.

6.58.1 Detailed Description

An exception thrown by the [Mercury](#) library.

Definition at line 10 of file MercuryException.cs.

6.58.2 Constructor & Destructor Documentation

6.58.2.1 Mercury.Exceptions.MercuryException.MercuryException ()

Initializes a new instance of the [MercuryException](#) class.

Definition at line 13 of file MercuryException.cs.

6.58.2.2 Mercury.Exceptions.MercuryException.MercuryException (string message)

Initializes a new instance of the [MercuryException](#) class with a specified error message.

Parameters

<i>message</i>	The message that describes the error.
----------------	---------------------------------------

Definition at line 22 of file MercuryException.cs.

6.58.2.3 Mercury.Exceptions.MercuryException.MercuryException (string message, Exception inner)

Initializes a new instance of the [MercuryException](#) class with a specified error message and an exception that caused the problem.

Parameters

<i>message</i>	The message that describes the error.
<i>inner</i>	The exception that caused the problem.

Definition at line 32 of file MercuryException.cs.

6.58.2.4 Mercury.Exceptions.MercuryException.MercuryException (SerializationInfo info, StreamingContext context) [protected]

Initializes a new instance of the [MercuryException](#) class with serialized data.

Parameters

<i>info</i>	The T:System.Runtime.Serialization.SerializationInfo that holds the serialized object data about the exception being thrown.
<i>context</i>	The T:System.Runtime.Serialization.StreamingContext that contains contextual information about the source or destination.

Definition at line 47 of file MercuryException.cs.

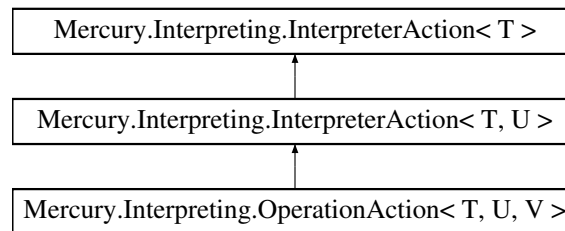
The documentation for this class was generated from the following file:

- Mercury/Mercury/Exceptions/[MercuryException.cs](#)

6.59 Mercury.Interpreting.OperationAction< T, U, V > Class Template Reference

Represents an operation on two values.

Inheritance diagram for Mercury.Interpreting.OperationAction< T, U, V >:



Public Member Functions

- [OperationAction](#) (string name, Func< U, U, V > f, [ArgumentType](#) flags=ArgumentType.None)
Initializes a new instance of the OperationAction{T, U, V} class.
- override V [Evaluate](#) ([Argument](#)[] arguments, RecursiveEval< T > eval, InterpreterContext< T > context)
Evaluates the action.

Additional Inherited Members

6.59.1 Detailed Description

Represents an operation on two values.

Template Parameters

<i>T</i>	Interpreter return type
<i>U</i>	Input value types
<i>V</i>	Output type

Type Constraints

***T* : class**

Definition at line 369 of file BasicActions.cs.

6.59.2 Constructor & Destructor Documentation

6.59.2.1 Mercury.Interpreting.OperationAction< T, U, V >.OperationAction (string name, Func< U, U, V > f, ArgumentType flags = ArgumentType.None)

Initializes a new instance of the OperationAction{T, U, V} class.

Parameters

<i>name</i>	The action name
<i>f</i>	The lambda function
<i>flags</i>	Additional flags

Exceptions

<i>System.ArgumentNull</i> <i>Exception</i>	When <i>f</i> is null
--	-----------------------

Definition at line 383 of file BasicActions.cs.

6.59.3 Member Function Documentation

6.59.3.1 `override V Mercury.Interpreting.OperationAction< T, U, V >.Evaluate ( Argument[] arguments, RecursiveEval< T > eval, InterpreterContext< T > context )` [virtual]

Evaluates the action.

Parameters

<i>arguments</i>	Arguments provided by the interpreter.
<i>eval</i>	The recursive evaluation delegate.
<i>context</i>	The interpreter context.

Returns

Implements [Mercury.Interpreting.InterpreterAction< T, U >](#).

Definition at line 395 of file BasicActions.cs.

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

- Mercury/Mercury/Interpreting/[BasicActions.cs](#)

6.60 Mercury.Syntax.ParserResult Class Reference

Represents the result returned by the parser.

Properties

- bool [Accepted](#) [get, set]
Indicates whether parser recognized the input.
- IReadOnlyList< [Token](#) > [InputTokens](#) [get, set]
Parser input.
- IReadOnlyList< [Edge](#) > [Chart](#) [get, set]
Content of chart after parsing.
- IReadOnlyList< Tree< [Symbol](#) > > [Trees](#) [get, set]
Derivation trees.

6.60.1 Detailed Description

Represents the result returned by the parser.

Definition at line 14 of file ParserResult.cs.

6.60.2 Property Documentation

6.60.2.1 `bool Mercury.Syntax.ParserResult.Accepted` `[get]`, `[set]`

Indicates whether parser recognized the input.

Definition at line 19 of file `ParserResult.cs`.

6.60.2.2 `IReadOnlyList<Edge> Mercury.Syntax.ParserResult.Chart` `[get]`, `[set]`

Content of chart after parsing.

Definition at line 25 of file `ParserResult.cs`.

6.60.2.3 `IReadOnlyList<Token> Mercury.Syntax.ParserResult.InputTokens` `[get]`, `[set]`

Parser input.

Definition at line 22 of file `ParserResult.cs`.

6.60.2.4 `IReadOnlyList<Tree<Symbol>> Mercury.Syntax.ParserResult.Trees` `[get]`, `[set]`

Derivation trees.

Definition at line 28 of file `ParserResult.cs`.

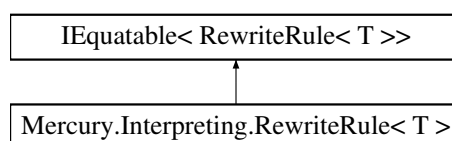
The documentation for this class was generated from the following file:

- `Mercury/Mercury/Syntax/ParserResult.cs`

6.61 `Mercury.Interpreting.RewriteRule< T >` Class Template Reference

Interpreter rewrite rule

Inheritance diagram for `Mercury.Interpreting.RewriteRule< T >`:



Public Member Functions

- `RewriteRule` (`Structure lhs`, `ActionCall< T > rhs`)

Creates a new `RewriteRule`. To be a valid rule, the following conditions must hold:

- override `bool Equals` (`object obj`)
- override `int GetHashCode` ()
- override `string ToString` ()
- `bool Equals` (`RewriteRule< T > other`)

Properties

- [IElement Head](#) [get, set]
Value of LHS's root node.
- [Structure LHS](#) [get, set]
The Left-Hand side of the rule.
- [ActionCall< T > RHS](#) [get, set]
The Right-Hand side of the rule.
- IReadOnlyList< [Variable](#) > [Variables](#) [get, set]

6.61.1 Detailed Description

Interpreter rewrite rule

Type Constraints

***T* : class**

Definition at line 145 of file RewriteRule.cs.

6.61.2 Constructor & Destructor Documentation

6.61.2.1 Mercury.Interpreting.RewriteRule< T >.RewriteRule (Structure lhs, ActionCall< T > rhs)

Creates s new [RewriteRule](#). To be a valid rule, the following conditions must hold:

- LHS is a valid structure
- RHS is a valid action call
- each variable in the LHS must be declared at most once
- each variable in the RHS must be declared in the LHS
- arguments of action calls match required types

Parameters

<i>lhs</i>	The Left-Hand Side Mercury.Interpreting.Structure .
<i>rhs</i>	The Right-Hand Side Mercury.Interpreting.ActionCall{T} .

Exceptions

<i>System.ArgumentNull</i> ↔ <i>Exception</i>	When LHS or RHS is null
--	-------------------------

Definition at line 162 of file RewriteRule.cs.

6.61.3 Member Function Documentation

6.61.3.1 override bool Mercury.Interpreting.RewriteRule< T >.Equals (object obj)

Definition at line 191 of file RewriteRule.cs.

6.61.3.2 **bool** Mercury.Interpreting.RewriteRule< T >.Equals (RewriteRule< T > other)

Definition at line 214 of file RewriteRule.cs.

6.61.3.3 **override int** Mercury.Interpreting.RewriteRule< T >.GetHashCode ()

Definition at line 197 of file RewriteRule.cs.

6.61.3.4 **override string** Mercury.Interpreting.RewriteRule< T >.ToString ()

Definition at line 200 of file RewriteRule.cs.

6.61.4 Property Documentation

6.61.4.1 **IElement** Mercury.Interpreting.RewriteRule< T >.Head [get], [set]

Value of LHS's root node.

Definition at line 177 of file RewriteRule.cs.

6.61.4.2 **Structure** Mercury.Interpreting.RewriteRule< T >.LHS [get], [set]

The Left-Hand side of the rule.

Definition at line 180 of file RewriteRule.cs.

6.61.4.3 **ActionCall<T>** Mercury.Interpreting.RewriteRule< T >.RHS [get], [set]

The Right-Hand side of the rule.

Definition at line 183 of file RewriteRule.cs.

6.61.4.4 **IReadOnlyList<Variable>** Mercury.Interpreting.RewriteRule< T >.Variables [get], [set]

Definition at line 185 of file RewriteRule.cs.

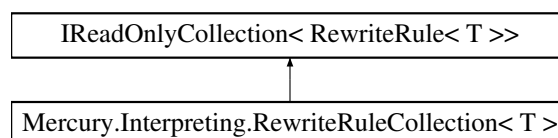
The documentation for this class was generated from the following file:

- Mercury/Mercury/Interpreting/[RewriteRule.cs](#)

6.62 Mercury.Interpreting.RewriteRuleCollection< T > Class Template Reference

A read-only collection of rewrite rules

Inheritance diagram for Mercury.Interpreting.RewriteRuleCollection< T >:



Public Member Functions

- `IEnumerable< RewriteRule< T > > RulesFor (Symbol head)`
Returns a list of rules whose LHS *Structure*'s head matches the given head .
- `IEnumerator< RewriteRule< T > > GetEnumerator ()`

Properties

- `int Count [get]`

6.62.1 Detailed Description

A read-only collection of rewrite rules

Type Constraints

***T* : class**

Definition at line 14 of file RewriteRuleCollection.cs.

6.62.2 Member Function Documentation

6.62.2.1 `IEnumerator< RewriteRule< T > > Mercury.Interpreting.RewriteRuleCollection< T >.GetEnumerator ( )`

Definition at line 60 of file RewriteRuleCollection.cs.

6.62.2.2 `IEnumerable< RewriteRule< T > > Mercury.Interpreting.RewriteRuleCollection< T >.RulesFor ( Symbol head )`

Returns a list of rules whose LHS *Structure*'s head matches the given *head* .

Parameters

<i>head</i>	The head to match.
-------------	--------------------

Returns

List of rules

Definition at line 44 of file RewriteRuleCollection.cs.

6.62.3 Property Documentation

6.62.3.1 `int Mercury.Interpreting.RewriteRuleCollection< T >.Count [get]`

Definition at line 58 of file RewriteRuleCollection.cs.

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

- Mercury/Mercury/Interpreting/[RewriteRuleCollection.cs](#)

6.63 Mercury.Interpreting.RewriteRuleCollectionBuilder< T > Class Template Reference

Can be used to construct Mercury.Interpreting.RewriteRuleCollection{T}

Public Member Functions

- [RewriteRuleCollectionBuilder](#) ()
Initializes a new instance of the RewriteRuleCollectionBuilder{T} class.
- [RewriteRuleCollectionBuilder](#)< T > [AddRewriteRule](#) (RewriteRule< T > rule)
Adds a RewriteRule to the collection
- [RewriteRuleCollectionBuilder](#)< T > [AddRules](#) (RewriteRuleCollection< T > rules)
Adds the collection of rules.
- RewriteRuleCollection< T > [GetCollection](#) ()
Creates a RewriteRuleCollection with rules from this builder.

6.63.1 Detailed Description

Can be used to construct Mercury.Interpreting.RewriteRuleCollection{T}

Type Constraints

T : class

Definition at line 78 of file RewriteRuleCollection.cs.

6.63.2 Constructor & Destructor Documentation

6.63.2.1 Mercury.Interpreting.RewriteRuleCollectionBuilder< T >.RewriteRuleCollectionBuilder ()

Initializes a new instance of the RewriteRuleCollectionBuilder{T} class.

Definition at line 91 of file RewriteRuleCollection.cs.

6.63.3 Member Function Documentation

6.63.3.1 RewriteRuleCollectionBuilder<T> Mercury.Interpreting.RewriteRuleCollectionBuilder< T >.AddRewriteRule (RewriteRule< T > rule)

Adds a RewriteRule to the collection

Parameters

<i>rule</i>	The rule.
-------------	-----------

Returns

This instance to support chaining.

Definition at line 102 of file RewriteRuleCollection.cs.

6.63.3.2 RewriteRuleCollectionBuilder<T> Mercury.Interpreting.RewriteRuleCollectionBuilder< T >.AddRules (RewriteRuleCollection< T > rules)

Adds the collection of rules.

