

DynGenPar

Generated by Doxygen 1.7.4

Fri Mar 2 2012 04:45:01

Contents

1	Main Page	1
1.1	Introduction	1
1.2	Basic Usage	1
1.3	Java Bindings	3
1.4	Features	5
1.4.1	Dynamic Grammar Additions	5
1.4.2	Incremental Parsing	5
1.4.3	Prediction	6
1.4.4	Rule Labels	6
1.4.5	Custom Parse Actions	6
1.4.6	Arbitrary Token Sources	6
1.4.6.1	Flex Lexer Token Source	6
1.4.7	Hierarchical Parsing	6
1.4.8	Parallel Multiple Context-Free Grammars (PMCFGs)	7
1.4.8.1	Grammatical Framework (GF) PGF Grammars	7
1.4.9	Next Token Constraints	7
1.5	TODO	7
1.6	Copyright	7
1.6.1	Acknowledgments	8
1.6.2	Licensing and Warranty Disclaimer	8
2	Namespace Index	9
2.1	Namespace List	9

3	Class Index	11
3.1	Class Hierarchy	11
4	Class Index	13
4.1	Class List	13
5	File Index	15
5.1	File List	15
6	Namespace Documentation	17
6.1	DynGenPar Namespace Reference	17
6.1.1	Typedef Documentation	19
6.1.1.1	Cat	19
6.1.1.2	CatArg	20
6.1.1.3	ConstrainedMultiPredictions	20
6.1.1.4	ConstrainedPredictions	20
6.1.1.5	MultiPredictions	20
6.1.1.6	Predictions	20
6.1.1.7	RuleSet	20
6.1.1.8	TokenSet	20
6.1.2	Enumeration Type Documentation	21
6.1.2.1	PgfReservedTokens	21
6.1.3	Function Documentation	21
6.1.3.1	multiHashToListHash	21
6.1.3.2	parseTreeToPmcfgSyntaxTree	21
6.1.3.3	qHash	21
6.1.3.4	qHash	21
6.1.4	Variable Documentation	21
6.1.4.1	PreludeBind	21

7	Class Documentation	23
7.1	DynGenPar::AbstractLexerStateData Class Reference	23
7.1.1	Detailed Description	23
7.1.2	Constructor & Destructor Documentation	24
7.1.2.1	~AbstractLexerStateData	24
7.1.3	Member Function Documentation	24
7.1.3.1	clone	24
7.2	DynGenPar::Action Class Reference	24
7.2.1	Detailed Description	24
7.2.2	Constructor & Destructor Documentation	24
7.2.2.1	~Action	24
7.2.3	Member Function Documentation	24
7.2.3.1	execute	24
7.3	DynGenPar::ActionInfo Struct Reference	25
7.3.1	Detailed Description	25
7.3.2	Constructor & Destructor Documentation	25
7.3.2.1	ActionInfo	25
7.3.2.2	ActionInfo	25
7.3.3	Member Data Documentation	25
7.3.3.1	tree	25
7.4	DynGenPar::Alternative Class Reference	26
7.4.1	Detailed Description	26
7.4.2	Constructor & Destructor Documentation	26
7.4.2.1	Alternative	26
7.4.2.2	Alternative	27
7.4.2.3	Alternative	27
7.4.2.4	Alternative	27
7.4.3	Member Function Documentation	27
7.4.3.1	add	27

7.4.3.2	label	27
7.4.3.3	operator+=	27
7.4.3.4	operator+=	27
7.4.3.5	operator<<	27
7.4.3.6	operator<<	27
7.4.3.7	setLabel	28
7.4.3.8	toList	28
7.5	DynGenPar::ByteTokenSource Class Reference	28
7.5.1	Detailed Description	29
7.5.2	Constructor & Destructor Documentation	29
7.5.2.1	ByteTokenSource	29
7.5.2.2	ByteTokenSource	29
7.5.2.3	~ByteTokenSource	29
7.5.3	Member Function Documentation	29
7.5.3.1	readToken	29
7.5.3.2	reset	30
7.5.3.3	rewindTo	30
7.5.3.4	setInputBuffer	30
7.5.3.5	setInputBytes	30
7.5.3.6	setInputFile	30
7.5.3.7	setInputStdin	30
7.5.3.8	setInputString	30
7.5.4	Member Data Documentation	31
7.5.4.1	stream	31
7.6	DynGenPar::Cfg Struct Reference	31
7.6.1	Detailed Description	31
7.6.2	Constructor & Destructor Documentation	31
7.6.2.1	Cfg	31
7.6.2.2	Cfg	32