Parameters

<i>rules</i>	The rules.
--------------	------------

Returns

This instance to support chaining.

Definition at line 120 of file RewriteRuleCollection.cs.

6.63.3.3 RewriteRuleCollection<T> Mercury.Interpreting.RewriteRuleCollectionBuilder<T>.GetCollection ()

Creates a RewriteRuleCollection with rules from this builder.

Definition at line 124 of file RewriteRuleCollection.cs.

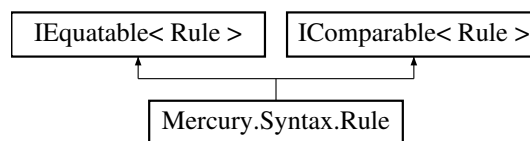
The documentation for this class was generated from the following file:

- Mercury/Mercury/Interpreting/[RewriteRuleCollection.cs](#)

6.64 Mercury.Syntax.Rule Class Reference

Represents a Context-Free [Grammar](#) rule. This is an immutable class.

Inheritance diagram for Mercury.Syntax.Rule:



Public Member Functions

- [Rule](#) ([Symbol](#) lhs, IEnumerable<[Symbol](#)> rhs)
Creates a rule. If the constructor does not throw any exception, the rule is guaranteed to be valid.
- override bool [Equals](#) (object obj)
- override int [GetHashCode](#) ()
- override string [ToString](#) ()
- bool [Equals](#) ([Rule](#) other)
- int [CompareTo](#) ([Rule](#) other)
Compares the rule with another rule. Defines a lexicographical order as follows:

Properties

- [Symbol LHS](#) [get, set]
The Left-Hand Side (LHS) of the rule.
- IReadOnlyList<[Symbol](#)> [RHS](#) [get, set]
The Right-Hand Side (RHS) of the rule.
- bool [IsEpsilon](#) [get, set]
Determines whether the rule is an epsilon rule.

6.64.1 Detailed Description

Represents a Context-Free [Grammar](#) rule. This is an immutable class.

Conditions for a symbol to be valid:

LHS:

a single NONTERMINAL symbol

RHS:

a) a non-empty list of Terminal and/or Nonterminal symbols

– OR –

b) a list of EXACTLY ONE epsilon symbol

Definition at line 23 of file Rule.cs.

6.64.2 Constructor & Destructor Documentation

6.64.2.1 `Mercury.Syntax.Rule.Rule ( Symbol lhs, IEnumerable< Symbol > rhs )`

Creates a rule. If the constructor does not throw any exception, the rule is guaranteed to be valid.

Parameters

<i>lhs</i>	The LHS symbol.
<i>rhs</i>	The RHS symbols.

Exceptions

<i>System.ArgumentNull</i> ↵ <i>Exception</i>	rhs
<i>Mercury.Exceptions</i> ↵ <i>InvalidRuleException</i>	When the rule is not valid.

Definition at line 47 of file Rule.cs.

6.64.3 Member Function Documentation

6.64.3.1 `int Mercury.Syntax.Rule.CompareTo ( Rule other )`

Compares the rule with another rule. Defines a lexicographical order as follows:

1. compare Left-Hand Sides
2. compare the length of Right-Hand Sides
3. lexicographically compare symbols in R-H Sides

Parameters

<i>other</i>	Another rule
--------------	--------------

Returns

A value that indicates the relative order of the rules being compared. See `Comparable`.

Definition at line 151 of file Rule.cs.

6.64.3.2 `override bool Mercury.Syntax.Rule.Equals ( object obj )`

Definition at line 83 of file Rule.cs.

6.64.3.3 bool Mercury.Syntax.Rule.Equals (Rule other)

Definition at line 120 of file Rule.cs.

6.64.3.4 override int Mercury.Syntax.Rule.GetHashCode ()

Definition at line 89 of file Rule.cs.

6.64.3.5 override string Mercury.Syntax.Rule.ToString ()

Definition at line 102 of file Rule.cs.

6.64.4 Property Documentation

6.64.4.1 bool Mercury.Syntax.Rule.IsEpsilon [get], [set]

Determines whether the rule is an epsilon rule.

true if the rule's RHS is epsilon, false otherwise.

Definition at line 77 of file Rule.cs.

6.64.4.2 Symbol Mercury.Syntax.Rule.LHS [get], [set]

The Left-Hand Side (LHS) of the rule.

The LHS [Symbol](#).

Definition at line 66 of file Rule.cs.

6.64.4.3 IReadOnlyList<Symbol> Mercury.Syntax.Rule.RHS [get], [set]

The Right-Hand Side (RHS) of the rule.

An unmodifiable list of RHS symbols.

Definition at line 70 of file Rule.cs.

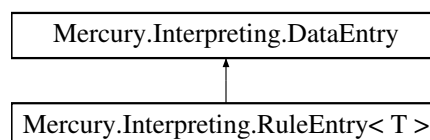
The documentation for this class was generated from the following file:

- Mercury/Mercury/Syntax/[Rule.cs](#)

6.65 Mercury.Interpreting.RuleEntry< T > Class Template Reference

Describes a rule that has been used to interpret a tree.

Inheritance diagram for Mercury.Interpreting.RuleEntry< T >:



Properties

- RewriteRule< T > [RewriteRule](#) [get, set]
Rewrite rule used.
- bool [Match](#) [get, set]
Indicates whether the LHS of rule matched the tree.
- T [Value](#) [get, set]
Returned value or null.
- bool [Error](#) [get, set]
Indicates whether an exception has been thrown when evaluating the rule action.
- [InstanceList](#) [Instances](#) [get, set]
List of instances or null if rule did not match.
- IReadOnlyList< [DataEntry](#) > [SubEntries](#) [get]
List of TextEntries or TreeEntries that occurred while evaluating action of this rule.

6.65.1 Detailed Description

Describes a rule that has been used to interpret a tree.

Template Parameters

<i>T</i>	Interpreter return type
----------	-------------------------

Type Constraints

***T* : class**

Definition at line 92 of file Entries.cs.

6.65.2 Property Documentation

6.65.2.1 bool Mercury.Interpreting.RuleEntry< T >.Error [get], [set]

Indicates whether an exception has been thrown when evaluating the rule action.

Definition at line 124 of file Entries.cs.

6.65.2.2 InstanceList Mercury.Interpreting.RuleEntry< T >.Instances [get], [set]

List of instances or null if rule did not match.

Definition at line 127 of file Entries.cs.

6.65.2.3 bool Mercury.Interpreting.RuleEntry< T >.Match [get], [set]

Indicates whether the LHS of rule matched the tree.

Definition at line 115 of file Entries.cs.

6.65.2.4 RewriteRule< T > Mercury.Interpreting.RuleEntry< T >.RewriteRule [get], [set]

Rewrite rule used.

Definition at line 112 of file Entries.cs.

6.65.2.5 IReadOnlyList<DataEntry> Mercury.Interpreting.RuleEntry< T >.SubEntries [get]

List of TextEntries or TreeEntries that occurred while evaluating action of this rule.

Definition at line 133 of file Entries.cs.

6.65.2.6 T Mercury.Interpreting.RuleEntry< T >.Value [get], [set]

Returned value or null.

Definition at line 118 of file Entries.cs.

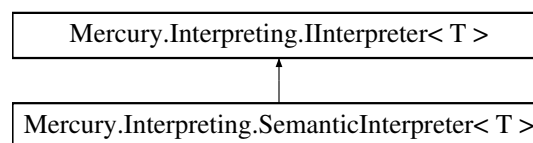
The documentation for this class was generated from the following file:

- Mercury/Mercury/Interpreting/[Entries.cs](#)

6.66 Mercury.Interpreting.SemanticInterpreter< T > Class Template Reference

Represent the semantic analysis with interpretation. Consists of two interpreters, the first one rewrites parse trees, the second one interprets them. This class serves as a wrapper so it could be used in the Mercury.Analyser{T,T $\leftrightarrow$ Result} class.

Inheritance diagram for Mercury.Interpreting.SemanticInterpreter< T >:



Public Member Functions

- [SemanticInterpreter](#) (IInterpreter< [Tree](#)< [Symbol](#) >> semintr, IInterpreter< T > finintr)
Initializes a new instance of the SemanticInterpreter{T} class.
- InterpreterResult< T > [Interpret](#) ([Tree](#)< [Symbol](#) > tree)
Interprets the tree using rewrite rules.
- IReadOnlyList< InterpreterResult< T > > [Interpret](#) (IReadOnlyList< [Tree](#)< [Symbol](#) >> trees)
Interprets trees using parallelism (if underlying interpreters use it).

Properties

- RewriteRuleCollection< [Tree](#)< [Symbol](#) >> [SemanticRules](#) [get]
Gets the semantic rules (those that only rewrite trees).
- RewriteRuleCollection< T > [RewriteRules](#) [get]
Returns the rewrite rules.

6.66.1 Detailed Description

Represent the semantic analysis with interpretation. Consists of two interpreters, the first one rewrites parse trees, the second one interprets them. This class serves as a wrapper so it could be used in the Mercury.Analyser{T,T $\leftrightarrow$ Result} class.

Template Parameters

<i>T</i>	Interpreter output type.
----------	--------------------------

Type Constraints

***T* : class**

Definition at line 431 of file Interpreter.cs.

6.66.2 Constructor & Destructor Documentation

6.66.2.1 Mercury.Interpreting.SemanticInterpreter< T >.SemanticInterpreter (IInterpreter< Tree< Symbol >> *semintr*, IInterpreter< T > *finintr*)

Initializes a new instance of the SemanticInterpreter{T} class.

Parameters

<i>semintr</i>	The semantic interpreter.
<i>finintr</i>	The actual interpreter.

Exceptions

<i>System.ArgumentNull</i> <i>Exception</i>	When either of parameters is <code>null</code> .
--	--

Definition at line 445 of file Interpreter.cs.

6.66.3 Member Function Documentation

6.66.3.1 InterpreterResult<T> Mercury.Interpreting.SemanticInterpreter< T >.Interpret (Tree< Symbol > *tree*)

Interprets the tree using rewrite rules.

Parameters

<i>tree</i>	The tree to interpret.
-------------	------------------------

Returns

Result of the interpretation.

Implements [Mercury.Interpreting.IInterpreter< T >](#).

Definition at line 472 of file Interpreter.cs.

6.66.3.2 IReadOnlyList<InterpreterResult<T>> Mercury.Interpreting.SemanticInterpreter< T >.Interpret (IReadOnlyList< Tree< Symbol >> *trees*)

Interprets trees using parallelism (if underlying interpreters use it).

Parameters

<i>trees</i>	List of trees
--------------	---------------

Returns

Implements [Mercury.Interpreting.IInterpreter< T >](#).

Definition at line 496 of file Interpreter.cs.

6.66.4 Property Documentation

6.66.4.1 RewriteRuleCollection<T> Mercury.Interpreting.SemanticInterpreter< T >.RewriteRules [get]

Returns the rewrite rules.

The rewrite rules.

Definition at line 463 of file Interpreter.cs.

6.66.4.2 RewriteRuleCollection<Tree<Symbol> > Mercury.Interpreting.SemanticInterpreter< T >.SemanticRules [get]

Gets the semantic rules (those that only rewrite trees).

The semantic rules.

Definition at line 459 of file Interpreter.cs.

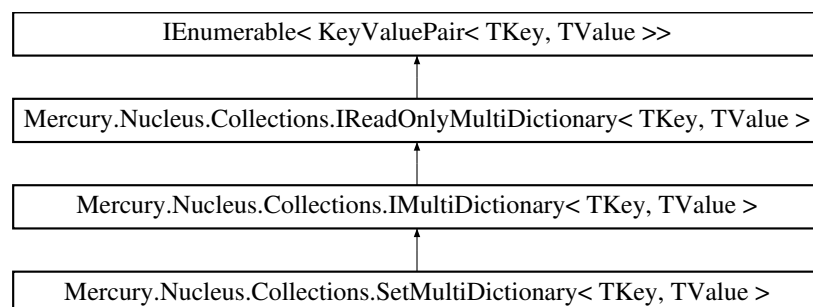
The documentation for this class was generated from the following file:

- Mercury/Mercury/Interpreting/[Interpreter.cs](#)

6.67 Mercury.Nucleus.Collections.SetMultiDictionary< TKey, TValue > Class Template Reference

An implementation of IDictionary{TKey,TValue} using T:System.Collections.Generic.HashSet{T} as an underlying collection of values.

Inheritance diagram for Mercury.Nucleus.Collections.SetMultiDictionary< TKey, TValue >:



Public Member Functions

- bool [Add](#) (TKey key, TValue value)
Adds the value to the dictionary. The actual behaviour and handling of duplicities is specified by an implementation.
- bool [Add](#) (KeyValuePair< TKey, TValue > pair)
Adds the value to the dictionary. The actual behaviour and handling of duplicities is specified by an implementation.

- bool [Add](#) (TKey key, IEnumerable< TValue > values)
Adds the collection of values to the dictionary. If the key is already present, NO value is added and `false` will be returned.
- bool [Contains](#) (TKey key, TValue value)
Indicates whether this collection contains the specified key and value.
- bool [ContainsKey](#) (TKey key)
Indicates whether this collection contains the specified key.
- void [Clear](#) ()
Removes the content of collection.
- bool [TryGetValue](#) (TKey key, out IReadOnlyList< TValue > collection)
Gets the value associated with the specified key.
- IEnumerator< KeyValuePair
< TKey, TValue > > [GetEnumerator](#) ()

Protected Attributes

- Dictionary< TKey, HashSet
< TValue > > [data](#) = new Dictionary<TKey, HashSet<TValue>>()

Properties

- ICollection< TKey > [Keys](#) [get]
- IReadOnlyList< TValue > [this\[TKey key\]](#) [get]

6.67.1 Detailed Description

An implementation of IMultiDictionary{TKey,TValue} using T:System.Collections.Generic.HashSet{T} as an underlying collection of values.

This implementation of IMultiDictionary{TKey,TValue} DOES NOT allow duplicit (key, value) pairs.

Template Parameters

<i>TKey</i>	The type of the key.
<i>TValue</i>	The type of the value.

Definition at line 127 of file MultiDictionary.cs.

6.67.2 Member Function Documentation

6.67.2.1 bool Mercury.Nucleus.Collections.SetMultiDictionary< TKey, TValue >.Add (TKey key, TValue value)

Adds the value to the dictionary. The actual behaviour and handling of duplicities is specified by an implementation.

Parameters

<i>key</i>	The key.
<i>value</i>	The value.

Returns

`true` if value added successfully, `false` otherwise (depends on the implementation).

Exceptions

<i>ArgumentNullException</i>	The key is null.
------------------------------	------------------

Implements [Mercury.Nucleus.Collections.IMultiDictionary< TKey, TValue >](#).

Definition at line 139 of file MultiDictionary.cs.

6.67.2.2 bool Mercury.Nucleus.Collections.SetMultiDictionary< TKey, TValue >.Add (KeyValuePair< TKey, TValue > tuple)

Adds the value to the dictionary. The actual behaviour and handling of duplicities is specified by an implementation.

Parameters

<i>tuple</i>	The tuple of key and value.
--------------	-----------------------------

Returns

`true` if value added successfully, `false` otherwise (depends on the implementation).

Exceptions

<i>ArgumentNullException</i>	The tuple is null.
------------------------------	--------------------

Implements [Mercury.Nucleus.Collections.IMultiDictionary< TKey, TValue >](#).

Definition at line 149 of file MultiDictionary.cs.

6.67.2.3 bool Mercury.Nucleus.Collections.SetMultiDictionary< TKey, TValue >.Add (TKey key, IEnumerable< TValue > values)

Adds the collection of values to the dictionary. If the *key* is already present, NO value is added and `false` will be returned.

Parameters

<i>key</i>	The key.
<i>values</i>	The collection of values.

Returns

`true` if *key* is not present yet and at least one value was successfully added; `false` otherwise.

Exceptions

<i>ArgumentNullException</i>	The key is null.
------------------------------	------------------

Implements [Mercury.Nucleus.Collections.IMultiDictionary< TKey, TValue >](#).

Definition at line 152 of file MultiDictionary.cs.

6.67.2.4 void Mercury.Nucleus.Collections.SetMultiDictionary< TKey, TValue >.Clear ()

Removes the content of collection.

Implements [Mercury.Nucleus.Collections.IMultiDictionary< TKey, TValue >](#).

Definition at line 172 of file MultiDictionary.cs.

6.67.2.5 bool Mercury.Nucleus.Collections.SetMultiDictionary< TKey, TValue >.Contains (TKey key, TValue value)

Indicates whether this collection contains the specified key and value.

Parameters

<i>key</i>	The key.
<i>value</i>	The value.

Returns

`true` if this collection contains the given key and value.

Implements [Mercury.Nucleus.Collections.IReadOnlyMultiDictionary< TKey, TValue >](#).

Definition at line 160 of file MultiDictionary.cs.

6.67.2.6 `bool Mercury.Nucleus.Collections.SetMultiDictionary< TKey, TValue >.ContainsKey ( TKey key )`

Indicates whether this collection contains the specified key.

Parameters

<i>key</i>	The key.
------------	----------

Returns

`true` if the collection contains the key, `false` otherwise.

Exceptions

<i>ArgumentNullException</i>	The key is null.
------------------------------	------------------

Implements [Mercury.Nucleus.Collections.IReadOnlyMultiDictionary< TKey, TValue >](#).

Definition at line 169 of file MultiDictionary.cs.

6.67.2.7 `IEnumerator<KeyValuePair<TKey, TValue> > Mercury.Nucleus.Collections.SetMultiDictionary< TKey, TValue >.GetEnumerator ( )`

Definition at line 194 of file MultiDictionary.cs.

6.67.2.8 `bool Mercury.Nucleus.Collections.SetMultiDictionary< TKey, TValue >.TryGetValue ( TKey key, out IReadOnlyList< TValue > collection )`

Gets the value associated with the specified key.

Parameters

<i>key</i>	The key.
<i>collection</i>	When this method returns <code>true</code> , this fuek contains the values associated with the given key, otherwise it is set to <code>null</code> .

Returns

`true` if the *key* was found, `false` otherwise

Implements [Mercury.Nucleus.Collections.IReadOnlyMultiDictionary< TKey, TValue >](#).

Definition at line 175 of file MultiDictionary.cs.

6.67.3 Member Data Documentation

6.67.3.1 `Dictionary<TKey, HashSet<TValue>> Mercury.Nucleus.Collections.SetMultiDictionary< TKey, TValue >.data = new Dictionary<TKey, HashSet<TValue>>()` [protected]

Definition at line 133 of file MultiDictionary.cs.

6.67.4 Property Documentation

6.67.4.1 `ICollection<TKey> Mercury.Nucleus.Collections.SetMultiDictionary< TKey, TValue >.Keys` [get]

Definition at line 185 of file MultiDictionary.cs.

6.67.4.2 `IReadOnlyList<TValue> Mercury.Nucleus.Collections.SetMultiDictionary< TKey, TValue >.this[TKey key]` [get]

Definition at line 188 of file MultiDictionary.cs.

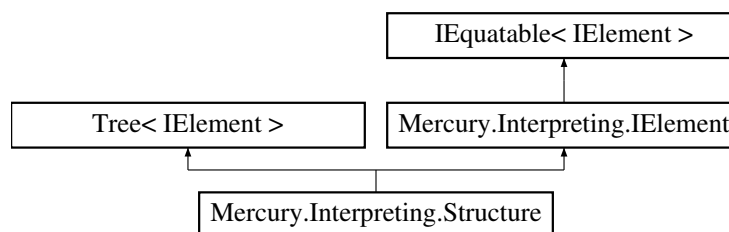
The documentation for this class was generated from the following file:

- [Mercury/Mercury/Nucleus/Collections/MultiDictionary.cs](#)

6.68 Mercury.Interpreting.Structure Class Reference

Represents a tree of [Mercury.Interpreting.IElement](#) instances. Must match precisely to the parse tree.

Inheritance diagram for Mercury.Interpreting.Structure:



Public Member Functions

- [Structure](#) ([IElement](#) head)
Creates a new [Structure](#) structure with root only.
- [Structure](#) ([IElement](#) head, [IEnumerable< IElement >](#) subelements)
Creates a new [Structure](#).
- bool [Equals](#) ([IElement](#) other)

Properties

- bool [HasWildcard](#) [get, set]
Indicates whether this structure contains wildcard amongst its direct child nodes.
- [ElementType](#) [Type](#) [get, set]

6.68.1 Detailed Description

Represents a tree of [Mercury.Interpreting.IElement](#) instances. Must match precisely to the parse tree.

Definition at line 439 of file [Element.cs](#).

6.68.2 Constructor & Destructor Documentation

6.68.2.1 [Mercury.Interpreting.Structure.Structure](#) ([IElement head](#))

Creates a new [Structure](#) structure with root only.

Parameters

<i>head</i>	Structure root. Must inherit from Mercury.Interpreting.Alternative .
-------------	--

Definition at line 452 of file [Element.cs](#).

6.68.2.2 [Mercury.Interpreting.Structure.Structure](#) ([IElement head](#), [IEnumerable< IElement > subelements](#))

Creates a new [Structure](#).

Parameters

<i>head</i>	Structure root. Must inherit from Mercury.Interpreting.Alternative .
<i>subelements</i>	Child elements.

Exceptions

<i>System.ArgumentNullException</i>	When value is null
<i>System.ArgumentException</i>	structure root must be an Alternative

Definition at line 463 of file [Element.cs](#).

6.68.3 Member Function Documentation

6.68.3.1 [bool Mercury.Interpreting.Structure.Equals](#) ([IElement other](#))

Definition at line 496 of file [Element.cs](#).

6.68.4 Property Documentation

6.68.4.1 [bool Mercury.Interpreting.Structure.HasWildcard](#) [[get](#)], [[set](#)]

Indicates whether this structure contains wildcard amongst its direct child nodes.

`true` if the wildcard is present, `false` otherwise.

Definition at line 484 of file [Element.cs](#).

6.68.4.2 [ElementType Mercury.Interpreting.Structure.Type](#) [[get](#)], [[set](#)]

Definition at line 490 of file [Element.cs](#).

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

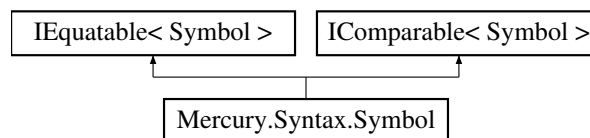
- [Mercury/Mercury/Interpreting/Element.cs](#)

6.69 Mercury.Syntax.Symbol Class Reference

Represents a single symbol of the grammar. This is an immutable class.

For a symbol name to be valid it must NOT contain control or separator characters, must not be empty and must meet these conditions:

Inheritance diagram for Mercury.Syntax.Symbol:



Public Member Functions

- [Symbol](#) ()
Creates an epsilon symbol.
- [Symbol](#) (string name)
Creates a new symbol and infers its type. If the constructor does not throw any exception, the symbol is guaranteed to be valid.
- [Symbol](#) (string name, [SymbolType](#) type)
Creates a new symbol with desired type. If the constructor does not throw any exception, the symbol is guaranteed to be valid.
- [Symbol](#) ([Token](#) token)
Creates a new terminal symbol from a token. If the constructor does not throw any exception, the symbol is guaranteed to be valid.
- override bool [Equals](#) (object obj)
- override int [GetHashCode](#) ()
- override string [ToString](#) ()
- bool [Equals](#) ([Symbol](#) other)
- int [CompareTo](#) ([Symbol](#) other)

Static Public Member Functions

- static [SymbolType InferType](#) (string name)
Infers the type of symbol according to conditions described in the constructor documentation.

Properties

- string [Name](#) [get, set]
Symbol name
- [SymbolType Type](#) [get, set]
Symbol type
- bool [IsEpsilon](#) [get]
Indicates whether the symbol is an epsilon symbol
- [Token Token](#) [get, set]
A token that created this symbol or null if other constructor was used

6.69.1 Detailed Description

Represents a single symbol of the grammar. This is an immutable class.

For a symbol name to be valid it must NOT contain control or separator characters, must not be empty and must meet these conditions:

- epsilon: must meet terminal (a) condition
- terminal: one of the following
 - a) both of the following:
 1. begins with a lowercase letter or a digit
 2. contains only letters and digits
 - b) consists of leading +/-, digits and one dot that is NOT the last character
 - c) exactly one punctuation (printable) symbol
- nonterminal: all of the following:
 - a) begins with an uppercase letter or a character from Mercury.Defaults.Alpha
 - b) contains at least one letter
 - c) contains only letters, digits or characters from Mercury.Defaults.Alpha

If a symbol is terminal, all uppercase characters will be converted to lowercase.

Definition at line 51 of file Symbol.cs.

6.69.2 Constructor & Destructor Documentation

6.69.2.1 Mercury.Syntax.Symbol.Symbol ()

Creates an epsilon symbol.

Definition at line 65 of file Symbol.cs.

6.69.2.2 Mercury.Syntax.Symbol.Symbol (string *name*)

Creates a new symbol and infers its type. If the constructor does not throw any exception, the symbol is guaranteed to be valid.

Parameters

<i>name</i>	Symbol name
-------------	-------------

Definition at line 77 of file Symbol.cs.

6.69.2.3 Mercury.Syntax.Symbol.Symbol (string *name*, SymbolType *type*)

Creates a new symbol with desired type. If the constructor does not throw any exception, the symbol is guaranteed to be valid.

Parameters

<i>name</i>	Symbol name
<i>type</i>	Desired symbol type

Exceptions

System.ArgumentNull ↔ <i>Exception</i>	name
Mercury.Exceptions ↔ InvalidSymbolException	When the symbol is not valid.

Definition at line 91 of file Symbol.cs.

6.69.2.4 Mercury.Syntax.Symbol.Symbol (**Token** *token*)

Creates a new terminal symbol from a token. If the constructor does not throw any exception, the symbol is guaranteed to be valid.