7.6.3	Member Function Documentation	32
7.6.3.1	addToken	32
7.6.3.2	isToken	32
7.6.4	Member Data Documentation	32
7.6.4.1	rules	32
7.6.4.2	startCat	32
7.6.4.3	tokens	32
7.7	DynGenPar::ConstrainedMultiPrediction Struct Reference	32
7.7.1	Detailed Description	33
7.7.2	Constructor & Destructor Documentation	33
7.7.2.1	ConstrainedMultiPrediction	33
7.7.2.2	ConstrainedMultiPrediction	33
7.7.2.3	ConstrainedMultiPrediction	33
7.7.3	Member Function Documentation	34
7.7.3.1	operator==	34
7.7.4	Member Data Documentation	34
7.7.4.1	cat	34
7.7.4.2	fullLiteral	34
7.7.4.3	nextTokenConstraints	34
7.8	DynGenPar::FlexLexerTokenSource Class Reference	34
7.8.1	Detailed Description	35
7.8.2	Constructor & Destructor Documentation	35
7.8.2.1	FlexLexerTokenSource	35
7.8.2.2	~FlexLexerTokenSource	35
7.8.3	Member Function Documentation	35
7.8.3.1	readToken	35
7.8.4	Member Data Documentation	36
7.8.4.1	flexLexer	36
7.9	DynGenPar::FullRule Struct Reference	36

7.9.1	Detailed Description	36
7.9.2	Constructor & Destructor Documentation	36
7.9.2.1	FullRule	36
7.9.2.2	FullRule	37
7.9.3	Member Data Documentation	37
7.9.3.1	cat	37
7.9.3.2	epsilonSkipped	37
7.9.3.3	rule	37
7.9.3.4	ruleNo	37
7.10	DynGenPar::Function Class Reference	37
7.10.1	Detailed Description	38
7.10.2	Constructor & Destructor Documentation	38
7.10.2.1	Function	38
7.10.2.2	Function	38
7.10.3	Member Function Documentation	38
7.10.3.1	add	38
7.10.3.2	operator+=	38
7.10.3.3	operator+=	39
7.10.3.4	operator<<	39
7.10.3.5	operator<<	39
7.10.3.6	toList	39
7.11	DynGenPar::LexerState Class Reference	39
7.11.1	Detailed Description	39
7.11.2	Constructor & Destructor Documentation	39
7.11.2.1	LexerState	39
7.11.2.2	LexerState	40
7.11.3	Member Function Documentation	40
7.11.3.1	clear	40
7.11.3.2	data	40

7.11.3.3	isNull	40
7.11.3.4	operator==	40
7.12	DynGenPar::ListTokenSource Class Reference	40
7.12.1	Detailed Description	41
7.12.2	Constructor & Destructor Documentation	41
7.12.2.1	ListTokenSource	41
7.12.2.2	~ListTokenSource	41
7.12.3	Member Function Documentation	41
7.12.3.1	readToken	41
7.12.3.2	rewindTo	42
7.12.4	Member Data Documentation	42
7.12.4.1	inputTokens	42
7.13	DynGenPar::Match Struct Reference	42
7.13.1	Detailed Description	42
7.13.2	Constructor & Destructor Documentation	43
7.13.2.1	Match	43
7.13.2.2	Match	43
7.13.2.3	Match	43
7.13.3	Member Data Documentation	43
7.13.3.1	len	43
7.13.3.2	nextTokenConstraints	43
7.13.3.3	rueno	43
7.13.3.4	scope	43
7.13.3.5	tree	44
7.14	DynGenPar::MultiPrediction Struct Reference	44
7.14.1	Detailed Description	44
7.14.2	Constructor & Destructor Documentation	44
7.14.2.1	MultiPrediction	44
7.14.2.2	MultiPrediction	45

7.14.3	Member Function Documentation	45
7.14.3.1	operator==	45
7.14.4	Member Data Documentation	45
7.14.4.1	cat	45
7.14.4.2	fullLiteral	45
7.15	DynGenPar::NextTokenConstraints Struct Reference	45
7.15.1	Detailed Description	46
7.15.2	Member Function Documentation	46
7.15.2.1	operator==	46
7.15.3	Member Data Documentation	46
7.15.3.1	expect	46
7.15.3.2	taboo	46
7.16	DynGenPar::Node Struct Reference	47
7.16.1	Detailed Description	47
7.16.2	Constructor & Destructor Documentation	47
7.16.2.1	Node	47
7.16.2.2	Node	47
7.16.2.3	Node	47
7.16.3	Member Function Documentation	48
7.16.3.1	operator==	48
7.16.4	Member Data Documentation	48
7.16.4.1	cat	48
7.16.4.2	children	48
7.16.4.3	data	48
7.17	DynGenPar::Parser Class Reference	48
7.17.1	Detailed Description	51
7.17.2	Constructor & Destructor Documentation	51
7.17.2.1	Parser	51
7.17.2.2	~Parser	51