Parameters

<i>token</i>	Input token
--------------	-------------

Exceptions

System.ArgumentNull ↔ <i>Exception</i>	token
Mercury.Exceptions ↔ InvalidSymbolException	When the symbol is not valid.

Definition at line 110 of file Symbol.cs.

6.69.3 Member Function Documentation

6.69.3.1 int Mercury.Syntax.Symbol.CompareTo (**Symbol** *other*)

Definition at line 235 of file Symbol.cs.

6.69.3.2 override bool Mercury.Syntax.Symbol.Equals (**object** *obj*)

Definition at line 203 of file Symbol.cs.

6.69.3.3 bool Mercury.Syntax.Symbol.Equals (**Symbol** *other*)

Definition at line 228 of file Symbol.cs.

6.69.3.4 override int Mercury.Syntax.Symbol.GetHashCode ()

Definition at line 209 of file Symbol.cs.

6.69.3.5 static **SymbolType** Mercury.Syntax.Symbol.InferType (**string** *name*) [static]

Infers the type of symbol according to conditions described in the constructor documentation.

If input meets terminal a) condition (= terminal or epsilon), this method returns `SymbolType.Terminal` | `SymbolType.Epsilon`.

Parameters

<i>name</i>	Symbol name
-------------	-----------------------------

Returns

Inferred symbol type

Definition at line 133 of file Symbol.cs.

6.69.3.6 `override string Mercury.Syntax.Symbol.ToString ( )`

Definition at line 212 of file Symbol.cs.

6.69.4 Property Documentation

6.69.4.1 `bool Mercury.Syntax.Symbol.IsEpsilon` [get]

Indicates whether the symbol is an epsilon symbol

`true` if this is epsilon; `false` otherwise

Definition at line 191 of file Symbol.cs.

6.69.4.2 `string Mercury.Syntax.Symbol.Name` [get], [set]

[Symbol](#) name

[Symbol](#) name

Definition at line 183 of file Symbol.cs.

6.69.4.3 `Token Mercury.Syntax.Symbol.Token` [get], [set]

A token that created this symbol or null if other constructor was used

Definition at line 197 of file Symbol.cs.

6.69.4.4 `SymbolType Mercury.Syntax.Symbol.Type` [get], [set]

[Symbol](#) type

The type of the symbol, inferred when constructed

Definition at line 187 of file Symbol.cs.

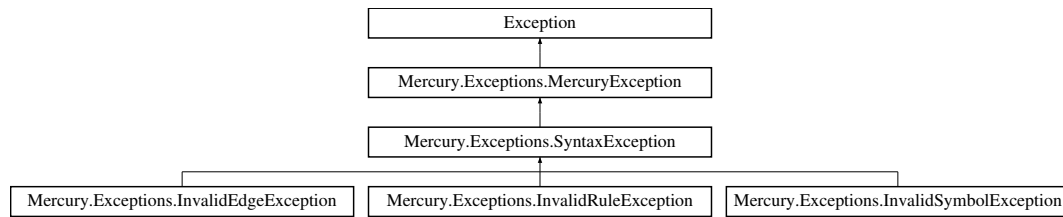
The documentation for this class was generated from the following file:

- Mercury/Mercury/Syntax/[Symbol.cs](#)

6.70 Mercury.Exceptions.SyntaxException Class Reference

An exception thrown when something wrong happens in the [Mercury.Syntax](#)

Inheritance diagram for Mercury.Exceptions.SyntaxException:



Public Member Functions

- [SyntaxException](#) ()
Initializes a new instance of the [SyntaxException](#) class.
- [SyntaxException](#) (string message)
Initializes a new instance of the [SyntaxException](#) class with a message that describes the error.
- [SyntaxException](#) (string message, [Exception](#) inner)
Initializes a new instance of the [SyntaxException](#) class.

Protected Member Functions

- [SyntaxException](#) (SerializationInfo info, StreamingContext context)
Initializes a new instance of the [SyntaxException](#) class.

6.70.1 Detailed Description

An exception thrown when something wrong happens in the [Mercury.Syntax](#) Definition at line 14 of file `SyntaxExceptions.cs`.

6.70.2 Constructor & Destructor Documentation

6.70.2.1 Mercury.Exceptions.SyntaxException.SyntaxException ()

Initializes a new instance of the [SyntaxException](#) class.

Definition at line 20 of file `SyntaxExceptions.cs`.

6.70.2.2 Mercury.Exceptions.SyntaxException.SyntaxException (string message)

Initializes a new instance of the [SyntaxException](#) class with a message that describes the error.

Parameters

<i>message</i>	The message that describes the error.
----------------	---------------------------------------

Definition at line 28 of file `SyntaxExceptions.cs`.

6.70.2.3 Mercury.Exceptions.SyntaxException.SyntaxException (string message, Exception inner)

Initializes a new instance of the [SyntaxException](#) class.

Parameters

<i>message</i>	The message that describes the error.
<i>inner</i>	The exception that caused the problem.

Definition at line 37 of file SyntaxExceptions.cs.

6.70.2.4 Mercury.Exceptions.SyntaxException.SyntaxException (*SerializationInfo info*, *StreamingContext context*) [protected]

Initializes a new instance of the [SyntaxException](#) class.

Parameters

<i>info</i>	The T:System.Runtime.Serialization.SerializationInfo that holds the serialized object data about the exception being thrown.
<i>context</i>	The T:System.Runtime.Serialization.StreamingContext that contains contextual information about the source or destination.

Definition at line 51 of file SyntaxExceptions.cs.

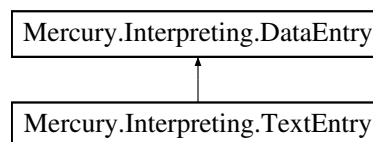
The documentation for this class was generated from the following file:

- Mercury/Mercury/Exceptions/[SyntaxExceptions.cs](#)

6.71 Mercury.Interpreting.TextEntry Class Reference

Stores informations logged by actions or internal Evaluator.

Inheritance diagram for Mercury.Interpreting.TextEntry:

**Properties**

- string [ActionName](#) [get, set]
The name of the action.
- string [Text](#) [get]
Entry text.

6.71.1 Detailed Description

Stores informations logged by actions or internal Evaluator.

Definition at line 25 of file Entries.cs.

6.71.2 Property Documentation

6.71.2.1 string Mercury.Interpreting.TextEntry.ActionName [get], [set]

The name of the action.

Definition at line 43 of file Entries.cs.

6.71.2.2 string Mercury.Interpreting.TextEntry.Text [get]

Entry text.

Definition at line 46 of file Entries.cs.

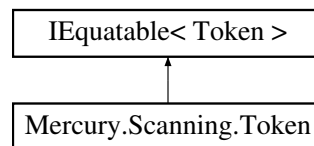
The documentation for this class was generated from the following file:

- Mercury/Mercury/Interpreting/[Entries.cs](#)

6.72 Mercury.Scanning.Token Class Reference

[Token](#), a part of an input text recognized by tokenizer

Inheritance diagram for Mercury.Scanning.Token:



Public Member Functions

- [Token](#) (string value, [TokenType](#) type, int position, bool quoted=false)
Creates a new [Token](#).
- override bool [Equals](#) (object obj)
- override int [GetHashCode](#) ()
- override string [ToString](#) ()
- bool [Equals](#) ([Token](#) other)

Properties

- string [Value](#) [get, set]
The token value.
- [TokenType](#) [Type](#) [get, set]
[Token](#) type.
- int [Position](#) [get, set]
Position in the input text.
- bool [Quoted](#) [get, set]
Indicates whether the token was quoted in the input.

6.72.1 Detailed Description

[Token](#), a part of an input text recognized by tokenizer

Definition at line 29 of file Token.cs.

6.72.2 Constructor & Destructor Documentation

6.72.2.1 Mercury.Scanning.Token.Token (string *value*, TokenType *type*, int *position*, bool *quoted* = false)

Creates a new [Token](#).

Parameters

<i>value</i>	Token value.
<i>type</i>	Token type.
<i>position</i>	Position in the input text.
<i>quoted</i>	Indicates whether the token value was quoted in the input.

Definition at line 38 of file Token.cs.

6.72.3 Member Function Documentation

6.72.3.1 override bool Mercury.Scanning.Token.Equals (object *obj*)

Definition at line 66 of file Token.cs.

6.72.3.2 bool Mercury.Scanning.Token.Equals ([Token](#) *other*)

Definition at line 84 of file Token.cs.

6.72.3.3 override int Mercury.Scanning.Token.GetHashCode ()

Definition at line 72 of file Token.cs.

6.72.3.4 override string Mercury.Scanning.Token.ToString ()

Definition at line 75 of file Token.cs.

6.72.4 Property Documentation

6.72.4.1 int Mercury.Scanning.Token.Position [get], [set]

Position in the input text.

Definition at line 57 of file Token.cs.

6.72.4.2 bool Mercury.Scanning.Token.Quoted [get], [set]

Indicates whether the token was quoted in the input.

Definition at line 60 of file Token.cs.

6.72.4.3 [TokenType](#) Mercury.Scanning.Token.Type [get], [set]

[Token](#) type.

Definition at line 54 of file Token.cs.

6.72.4.4 string Mercury.Scanning.Token.Value [get], [set]

The token value.

Definition at line 51 of file Token.cs.

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

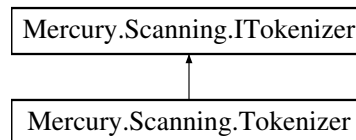
- [Mercury/Mercury/Scanning/Token.cs](#)

6.73 Mercury.Scanning.Tokenizer Class Reference

A default implementation of [ITokenizer](#) interface using Microsoft Regular Expressions.

Splits tokens in the following fashion:

Inheritance diagram for Mercury.Scanning.Tokenizer:



Public Member Functions

- [Tokenizer](#) ()
Creates a default tokenizer
- [Tokenizer](#) (IEnumerable< char > alpha)
Creates a tokenizer with custom alpha characters
- [Tokenizer](#) (IEnumerable< char > alpha, IEnumerable< string > keywords, [TokenizerOptions](#) options)
Creates a tokenizer with custom alphanumeric characters, keywords and options.
- IList< [Token](#) > [Tokenize](#) (string input)
Splits source string into tokens.

Properties

- string[] [Keywords](#) [get, set]
Tokenizer keywords.
- char[] [Alpha](#) [get, set]
Alpha characters.
- [TokenizerOptions](#) [Options](#) [get, set]
Tokenizer options.

6.73.1 Detailed Description

A default implementation of [ITokenizer](#) interface using Microsoft Regular Expressions.

Splits tokens in the following fashion:

- Control characters are not permitted
- Whitespaces are considered separators unless in quotations
- Quoted sequence is always a String token regardless its content

- Escape sequences `""`, `\` and `'` will lose their control function and will be interpreted as characters `""`, `\"` and `'` respectively. Other escapes will be ignored and left unchanged
- Number
 - a) must begin with `+`, `-` or a digit
 - b) can contain one `.` between two digits
 - c) must end with a digit
 - d) must contain only digits, `+`, `-` and `.`
 - e) if StrictNumbers option is used, a number must be separated by whitespaces
- Keyword
 - a) must be specified in the keyword list
 - b) must be separated from other tokens by whitespaces
 - c) must NOT be quoted, otherwise will be marked as string
- Symbol
 - a) a single non-alphanumeric character that is not a whitespace
 - b) must NOT be specified in `alpha` parameter, otherwise will be marked as string
- String
 - a) anything else

Definition at line 71 of file Tokenizer.cs.

6.73.2 Constructor & Destructor Documentation

6.73.2.1 Mercury.Scanning.Tokenizer.Tokenizer ()

Creates a default tokenizer

Definition at line 141 of file Tokenizer.cs.

6.73.2.2 Mercury.Scanning.Tokenizer.Tokenizer (IEnumerable< char > *alpha*)

Creates a tokenizer with custom alpha characters

Parameters

<i>alpha</i>	Custom alphanumeric characters.
--------------	---------------------------------

Definition at line 147 of file Tokenizer.cs.

6.73.2.3 Mercury.Scanning.Tokenizer.Tokenizer (IEnumerable< char > *alpha*, IEnumerable< string > *keywords*, TokenizerOptions *options*)

Creates a tokenizer with custom alphanumeric characters, keywords and options.

Parameters

<i>alpha</i>	Custom alphanumeric characters.
--------------	---------------------------------

<i>keywords</i>	Keywords.
<i>options</i>	Tokenizer options.

Definition at line 158 of file Tokenizer.cs.

6.73.3 Member Function Documentation

6.73.3.1 IList<Token> Mercury.Scanning.Tokenizer.Tokenize (string *input*)

Splits source string into tokens.

Parameters

<i>input</i>	Source string.
--------------	----------------

Returns

List of tokens.

Implements [Mercury.Scanning.ITokenizer](#).

Definition at line 196 of file Tokenizer.cs.

6.73.4 Property Documentation

6.73.4.1 char [] Mercury.Scanning.Tokenizer.Alpha [get], [set]

Alpha characters.

Definition at line 187 of file Tokenizer.cs.

6.73.4.2 string [] Mercury.Scanning.Tokenizer.Keywords [get], [set]

[Tokenizer](#) keywords.

Definition at line 184 of file Tokenizer.cs.

6.73.4.3 TokenizerOptions Mercury.Scanning.Tokenizer.Options [get], [set]

[Tokenizer](#) options.

Definition at line 190 of file Tokenizer.cs.

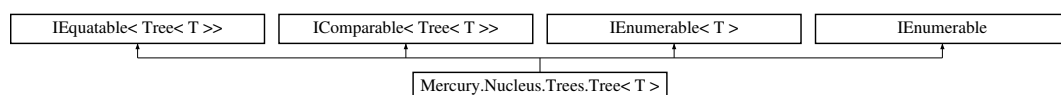
The documentation for this class was generated from the following file:

- Mercury/Mercury/Scanning/[Tokenizer.cs](#)

6.74 Mercury.Nucleus.Trees.Tree< T > Class Template Reference

A non-balanced tree implementation that can be referenced from many other trees. Because of that, there is no way to get back to the parent node.

Inheritance diagram for Mercury.Nucleus.Trees.Tree< T >:



Public Member Functions

- [Tree](#) (T value)
Initializes a new instance of the Tree{T} class with no child nodes (Leaf)
- [Tree](#) (Tree< T > other, IEnumerable< Tree< T >> children)
"Copy" constructor
- [Tree](#) (T value, IEnumerable< Tree< T >> children)
Initializes a new instance of the Tree{T} class with the given parent and children
- override bool [Equals](#) (object obj)
- override int [GetHashCode](#) ()
- string [ToString](#) (int maxdepth)
- override string [ToString](#) ()
- bool [Equals](#) (Tree< T > other)
- int [CompareTo](#) (Tree< T > other)
- IEnumerator< T > [GetEnumerator](#) ()

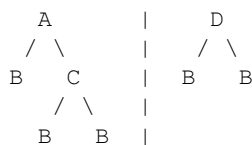
Properties

- int [Depth](#) [get, set]
Maximum depth of the tree
- int [LeavesCount](#) [get, set]
Number of leaves in the tree
- int [NodesCount](#) [get, set]
Number of nodes in the tree
- T [Value](#) [get, set]
The value of this node
- bool [IsLeaf](#) [get]
Checks whether this node has no children
- IReadOnlyList< Tree< T >> [Children](#) [get, set]
Tree's children
- DFSTreeEnumerator< T > [DepthFirstEnumerator](#) [get]
Depth-First enumerator (stack based)
- BFSTreeEnumerator< T > [BreadthFirstEnumerator](#) [get]
Breadth-First Enumerator (queue based)
- Tree< T > [this\[int ix\]](#) [get]
Indexer returns the child

6.74.1 Detailed Description

A non-balanced tree implementation that can be referenced from many other trees. Because of that, there is no way to get back to the parent node.

Example:



All "B" subtrees here may be the same instance of the Tree{T} class.

Template Parameters

<i>T</i>	The type of node values
----------	-------------------------

Definition at line 30 of file Tree.cs.

6.74.2 Constructor & Destructor Documentation

6.74.2.1 Mercury.Nucleus.Trees.Tree< T >.Tree (T *value*)

Initializes a new instance of the Tree{T} class with no child nodes (Leaf)

Parameters

<i>value</i>	The value of the node
--------------	-----------------------

Definition at line 49 of file Tree.cs.

6.74.2.2 Mercury.Nucleus.Trees.Tree< T >.Tree (Tree< T > *other*, IEnumerable< Tree< T >> *children*)

"Copy" constructor

Parameters

<i>other</i>	Source tree
<i>children</i>	Additional children

Definition at line 56 of file Tree.cs.

6.74.2.3 Mercury.Nucleus.Trees.Tree< T >.Tree (T *value*, IEnumerable< Tree< T >> *children*)

Initializes a new instance of the Tree{T} class with the given parent and children

Parameters

<i>value</i>	The value of the node
<i>children</i>	The children to be added

Definition at line 66 of file Tree.cs.

6.74.3 Member Function Documentation

6.74.3.1 int Mercury.Nucleus.Trees.Tree< T >.CompareTo (Tree< T > *other*)

Definition at line 197 of file Tree.cs.

6.74.3.2 override bool Mercury.Nucleus.Trees.Tree< T >.Equals (object *obj*)

Definition at line 125 of file Tree.cs.

6.74.3.3 bool Mercury.Nucleus.Trees.Tree< T >.Equals (Tree< T > *other*)

Definition at line 178 of file Tree.cs.

6.74.3.4 IEnumerator<T> Mercury.Nucleus.Trees.Tree< T >.GetEnumerator ()

Definition at line 213 of file Tree.cs.

6.74.3.5 `override int Mercury.Nucleus.Trees.Tree< T >.GetHashCode ( )`

Definition at line 131 of file Tree.cs.

6.74.3.6 `string Mercury.Nucleus.Trees.Tree< T >.ToString ( int maxdepth )`

Definition at line 148 of file Tree.cs.

6.74.3.7 `override string Mercury.Nucleus.Trees.Tree< T >.ToString ( )`

Definition at line 169 of file Tree.cs.

6.74.4 Property Documentation

6.74.4.1 `BFSTreeEnumerator<T> Mercury.Nucleus.Trees.Tree< T >.BreadthFirstEnumerator` [get]

Breadth-First Enumerator (queue based)

Definition at line 112 of file Tree.cs.

6.74.4.2 `IReadOnlyList<Tree<T>> Mercury.Nucleus.Trees.Tree< T >.Children` [get], [set]

Tree's children

Gets the children list as read-only

Definition at line 104 of file Tree.cs.

6.74.4.3 `int Mercury.Nucleus.Trees.Tree< T >.Depth` [get], [set]

Maximum depth of the tree

Definition at line 88 of file Tree.cs.

6.74.4.4 `DFSTreeEnumerator<T> Mercury.Nucleus.Trees.Tree< T >.DepthFirstEnumerator` [get]

Depth-First enumerator (stack based)

Definition at line 108 of file Tree.cs.

6.74.4.5 `bool Mercury.Nucleus.Trees.Tree< T >.IsLeaf` [get]

Checks whether this node has no children

Definition at line 100 of file Tree.cs.

6.74.4.6 `int Mercury.Nucleus.Trees.Tree< T >.LeavesCount` [get], [set]

Number of leaves in the tree

Definition at line 91 of file Tree.cs.

6.74.4.7 `int Mercury.Nucleus.Trees.Tree< T >.NodesCount` `[get]`, `[set]`

Number of nodes in the tree

Definition at line 94 of file Tree.cs.

6.74.4.8 `Tree<T> Mercury.Nucleus.Trees.Tree< T >.this[int ix]` `[get]`

Indexer returns the child

Parameters

<code>ix</code>	
-----------------	--

Returns

Definition at line 119 of file Tree.cs.

6.74.4.9 `T Mercury.Nucleus.Trees.Tree< T >.Value` `[get]`, `[set]`

The value of this node

Definition at line 97 of file Tree.cs.

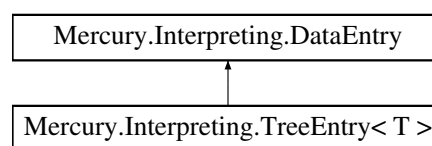
The documentation for this class was generated from the following file:

- Mercury/Mercury/Nucleus/Trees/[Tree.cs](#)

6.75 Mercury.Interpreting.TreeEntry< T > Class Template Reference

Represents an entry for tree interpretation.

Inheritance diagram for Mercury.Interpreting.TreeEntry< T >:



Properties

- `Tree< Symbol > Tree` `[get]`, `[set]`
The tree that has been attempted to be interpreted.
- `bool Succeeded` `[get]`, `[set]`
Indicates whether the interpretation was successful.
- `ReadOnlyList< DataEntry > RuleEntries` `[get]`
List of rule entries that were tried.

6.75.1 Detailed Description

Represents an entry for tree interpretation.

Type Constraints

***T* : class**

Definition at line 52 of file Entries.cs.

6.75.2 Property Documentation

6.75.2.1 IReadOnlyList<DataEntry> Mercury.Interpreting.TreeEntry< T >.RuleEntries [get]

List of rule entries that were tried.

Definition at line 75 of file Entries.cs.

6.75.2.2 bool Mercury.Interpreting.TreeEntry< T >.Succeeded [get], [set]

Indicates whether the interpretation was successful.

Definition at line 72 of file Entries.cs.

6.75.2.3 Tree<Symbol> Mercury.Interpreting.TreeEntry< T >.Tree [get], [set]

The tree that has been attempted to be interpreted.

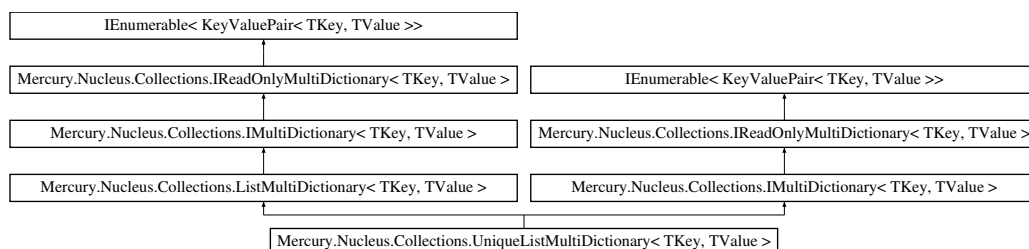
Definition at line 69 of file Entries.cs.

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

- Mercury/Mercury/Interpreting/[Entries.cs](#)

6.76 Mercury.Nucleus.Collections.UniqueListMultiDictionary< TKey, TValue > Class Template Reference

An implementation of `IMultiDictionary{TKey,TValue}` using `T:System.Collections.Generic.List{T}` as underlying collection, but opposed to `ListMultiDictionary{TKey,TValue}` does not allow duplicate values associated with the same key

Inheritance diagram for `Mercury.Nucleus.Collections.UniqueListMultiDictionary< TKey, TValue >`:

Public Member Functions

- override bool [Add](#) (TKey key, TValue value)
Adds the value to the dictionary. The actual behaviour and handling of duplicities is specified by an implementation.
- override bool [Contains](#) (TKey key, TValue value)
Indicates whether this collection contains the specified key and value.

- override void [Clear](#) ()
Removes the content of collection.

Protected Attributes

- SetMultiDictionary< TKey, TValue > [data](#) = new SetMultiDictionary<TKey, TValue>()

Additional Inherited Members

6.76.1 Detailed Description

An implementation of IMultiDictionary{TKey,TValue} using T:System.Collections.Generic.List{T} as underlying collection, but opposed to ListMultiDictionary{TKey,TValue} does not allow duplicate values associated with the same key

This implementation of IMultiDictionary{TKey,TValue} DOES NOT allow duplicit (key,value) pairs.

Template Parameters

<i>TKey</i>	The type of the key.
<i>TValue</i>	The type of the value.

Definition at line 351 of file MultiDictionary.cs.

6.76.2 Member Function Documentation

6.76.2.1 override bool Mercury.Nucleus.Collections.UniqueListMultiDictionary< TKey, TValue >.Add (TKey key, TValue value)

Adds the value to the dictionary. The actual behaviour and handling of duplicities is specified by an implementation.

Parameters

<i>key</i>	The key.
<i>value</i>	The value.

Returns

`true` if value added successfully, `false` otherwise (depends on the implementation).

Exceptions

<i>ArgumentNullException</i>	The key is null.
------------------------------	------------------

Implements [Mercury.Nucleus.Collections.IMultiDictionary< TKey, TValue >](#).

Definition at line 363 of file MultiDictionary.cs.

6.76.2.2 override void Mercury.Nucleus.Collections.UniqueListMultiDictionary< TKey, TValue >.Clear ()

Removes the content of collection.

Implements [Mercury.Nucleus.Collections.IMultiDictionary< TKey, TValue >](#).

Definition at line 369 of file MultiDictionary.cs.

6.76.2.3 override bool Mercury.Nucleus.Collections.UniqueListMultiDictionary< TKey, TValue >.Contains (TKey key, TValue value) [virtual]

Indicates whether this collection contains the specified key and value.

Parameters

<i>key</i>	The key.
<i>value</i>	The value.

Returns

`true` if this collection contains the given key and value.

Reimplemented from [Mercury.Nucleus.Collections.ListMultiDictionary< TKey, TValue >](#).

Definition at line 366 of file MultiDictionary.cs.

6.76.3 Member Data Documentation

6.76.3.1 `SetMultiDictionary<TKey, TValue> Mercury.Nucleus.Collections.UniqueListMultiDictionary< TKey, TValue >.data = new SetMultiDictionary<TKey, TValue>() [protected]`

Definition at line 357 of file MultiDictionary.cs.