7.17.3 Member Function Documentation	51
7.17.3.1 addPmcfgrule	51
7.17.3.2 addRule	51
7.17.3.3 addToken	52
7.17.3.4 computeConstrainedMultiPredictions	52
7.17.3.5 computeConstrainedMultiPredictions	52
7.17.3.6 computeConstrainedMultiPredictionsJava	52
7.17.3.7 computeConstrainedPredictions	53
7.17.3.8 computeConstrainedPredictions	53
7.17.3.9 computeConstrainedPredictionsJava	53
7.17.3.10 computeMultiPredictions	53
7.17.3.11 computeMultiPredictions	54
7.17.3.12 computeMultiPredictionsJava	54
7.17.3.13 computePredictions	54
7.17.3.14 computePredictions	55
7.17.3.15 expandNonterminalPrediction	55
7.17.3.16 expandNonterminalPredictionC	55
7.17.3.17 expandNonterminalPredictionC	55
7.17.3.18 expandNonterminalPredictionC	56
7.17.3.19 expandNonterminalPredictionMulti	56
7.17.3.20 expandNonterminalPredictionMultiC	56
7.17.3.21 expandNonterminalPredictionMultiC	56
7.17.3.22 expandNonterminalPredictionMultiC	57
7.17.3.23 initCaches	57
7.17.3.24 isLiteral	57
7.17.3.25 isToken	57
7.17.3.26 loadCfg	57
7.17.3.27 loadPmcfgrule	58
7.17.3.28 parse	58

7.17.3.29 parse	58
7.17.4 Member Data Documentation	59
7.17.4.1 catComponents	59
7.17.4.2 componentCats	60
7.17.4.3 inputSource	60
7.17.4.4 pseudoCats	60
7.17.4.5 rules	60
7.17.4.6 startCat	61
7.17.4.7 tokens	61
7.18 DynGenPar::ParseState Struct Reference	61
7.18.1 Detailed Description	61
7.18.2 Constructor & Destructor Documentation	62
7.18.2.1 ParseState	62
7.18.2.2 ParseState	62
7.18.3 Member Function Documentation	62
7.18.3.1 reset	62
7.18.4 Member Data Documentation	62
7.18.4.1 errorPos	62
7.18.4.2 errorToken	62
7.18.4.3 incrementalMatches	62
7.18.4.4 incrementalPos	62
7.18.4.5 incrementalStacks	62
7.18.4.6 lexerState	63
7.19 DynGenPar::Pgf Struct Reference	63
7.19.1 Detailed Description	63
7.19.2 Constructor & Destructor Documentation	64
7.19.2.1 Pgf	64
7.19.2.2 Pgf	64
7.19.3 Member Data Documentation	64

7.19.3.1	catNames	64
7.19.3.2	componentNames	64
7.19.3.3	firstFunction	64
7.19.3.4	functionNames	64
7.19.3.5	pmcfg	65
7.19.3.6	suffixes	65
7.19.3.7	tokenHash	65
7.20	DynGenPar::PgParser Class Reference	65
7.20.1	Detailed Description	66
7.20.2	Constructor & Destructor Documentation	66
7.20.2.1	PgParser	66
7.20.2.2	PgParser	66
7.20.2.3	~PgParser	66
7.20.3	Member Function Documentation	66
7.20.3.1	catName	66
7.20.3.2	filterCoercionsFromSyntaxTree	67
7.20.3.3	filterCoercionsFromSyntaxTree	67
7.20.3.4	functionName	67
7.20.3.5	setInputBuffer	67
7.20.3.6	setInputBytes	67
7.20.3.7	setInputFile	67
7.20.3.8	setInputStdin	67
7.20.3.9	setInputString	67
7.20.4	Member Data Documentation	67
7.20.4.1	pgf	67
7.21	DynGenPar::Pmcfg Struct Reference	68
7.21.1	Detailed Description	68
7.21.2	Member Function Documentation	68
7.21.2.1	addFunction	68