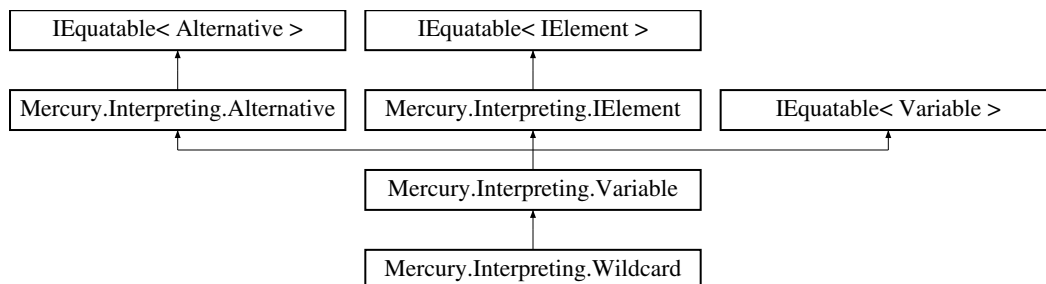
The documentation for this class was generated from the following file:

- [Mercury/Mercury/Nucleus/Collections/MultiDictionary.cs](#)

6.77 Mercury.Interpreting.Variable Class Reference

Represents a variable element that captures value and can be used on the RHS of the RewriteRule as an argument

Inheritance diagram for Mercury.Interpreting.Variable:



Public Member Functions

- [Variable \(\)](#)
Creates an anonymous variable, an equivalent of [Mercury.Interpreting.Wildcard](#) that matches exactly one symbol
- [Variable \(string name, SymbolType ws, IEnumerable< Symbol > symbols\)](#)
Initializes a new instance of the variable class. See [Mercury.Interpreting.Alternative](#)'s constructor for more details.
- override bool [Equals \(object obj\)](#)
- override int [GetHashCode \(\)](#)
- override string [ToString \(\)](#)
- bool [Equals \(IElement other\)](#)
- bool [Equals \(Variable other\)](#)

Static Public Member Functions

- static bool [IsValidName](#) (string name)
Determines whether the specified string is a valid variable name.

Properties

- string [Name](#) [get, set]
Name of the variable
- [ElementType](#) [Type](#) [get, set]

Additional Inherited Members

6.77.1 Detailed Description

Represents a variable element that captures value and can be used on the RHS of the RewriteRule as an argument
Definition at line 261 of file Element.cs.

6.77.2 Constructor & Destructor Documentation

6.77.2.1 [Mercury.Interpreting.Variable.Variable](#) ()

Creates an anonymous variable, an equivalent of [Mercury.Interpreting.Wildcard](#) that matches exactly one symbol
Definition at line 269 of file Element.cs.

6.77.2.2 [Mercury.Interpreting.Variable.Variable](#) (string name, SymbolType ws, IEnumerable< Symbol > symbols)

Initializes a new instance of the variable class. See [Mercury.Interpreting.Alternative](#)'s constructor for more details.

Parameters

<i>name</i>	The name of variable.
<i>ws</i>	The symbol mask.
<i>symbols</i>	The symbols.

Exceptions

System.ArgumentException	
--	--

Definition at line 282 of file Element.cs.

6.77.3 Member Function Documentation

6.77.3.1 override bool [Mercury.Interpreting.Variable.Equals](#) (object obj)

Definition at line 318 of file Element.cs.

6.77.3.2 bool [Mercury.Interpreting.Variable.Equals](#) (IElement other)

Definition at line 342 of file Element.cs.

6.77.3.3 bool Mercury.Interpreting.Variable.Equals (Variable *other*)

Definition at line 349 of file Element.cs.

6.77.3.4 override int Mercury.Interpreting.Variable.GetHashCode ()

Definition at line 324 of file Element.cs.

6.77.3.5 static bool Mercury.Interpreting.Variable.IsValidName (string *name*) [static]

Determines whether the specified string is a valid variable name.

- contains only letters or digits
- the first character is letter

Parameters

<i>name</i>	The name.
-------------	-----------

Returns

true if the name is valid, otherwise it returns (surprisingly) false.

Definition at line 307 of file Element.cs.

6.77.3.6 override string Mercury.Interpreting.Variable.ToString ()

Definition at line 327 of file Element.cs.

6.77.4 Property Documentation

6.77.4.1 string Mercury.Interpreting.Variable.Name [get], [set]

Name of the variable

Definition at line 293 of file Element.cs.

6.77.4.2 ElementType Mercury.Interpreting.Variable.Type [get], [set]

Definition at line 336 of file Element.cs.

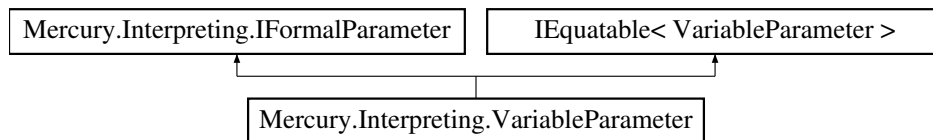
The documentation for this class was generated from the following file:

- Mercury/Mercury/Interpreting/[Element.cs](#)

6.78 Mercury.Interpreting.VariableParameter Class Reference

Represents a named parameter that will be replaced by an actual value of a variable captured by interpreter.

Inheritance diagram for Mercury.Interpreting.VariableParameter:



Public Member Functions

- [VariableParameter](#) (string name)
Creates a new [Mercury.Interpreting.VariableParameter](#) instance.
- override bool [Equals](#) (object obj)
- override int [GetHashCode](#) ()
- override string [ToString](#) ()
- bool [Equals](#) ([VariableParameter](#) other)

Properties

- string [Name](#) [get, set]
[Variable](#) name.
- [ParameterType](#) Type [get, set]

6.78.1 Detailed Description

Represents a named parameter that will be replaced by an actual value of a variable captured by interpreter.

Definition at line 160 of file FormalParameter.cs.

6.78.2 Constructor & Destructor Documentation

6.78.2.1 Mercury.Interpreting.VariableParameter.VariableParameter (string name)

Creates a new [Mercury.Interpreting.VariableParameter](#) instance.

Parameters

<i>name</i>	The name of the paramter.
-------------	---------------------------

Exceptions

<i>System.ArgumentNull↔ Exception</i>	When name is not valid.
---	-------------------------

Definition at line 169 of file FormalParameter.cs.

6.78.3 Member Function Documentation

6.78.3.1 override bool Mercury.Interpreting.VariableParameter.Equals (object obj)

Definition at line 187 of file FormalParameter.cs.

6.78.3.2 bool Mercury.Interpreting.VariableParameter.Equals (VariableParameter other)

Definition at line 211 of file FormalParameter.cs.

6.78.3.3 override int Mercury.Interpreting.VariableParameter.GetHashCode ()

Definition at line 193 of file FormalParameter.cs.

6.78.3.4 override string Mercury.Interpreting.VariableParameter.ToString ()

Definition at line 196 of file FormalParameter.cs.

6.78.4 Property Documentation

6.78.4.1 string Mercury.Interpreting.VariableParameter.Name [get], [set]

[Variable](#) name.

Definition at line 181 of file FormalParameter.cs.

6.78.4.2 ParameterType Mercury.Interpreting.VariableParameter.Type [get], [set]

Definition at line 205 of file FormalParameter.cs.

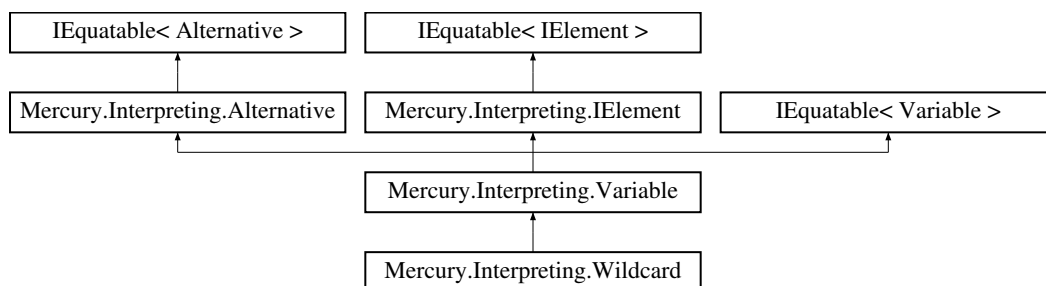
The documentation for this class was generated from the following file:

- Mercury/Mercury/Interpreting/[FormalParameter.cs](#)

6.79 Mercury.Interpreting.Wildcard Class Reference

An element that matches zero or more values. Can be used as a variable, but the actioncall must be of arity (0, `int.MAX_VALUE`) and must accept `Tree` or `Symbol` as the argument.

Inheritance diagram for Mercury.Interpreting.Wildcard:



Public Member Functions

- [Wildcard](#) ()
Creates a new [Wildcard](#) with a random name
- [Wildcard](#) (string name, [SymbolType](#) ws, IEnumerable< [Symbol](#) > symbols)
Creates a new the [Wildcard](#). See the [Mercury.Interpreting.Alternative](#)'s constructor for more details.
- override bool [Equals](#) (object obj)
- override int [GetHashCode](#) ()
- override string [ToString](#) ()
- new bool [Equals](#) ([IElement](#) other)
- bool [Equals](#) ([Wildcard](#) other)

Additional Inherited Members

6.79.1 Detailed Description

An element that matches zero or more values. Can be used as a variable, but the actioncall must be of arity (0, `int.MAX_VALUE`) and must accept `Tree` or `Symbol` as the argument.

Definition at line 370 of file `Element.cs`.

6.79.2 Constructor & Destructor Documentation

6.79.2.1 `Mercury.Interpreting.Wildcard.Wildcard ( )`

Creates a new [Wildcard](#) with a random name

Definition at line 375 of file `Element.cs`.

6.79.2.2 `Mercury.Interpreting.Wildcard.Wildcard ( string name, SymbolType ws, IEnumerable< Symbol > symbols )`

Creates a new the [Wildcard](#). See the [Mercury.Interpreting.Alternative](#)'s constructor for more details.

Parameters

<i>name</i>	The name of variable.
<i>ws</i>	
<i>symbols</i>	The symbols.

Definition at line 387 of file `Element.cs`.

6.79.3 Member Function Documentation

6.79.3.1 `override bool Mercury.Interpreting.Wildcard.Equals ( object obj )`

Definition at line 395 of file `Element.cs`.

6.79.3.2 `new bool Mercury.Interpreting.Wildcard.Equals ( IElement other )`

Definition at line 413 of file `Element.cs`.

6.79.3.3 `bool Mercury.Interpreting.Wildcard.Equals ( Wildcard other )`

Definition at line 420 of file `Element.cs`.

6.79.3.4 `override int Mercury.Interpreting.Wildcard.GetHashCode ( )`

Definition at line 401 of file `Element.cs`.

6.79.3.5 `override string Mercury.Interpreting.Wildcard.ToString ( )`

Definition at line 404 of file `Element.cs`.

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

- `Mercury/Mercury/Interpreting/Element.cs`

Chapter 7

File Documentation

7.1 Mercury/Mercury/Analyser.cs File Reference

Classes

- class [Mercury.Analyser< T, TResult >](#)
Wrapper class for each step of analysis.
- class [Mercury.Analyser< T >](#)
Wrapper class for each step of analysis

Namespaces

- package [Mercury](#)

7.2 Mercury/Mercury/AnalyserResult.cs File Reference

Classes

- class [Mercury.AnalyserResult< T, TResult >](#)
Holds the result of analysis.
The fields hold information according to the following table:

Namespaces

- package [Mercury](#)

Enumerations

- enum [Mercury.AnalyserStatus](#) {
[Mercury.AnalyserStatus.Success](#), [Mercury.AnalyserStatus.TokenizerFail](#), [Mercury.AnalyserStatus.Scanner↵
Fail](#), [Mercury.AnalyserStatus.ParserFail](#),
[Mercury.AnalyserStatus.InterpreterFail](#), [Mercury.AnalyserStatus.Undefined](#) }
The status of analysis. If failed, see [FailReason](#)

- enum [Mercury.FailReason](#) { [Mercury.FailReason.None](#), [Mercury.FailReason.Exception](#), [Mercury.FailReason.NoResult](#) }

Specifies the reason of analysis failure

7.3 Mercury/Mercury/Defaults.cs File Reference

Classes

- class [Mercury.Defaults](#)
Global class containing cool static / convenience stuff.

Namespaces

- package [Mercury](#)

7.4 Mercury/Mercury/Exceptions/InterpretingExceptions.cs File Reference

Classes

- class [Mercury.Exceptions.InterpretingException](#)
An exception thrown because of problems in the [Interpreting](#) namespace.
- class [Mercury.Exceptions.InvalidActionCallException](#)
An exception representing a problem with [Mercury.Interpreting.ActionCall{T}](#)
- class [Mercury.Exceptions.InvalidElementException](#)
An exception representing a problem with [Mercury.Interpreting.IElement](#) descendants.
- class [Mercury.Exceptions.InvalidActionException](#)
An exception representing a problem with [Mercury.Interpreting.InterpreterAction{T}](#) class.
- class [Mercury.Exceptions.InvalidActionArgumentException](#)
An exception representing a problem while type checking the [Mercury.Interpreting.RewriteRule{T}](#).
- class [Mercury.Exceptions.InvalidRewriteRuleException](#)
An exception representing a problem with [Mercury.Interpreting.RewriteRule{T}](#).
- class [Mercury.Exceptions.ErrorActionException](#)
Exception thrown by basic actions when a problem was encountered.

Namespaces

- package [Mercury.Exceptions](#)

7.5 Mercury/Mercury/Exceptions/MercuryException.cs File Reference

Classes

- class [Mercury.Exceptions.MercuryException](#)
An exception thrown by the [Mercury](#) library.

Namespaces

- package [Mercury.Exceptions](#)

7.6 Mercury/Mercury/Exceptions/SyntaxExceptions.cs File Reference

Classes

- class [Mercury.Exceptions.SyntaxException](#)
An exception thrown when something wrong happens in the [Mercury.Syntax](#)
- class [Mercury.Exceptions.InvalidSymbolException](#)
An exception thrown when attempting to create an invalid [Mercury.Syntax.Symbol](#) instance.
- class [Mercury.Exceptions.InvalidRuleException](#)
An exception thrown when attempting to create an invalid [Mercury.Syntax.Rule](#) instance.
- class [Mercury.Exceptions.InvalidEdgeException](#)
An exception thrown when attempting to create an invalid [Mercury.Parser.Edge](#) instance.

Namespaces

- package [Mercury.Exceptions](#)

7.7 Mercury/Mercury/Extensions.cs File Reference

Classes

- class [Mercury.Extensions](#)
Provides extension methods for the [Mercury](#) library.

Namespaces

- package [Mercury](#)

7.8 Mercury/Mercury/Interpreting/ActionCall.cs File Reference

Classes

- class [Mercury.Interpreting.ActionCall< T >](#)
This class represents the application of [Mercury.Interpreting.InterpreterAction{T}](#) on the [Mercury.Interpreting.I-FormalParameter](#) list.

Namespaces

- package [Mercury.Interpreting](#)

7.9 Mercury/Mercury/Interpreting/Argument.cs File Reference

Classes

- class [Mercury.Interpreting.Argument](#)
Represents an argument passed to the [Mercury.Interpreting.InterpreterAction{T}](#), the actual parameter given to the [Mercury.Interpreting.Evaluator{T}](#) in place of a formal parameter.

Namespaces

- package [Mercury.Interpreting](#)

Enumerations

- enum [Mercury.Interpreting.ArgumentType](#) {
[Mercury.Interpreting.ArgumentType.None](#) = 0, [Mercury.Interpreting.ArgumentType.Integer](#) = 1, [Mercury.Interpreting.ArgumentType.Real](#) = 2, [Mercury.Interpreting.ArgumentType.Boolean](#) = 4,
[Mercury.Interpreting.ArgumentType.String](#) = 8, [Mercury.Interpreting.ArgumentType.Symbol](#) = 16, [Mercury.Interpreting.ArgumentType.Tree](#) = 32, [Mercury.Interpreting.ArgumentType.TValue](#) = 64,
[Mercury.Interpreting.ArgumentType.Primitive](#) = Integer | Real | String | Boolean, [Mercury.Interpreting.ArgumentType.ParserEntity](#) = Symbol | Tree, [Mercury.Interpreting.ArgumentType.Any](#) = Primitive | ParserEntity | TValue, [Mercury.Interpreting.ArgumentType.Null](#) = 128,
[Mercury.Interpreting.ArgumentType.Lazy](#) = 256 | Null, [Mercury.Interpreting.ArgumentType.Flags](#) = Null | Lazy
 }

Defines argument types using in the [Mercury](#) library

7.10 Mercury/Mercury/Interpreting/BasicActions.cs File Reference

Classes

- class [Mercury.Interpreting.BasicActions< T >](#)
Contains basic actions for interpretation.
Type specifiers of actions are defined as follows:
- class [Mercury.Interpreting.OperationAction< T, U, V >](#)
Represents an operation on two values.
- class [Mercury.Interpreting.FoldingAction< T, U, V >](#)
"Folds" all its arguments into one value. For instance, for "x y z" arguments this action computes $f(f(f(s, x), y), z)$ where f is the lamda function and s is the seed parameter of the constructor
- class [Mercury.Interpreting.ConvertAction< T, U, V >](#)
Applies the lambda function on a single value.
- class [Mercury.Interpreting.ConstantAction< T, U >](#)
Wraps a value as an action with no parameters
- class [Mercury.Interpreting.ArithmeticAction< T >](#)
Represents the arithmetic action. It is similar to the [Mercury.Interpreting.FoldingAction{T,U,V}](#) but does not require initial value. The minimal number of arguments is therefore 2. For instance, "1 2 3" would be evaluated as $f(f(1, 2), 3)$
- class [Mercury.Interpreting.JoinAction< T >](#)
- class [Mercury.Interpreting.ConcatAction< T >](#)
- class [Mercury.Interpreting.FlattenAction< T >](#)
- class [Mercury.Interpreting.SubTreeAction< T >](#)
- class [Mercury.Interpreting.IdentityAction< T >](#)
- class [Mercury.Interpreting.LetAction< T >](#)
- class [Mercury.Interpreting.VarAction< T >](#)
- class [Mercury.Interpreting.TryAction< T >](#)
- class [Mercury.Interpreting.IfAction< T >](#)
- class [Mercury.Interpreting.SelectAction< T >](#)
- class [Mercury.Interpreting.CaseAction< T >](#)
- class [Mercury.Interpreting.ArgsCountAction< T >](#)
- class [Mercury.Interpreting.EvaluateAction< T >](#)

- class **Mercury.Interpreting.FailAction**< T >
- class **Mercury.Interpreting.SimpleAction**< T >
- class **Mercury.Interpreting.TraceAction**< T >

Namespaces

- package [Mercury.Interpreting](#)

7.11 Mercury/Mercury/Interpreting/Element.cs File Reference

Classes

- interface [Mercury.Interpreting.IElement](#)
Element interface
- class [Mercury.Interpreting.Alternative](#)
Represents an alternative of elements. It may match specified symbols or symboltypes (Nonterminals, ...).
- class [Mercury.Interpreting.Constant](#)
Constant element. Basically equivalent of [Mercury.Interpreting.Alternative](#).
- class [Mercury.Interpreting.Variable](#)
Represents a variable element that captures value and can be used on the RHS of the RewriteRule as an argument
- class [Mercury.Interpreting.Wildcard](#)
An element that matches zero or more values. Can be used as a variable, but the actioncall must be of arity (0, `int.MAX_VALUE`) and must accept `Tree` or `Symbol` as the argument.
- class [Mercury.Interpreting.Structure](#)
Represents a tree of [Mercury.Interpreting.IElement](#) instances. Must match precisely to the parse tree.

Namespaces

- package [Mercury.Interpreting](#)

Enumerations

- enum [Mercury.Interpreting.ElementType](#) { [Mercury.Interpreting.ElementType.Constant](#), [Mercury.Interpreting.ElementType.Variable](#), [Mercury.Interpreting.ElementType.Wildcard](#), [Mercury.Interpreting.ElementType.Structure](#) }
Element Type enumeration

7.12 Mercury/Mercury/Interpreting/Entries.cs File Reference

Classes

- interface [Mercury.Interpreting.DataEntry](#)
Entry base (dummy) interface.
- class [Mercury.Interpreting.TextEntry](#)
Stores informations logged by actions or internal Evaluator.
- class [Mercury.Interpreting.TreeEntry](#)< T >
Represents an entry for tree interpretation.
- class [Mercury.Interpreting.MemoEntry](#)< T >
Represents memoized value

- class [Mercury.Interpreting.RuleEntry< T >](#)
Describes a rule that has been used to interpret a tree.
- class [Mercury.Interpreting.EntryWriter< T >](#)
An auxiliary class that formats the [DataEntry](#) structure and writes it into a stream.

Namespaces

- package [Mercury.Interpreting](#)

7.13 Mercury/Mercury/Interpreting/Evaluator.cs File Reference

Classes

- class **Mercury.Interpreting.Evaluator< T >**
Carries out the actual evaluation of [Mercury.Interpreting.InterpreterAction{T}](#) and its arguments arguments.

Namespaces

- package [Mercury.Interpreting](#)

7.14 Mercury/Mercury/Interpreting/FormalParameter.cs File Reference

Classes

- interface [Mercury.Interpreting.IFormalParameter](#)
Represents formal parameters.
- class [Mercury.Interpreting.ConstantParameter](#)
[Constant](#) parameter.
- class [Mercury.Interpreting.VariableParameter](#)
Represents a named parameter that will be replaced by an actual value of a variable captured by interpreter.

Namespaces

- package [Mercury.Interpreting](#)

Enumerations

- enum [Mercury.Interpreting.ParameterType](#) {
[Mercury.Interpreting.ParameterType.IntegralConstant](#), [Mercury.Interpreting.ParameterType.RealConstant](#),
[Mercury.Interpreting.ParameterType.StringConstant](#), [Mercury.Interpreting.ParameterType.BooleanConstant](#),
[Mercury.Interpreting.ParameterType.Variable](#), [Mercury.Interpreting.ParameterType.ActionCall](#) }
Formal Parameter Type

7.15 Mercury/Mercury/Interpreting/Instance.cs File Reference

Classes

- class [Mercury.Interpreting.Instance](#)

Represents a binding between a [Mercury.Interpreting.Variable](#) and its value

- class [Mercury.Interpreting.InstanceList](#)

Represents the list of instances

Namespaces

- package [Mercury.Interpreting](#)

7.16 Mercury/Mercury/Interpreting/Interpreter.cs File Reference

Classes

- interface [Mercury.Interpreting.Interpreter< T >](#)

Interface for interpreters that create objects from derivation trees according to RewriteRules.

- class [Mercury.Interpreting.Interpreter< T >](#)

Interprets derivation trees using rewrite rules. The type parameter T represents the target language.

- class [Mercury.Interpreting.SemanticInterpreter< T >](#)

Represent the semantic analysis with interpretation. Consists of two interpreters, the first one rewrites parse trees, the second one interprets them. This class serves as a wrapper so it could be used in the Mercury.Analyser{T,TResult} class.

Namespaces

- package [Mercury.Interpreting](#)

7.17 Mercury/Mercury/Interpreting/InterpreterAction.cs File Reference

Classes

- class [Mercury.Interpreting.InterpreterAction< T >](#)

Function that wraps constant values in the RHS

- class [Mercury.Interpreting.InterpreterAction< T, U >](#)

Interpreter action with a safer return type.

- class [Mercury.Interpreting.GenericAction< T >](#)

Represents an action that infers its return type only after it is provided with types of its arguments.

Namespaces

- package [Mercury.Interpreting](#)

Functions

- delegate T [Mercury.Interpreting.RecursiveEval< T >](#) (Tree< [Symbol](#) > tree, InterpreterContext< T > context)

Delegate type for recursive evaluation (interpretation)

7.18 Mercury/Mercury/Interpreting/InterpreterContext.cs File Reference

Classes

- class [Mercury.Interpreting.InterpreterContext< T >](#)
Interpreter Context class. Some languages may use it to override it and remember data when parse trees are being interpreted.
- interface [Mercury.Interpreting.IInterpreterContextFactory](#)
Factory of context.
- class **Mercury.Interpreting.InternalContext< T >**
- class [Mercury.Interpreting.EmptyContextFactory](#)
Produces contexts with no custom data

Namespaces

- package [Mercury.Interpreting](#)

7.19 Mercury/Mercury/Interpreting/InterpreterResult.cs File Reference

Classes

- class [Mercury.Interpreting.InterpreterResult< T >](#)
Result of the interpretation.

Namespaces

- package [Mercury.Interpreting](#)

7.20 Mercury/Mercury/Interpreting/RewriteRule.cs File Reference

Classes

- class **Mercury.Interpreting.RewriteRuleValidator< T >**
- class [Mercury.Interpreting.RewriteRule< T >](#)
Interpreter rewrite rule

Namespaces

- package [Mercury.Interpreting](#)

7.21 Mercury/Mercury/Interpreting/RewriteRuleCollection.cs File Reference

Classes

- class [Mercury.Interpreting.RewriteRuleCollection< T >](#)
A read-only collection of rewrite rules
- class [Mercury.Interpreting.RewriteRuleCollectionBuilder< T >](#)
Can be used to construct Mercury.Interpreting.RewriteRuleCollection{T}

Namespaces

- package [Mercury.Interpreting](#)

7.22 Mercury/Mercury/LanguageInfo.cs File Reference

Classes

- class [Mercury.LanguageInfo< T >](#)
Represents a collection of information required to analyze a language with [Mercury](#) [Mercury.Analyser{T,TValue}](#).