7.21.2.2	lookupFunction	68
7.21.3	Member Data Documentation	69
7.21.3.1	cfRules	69
7.21.3.2	functionIndices	69
7.21.3.3	functionNames	69
7.21.3.4	functions	69
7.21.3.5	rules	69
7.21.3.6	startCat	70
7.21.3.7	tokens	70
7.22	DynGenPar::PmcfComponentInfo Struct Reference	70
7.22.1	Detailed Description	70
7.22.2	Constructor & Destructor Documentation	70
7.22.2.1	PmcfComponentInfo	70
7.22.2.2	PmcfComponentInfo	71
7.22.3	Member Data Documentation	71
7.22.3.1	argPositions	71
7.22.3.2	pmcfRule	71
7.23	DynGenPar::PseudoCatScope Class Reference	71
7.23.1	Detailed Description	71
7.23.2	Constructor & Destructor Documentation	72
7.23.2.1	PseudoCatScope	72
7.23.3	Member Function Documentation	72
7.23.3.1	data	72
7.23.3.2	hasMcfgConstraint	72
7.23.3.3	hasPConstraint	72
7.23.3.4	isNull	72
7.23.3.5	mcfgConstraint	72
7.23.3.6	mcfgConstraints	72
7.23.3.7	operator==	72

7.23.3.8	pConstraint	73
7.23.3.9	pConstraints	73
7.24	DynGenPar::PseudoCatScopeData Class Reference	73
7.24.1	Detailed Description	73
7.24.2	Member Data Documentation	74
7.24.2.1	mcfgConstraints	74
7.24.2.2	pConstraints	74
7.25	QList Class Reference	74
7.26	QSharedData Class Reference	75
7.27	DynGenPar::Rule Class Reference	75
7.27.1	Detailed Description	76
7.27.2	Constructor & Destructor Documentation	76
7.27.2.1	Rule	76
7.27.2.2	Rule	76
7.27.2.3	Rule	76
7.27.2.4	Rule	76
7.27.3	Member Function Documentation	77
7.27.3.1	add	77
7.27.3.2	label	77
7.27.3.3	operator+=	77
7.27.3.4	operator+=	77
7.27.3.5	operator<<	77
7.27.3.6	operator<<	77
7.27.3.7	setLabel	77
7.27.3.8	toList	77
7.27.4	Member Data Documentation	77
7.27.4.1	action	77
7.27.4.2	nextTokenConstraints	78
7.28	DynGenPar::Sequence Class Reference	78

7.28.1	Detailed Description	79
7.28.2	Constructor & Destructor Documentation	79
7.28.2.1	Sequence	79
7.28.2.2	Sequence	79
7.28.2.3	Sequence	79
7.28.2.4	Sequence	79
7.28.3	Member Function Documentation	79
7.28.3.1	add	79
7.28.3.2	operator+=	79
7.28.3.3	operator+=	79
7.28.3.4	operator<<	80
7.28.3.5	operator<<	80
7.28.3.6	toList	80
7.28.4	Member Data Documentation	80
7.28.4.1	nextTokenConstraints	80
7.29	DynGenPar::StackItem Class Reference	80
7.29.1	Detailed Description	81
7.29.2	Constructor & Destructor Documentation	81
7.29.2.1	StackItem	81
7.29.2.2	StackItem	81
7.29.2.3	StackItem	81
7.29.2.4	StackItem	81
7.29.2.5	StackItem	82
7.29.2.6	StackItem	82
7.29.2.7	StackItem	82
7.29.2.8	StackItem	82
7.29.3	Member Function Documentation	82
7.29.3.1	addParent	82
7.29.3.2	data	82

7.29.3.3	setParents	82
7.29.3.4	type	83
7.30	DynGenPar::StackItemData Class Reference	83
7.30.1	Detailed Description	83
7.30.2	Constructor & Destructor Documentation	83
7.30.2.1	~StackItemData	83
7.30.3	Member Function Documentation	84
7.30.3.1	addParent	84
7.30.3.2	clone	84
7.30.3.3	setParents	84
7.30.3.4	type	84
7.31	DynGenPar::StackItemType0 Class Reference	84
7.31.1	Detailed Description	85
7.31.2	Constructor & Destructor Documentation	85
7.31.2.1	StackItemType0	85
7.31.2.2	~StackItemType0	85
7.31.3	Member Function Documentation	85
7.31.3.1	addParent	85
7.31.3.2	cat	85
7.31.3.3	clone	86
7.31.3.4	effCat	86
7.31.3.5	parents	86
7.31.3.6	pos	86
7.31.3.7	scope	86
7.31.3.8	setParents	86
7.31.3.9	type	86
7.32	DynGenPar::StackItemType1 Class Reference	87
7.32.1	Detailed Description	87
7.32.2	Constructor & Destructor Documentation	88