Namespaces

- package [Mercury](#)

7.23 Mercury/Mercury/Nucleus/Collections/MultiDictionary.cs File Reference

Classes

- interface [Mercury.Nucleus.Collections.IReadOnlyMultiDictionary< TKey, TValue >](#)
Represents a multi-valued read-only dictionary, a map between a key and a collection of values.
- interface [Mercury.Nucleus.Collections.IMultiDictionary< TKey, TValue >](#)
Represents a multi-valued dictionary, in other words a map between a key and a collection of values.
- class [Mercury.Nucleus.Collections.SetMultiDictionary< TKey, TValue >](#)
An implementation of [IMultiDictionary{TKey,TValue}](#) using [T:System.Collections.Generic.HashSet{T}](#) as an underlying collection of values.
- class [Mercury.Nucleus.Collections.ListMultiDictionary< TKey, TValue >](#)
An implementation of [IMultiDictionary{TKey,TValue}](#) using [T:System.Collections.Generic.List{T}](#) as an underlying collection of values.
- class [Mercury.Nucleus.Collections.UniqueListMultiDictionary< TKey, TValue >](#)
An implementation of [IMultiDictionary{TKey,TValue}](#) using [T:System.Collections.Generic.List{T}](#) as underlying collection, but opposed to [ListMultiDictionary{TKey,TValue}](#) does not allow duplicate values associated with the same key

Namespaces

- package [Mercury.Nucleus.Collections](#)

7.24 Mercury/Mercury/Nucleus/Trees/Tree.cs File Reference

Classes

- class [Mercury.Nucleus.Trees.Tree< T >](#)
A non-balanced tree implementation that can be referenced from many other trees. Because of that, there is no way to get back to the parent node.

Namespaces

- package [Mercury.Nucleus.Trees](#)

7.25 Mercury/Mercury/Nucleus/Trees/TreeEnumerators.cs File Reference

Classes

- class [Mercury.Nucleus.Trees.DFSTreeEnumerator< T >](#)
Depth-First Search tree enumerator
- class [Mercury.Nucleus.Trees.BFSTreeEnumerator< T >](#)
Breadth-First Search tree enumerator

Namespaces

- package [Mercury.Nucleus.Trees](#)

7.26 Mercury/Mercury/obj/Debug/TemporaryGeneratedFile_036C0B5B-1481-4323-8D20-8F5ADCB23D92.cs File Reference

7.27 Mercury/Mercury/obj/Release/TemporaryGeneratedFile_036C0B5B-1481-4323-8D20-8F5ADCB23D92.cs File Reference

7.28 Mercury/Mercury/obj/Debug/TemporaryGeneratedFile_5937a670-0e60-4077-877b-f7221da3dda1.cs File Reference

7.29 Mercury/Mercury/obj/Release/TemporaryGeneratedFile_5937a670-0e60-4077-877b-f7221da3dda1.cs File Reference

7.30 Mercury/Mercury/obj/Debug/TemporaryGeneratedFile_E7A71F73-0F8D-4B9B-B56E-8E70B10BC5D3.cs File Reference

7.31 Mercury/Mercury/obj/Release/TemporaryGeneratedFile_E7A71F73-0F8D-4B9B-B56E-8E70B10BC5D3.cs File Reference

7.32 Mercury/Mercury/Properties/AssemblyInfo.cs File Reference

7.33 Mercury/Mercury/Scanning/Token.cs File Reference

Classes

- class [Mercury.Scanning.Token](#)
Token, a part of an input text recognized by tokenizer

Namespaces

- package [Mercury.Scanning](#)

Enumerations

- enum [Mercury.Scanning.TokenType](#) { [Mercury.Scanning.TokenType.String](#), [Mercury.Scanning.TokenType.L←Symbol](#), [Mercury.Scanning.TokenType.Keyword](#), [Mercury.Scanning.TokenType.Number](#) }
- Defines token types. The actual conditions depend on the [Mercury.Scanning.ITokenizer](#) implementation.*

7.34 Mercury/Mercury/Scanning/Tokenizer.cs File Reference

Classes

- interface [Mercury.Scanning.ITokenizer](#)
The tokenizer interface.
- class [Mercury.Scanning.Tokenizer](#)
A default implementation of [ITokenizer](#) interface using Microsoft Regular Expressions.

Splits tokens in the following fashion:

Namespaces

- package [Mercury.Scanning](#)

Enumerations

- enum [Mercury.Scanning.TokenizerOptions](#) {
[Mercury.Scanning.TokenizerOptions.None](#) = 0, [Mercury.Scanning.TokenizerOptions.CompiledMatch](#) = 1,
[Mercury.Scanning.TokenizerOptions.IgnoreNumbers](#) = 2, [Mercury.Scanning.TokenizerOptions.IgnoreCase](#) =
4,
[Mercury.Scanning.TokenizerOptions.IgnoreQuotes](#) = 8, [Mercury.Scanning.TokenizerOptions.IgnoreSigns](#) =
16, [Mercury.Scanning.TokenizerOptions.StrictNumbers](#) = 32 }
- Options for the tokenizer*

7.35 Mercury/Mercury/Syntax/Edge.cs File Reference

Classes

- class [Mercury.Syntax.Edge](#)
Represents an edge in the chart created by the [ChartParser](#). This is an immutable class.
The [Edge](#) contains four values:

Namespaces

- package [Mercury.Syntax](#)

7.36 Mercury/Mercury/Syntax/EdgeDerivation.cs File Reference

Classes

- class **[Mercury.Syntax.EdgeDerivation](#)**
[Edge](#) Derivation

Namespaces

- package [Mercury.Syntax](#)

7.37 Mercury/Mercury/Syntax/Grammar.cs File Reference

Classes

- class [Mercury.Syntax.Grammar](#)
Represents a Context-Free [Grammar](#). This is an immutable class, to create a grammar use [Mercury.Syntax.GrammarBuilder](#).
- class **Mercury.Syntax.ExtendedGrammar**
Represents a Context-Free [Grammar](#) with convenience structures for [ChartParser](#).

Namespaces

- package [Mercury.Syntax](#)

7.38 Mercury/Mercury/Syntax/GrammarBuilder.cs File Reference

Classes

- class [Mercury.Syntax.GrammarBuilder](#)
Class that creates grammars.
- class **Mercury.Syntax.ExtendedGrammarBuilder**

Namespaces

- package [Mercury.Syntax](#)

7.39 Mercury/Mercury/Syntax/Parser.cs File Reference

Classes

- interface [Mercury.Syntax.IParser](#)
Creates a derivation tree from the list of input tokens
- class **Mercury.Syntax.ChartParserInternal**
The Bottom-Up Chart Parser implementation.
- class [Mercury.Syntax.ChartParser](#)
Thread-safe wrapper for ChartParserInternal class.

Namespaces

- package [Mercury.Syntax](#)

7.40 Mercury/Mercury/Syntax/ParserResult.cs File Reference

Classes

- class [Mercury.Syntax.ParserResult](#)
Represents the result returned by the parser.

Namespaces

- package [Mercury.Syntax](#)

7.41 Mercury/Mercury/Syntax/Rule.cs File Reference

Classes

- class [Mercury.Syntax.Rule](#)
Represents a Context-Free [Grammar](#) rule. This is an immutable class.

Namespaces

- package [Mercury.Syntax](#)

7.42 Mercury/Mercury/Syntax/Scanner.cs File Reference

Classes

- interface [Mercury.Syntax.IScanner](#)
Scanner scans input tokens and creates additional rules for parsing.
- class [Mercury.Syntax.CompositeScanner](#)
[Mercury.Syntax.IScanner](#) implementation that behaves like container of scanners using the Composite Design Pattern.
- class [Mercury.Syntax.BasicScannerMap](#)
Contains mapping between [TokenType](#) and [Mercury.Syntax.Mapping](#) of [Mercury.Syntax.BasicScanner](#).
- class [Mercury.Syntax.BasicScanner](#)
Generates four Nonterminals [@Symbol](#), [@Keyword](#), [@Number](#) and [@String](#) based on the given map and types of input tokens.

Namespaces

- package [Mercury.Syntax](#)

Enumerations

- enum [Mercury.Syntax.Mapping](#) {
 [Mercury.Syntax.Mapping.None](#) = 0, [Mercury.Syntax.Mapping.AsSymbol](#) = 1, [Mercury.Syntax.Mapping.AsKeyword](#) = 2, [Mercury.Syntax.Mapping.AsNumber](#) = 4,
 [Mercury.Syntax.Mapping.AsString](#) = 8, [Mercury.Syntax.Mapping.All](#) = 15 }

7.43 Mercury/Mercury/Syntax/Symbol.cs File Reference

Classes

- class [Mercury.Syntax.Symbol](#)

Represents a single symbol of the grammar. This is an immutable class.

For a symbol name to be valid it must NOT contain control or separator characters, must not be empty and must meet these conditions:

Namespaces

- package [Mercury.Syntax](#)

Enumerations

- enum [Mercury.Syntax.SymbolType](#) {
 [Mercury.Syntax.SymbolType.None](#) = 0, [Mercury.Syntax.SymbolType.Epsilon](#) = 1, [Mercury.Syntax.SymbolType.Terminal](#) = 2, [Mercury.Syntax.SymbolType.Nonterminal](#) = 4,
 [Mercury.Syntax.SymbolType.Any](#) = Epsilon | Terminal | Nonterminal }

Represents a type of symbol. Implemented as Flags since Symbol type inference is ambiguous.

Index

ActionCall
 Mercury::Interpreting, 18

All
 Mercury::Syntax, 22

Any
 Mercury::Interpreting, 18
 Mercury::Syntax, 22

AsKeyword
 Mercury::Syntax, 22

AsNumber
 Mercury::Syntax, 22

AsString
 Mercury::Syntax, 22

AsSymbol
 Mercury::Syntax, 22

Boolean
 Mercury::Interpreting, 18

BooleanConstant
 Mercury::Interpreting, 18

CompiledMatch
 Mercury::Scanning, 20

Constant
 Mercury::Interpreting, 18

Epsilon
 Mercury::Syntax, 22

Exception
 Mercury, 14

Flags
 Mercury::Interpreting, 18

IgnoreCase
 Mercury::Scanning, 20

IgnoreNumbers
 Mercury::Scanning, 20

IgnoreQuotes
 Mercury::Scanning, 20

IgnoreSigns
 Mercury::Scanning, 20

Integer
 Mercury::Interpreting, 17

IntegralConstant
 Mercury::Interpreting, 18

InterpreterFail
 Mercury, 14

Keyword
 Mercury::Scanning, 20

Lazy
 Mercury::Interpreting, 18

Mercury, 13
 Exception, 14
 InterpreterFail, 14
 NoResult, 14
 None, 14
 ParserFail, 14
 ScannerFail, 14
 Success, 14
 TokenizerFail, 14
 Undefined, 14

Mercury::Interpreting
 ActionCall, 18
 Any, 18
 Boolean, 18
 BooleanConstant, 18
 Constant, 18
 Flags, 18
 Integer, 17
 IntegralConstant, 18
 Lazy, 18
 None, 17
 Null, 18
 ParserEntity, 18
 Primitive, 18
 Real, 17
 RealConstant, 18
 String, 18
 StringConstant, 18
 Structure, 18
 Symbol, 18
 TValue, 18
 Tree, 18
 Variable, 18
 Wildcard, 18

Mercury::Scanning
 CompiledMatch, 20
 IgnoreCase, 20
 IgnoreNumbers, 20
 IgnoreQuotes, 20
 IgnoreSigns, 20
 Keyword, 20
 None, 20
 Number, 21
 StrictNumbers, 20
 String, 20
 Symbol, 20

Mercury::Syntax

- All, [22](#)
- Any, [22](#)
- AsKeyword, [22](#)
- AsNumber, [22](#)
- AsString, [22](#)
- AsSymbol, [22](#)
- Epsilon, [22](#)
- None, [22](#)
- Nonterminal, [22](#)
- Terminal, [22](#)
- NoResult
 - Mercury, [14](#)
- None
 - Mercury, [14](#)
 - Mercury::Interpreting, [17](#)
 - Mercury::Scanning, [20](#)
 - Mercury::Syntax, [22](#)
- Nonterminal
 - Mercury::Syntax, [22](#)
- Null
 - Mercury::Interpreting, [18](#)
- Number
 - Mercury::Scanning, [21](#)
- ParserEntity
 - Mercury::Interpreting, [18](#)
- ParserFail
 - Mercury, [14](#)
- Primitive
 - Mercury::Interpreting, [18](#)
- Real
 - Mercury::Interpreting, [17](#)
- RealConstant
 - Mercury::Interpreting, [18](#)
- ScannerFail
 - Mercury, [14](#)
- StrictNumbers
 - Mercury::Scanning, [20](#)
- String
 - Mercury::Interpreting, [18](#)
 - Mercury::Scanning, [20](#)
- StringConstant
 - Mercury::Interpreting, [18](#)
- Structure
 - Mercury::Interpreting, [18](#)
- Success
 - Mercury, [14](#)
- Symbol
 - Mercury::Interpreting, [18](#)
 - Mercury::Scanning, [20](#)
- TValue
 - Mercury::Interpreting, [18](#)
- Terminal
 - Mercury::Syntax, [22](#)
- TokenizerFail
 - Mercury, [14](#)
- Tree
 - Mercury::Interpreting, [18](#)
- Undefined
 - Mercury, [14](#)
- Variable
 - Mercury::Interpreting, [18](#)
- Wildcard
 - Mercury::Interpreting, [18](#)