7.32.2.1	StackItemType1	88
7.32.2.2	~StackItemType1	88
7.32.3	Member Function Documentation	88
7.32.3.1	addParent	88
7.32.3.2	cat	88
7.32.3.3	clone	88
7.32.3.4	effCat	88
7.32.3.5	parents	88
7.32.3.6	scope	88
7.32.3.7	setParents	89
7.32.3.8	type	89
7.33	DynGenPar::StackItemType2 Class Reference	89
7.33.1	Detailed Description	90
7.33.2	Constructor & Destructor Documentation	90
7.33.2.1	StackItemType2	90
7.33.2.2	~StackItemType2	90
7.33.3	Member Function Documentation	90
7.33.3.1	addParent	90
7.33.3.2	clone	90
7.33.3.3	parent	91
7.33.3.4	setParents	91
7.33.3.5	type	91
7.34	DynGenPar::StackItemType3 Class Reference	91
7.34.1	Detailed Description	92
7.34.2	Constructor & Destructor Documentation	92
7.34.2.1	StackItemType3	92
7.34.2.2	~StackItemType3	92
7.34.3	Member Function Documentation	92
7.34.3.1	addParent	92

7.34.3.2	clone	93
7.34.3.3	curr	93
7.34.3.4	i	93
7.34.3.5	len	93
7.34.3.6	nextTokenConstraints	93
7.34.3.7	parent	93
7.34.3.8	rule	93
7.34.3.9	rueno	93
7.34.3.10	scope	93
7.34.3.11	setParents	94
7.34.3.12	tree	94
7.34.3.13	type	94
7.35	DynGenPar::StackItemType4 Class Reference	94
7.35.1	Detailed Description	95
7.35.2	Constructor & Destructor Documentation	95
7.35.2.1	StackItemType4	95
7.35.2.2	~StackItemType4	95
7.35.3	Member Function Documentation	95
7.35.3.1	addParent	95
7.35.3.2	clone	95
7.35.3.3	len	96
7.35.3.4	parent	96
7.35.3.5	pos	96
7.35.3.6	setParents	96
7.35.3.7	target	96
7.35.3.8	type	96
7.36	DynGenPar::StackItemType5 Class Reference	97
7.36.1	Detailed Description	97
7.36.2	Constructor & Destructor Documentation	97

7.36.2.1	StackItemType5	97
7.36.2.2	~StackItemType5	98
7.36.3	Member Function Documentation	98
7.36.3.1	addParent	98
7.36.3.2	cat	98
7.36.3.3	clone	98
7.36.3.4	parent	98
7.36.3.5	scope	98
7.36.3.6	setParents	98
7.36.3.7	type	98
7.37	DynGenPar::StackItemType6 Class Reference	99
7.37.1	Detailed Description	99
7.37.2	Constructor & Destructor Documentation	100
7.37.2.1	StackItemType6	100
7.37.2.2	~StackItemType6	100
7.37.3	Member Function Documentation	100
7.37.3.1	addParent	100
7.37.3.2	clone	100
7.37.3.3	i	100
7.37.3.4	leaves	100
7.37.3.5	nextTokenConstraints	100
7.37.3.6	parent	100
7.37.3.7	scope	101
7.37.3.8	setParents	101
7.37.3.9	tree	101
7.37.3.10	type	101
7.38	DynGenPar::Term Struct Reference	101
7.38.1	Detailed Description	102
7.38.2	Constructor & Destructor Documentation	102

7.38.2.1	Term	102
7.38.2.2	Term	102
7.38.2.3	Term	102
7.38.3	Member Function Documentation	102
7.38.3.1	isComponent	102
7.38.3.2	isToken	102
7.38.3.3	operator==	102
7.38.4	Member Data Documentation	102
7.38.4.1	arg	102
7.38.4.2	component	103
7.38.4.3	token	103
7.39	DynGenPar::TextByteLexerStateData Class Reference	103
7.39.1	Detailed Description	104
7.39.2	Constructor & Destructor Documentation	104
7.39.2.1	TextByteLexerStateData	104
7.39.3	Member Function Documentation	104
7.39.3.1	clone	104
7.39.4	Member Data Documentation	104
7.39.4.1	streamPos	104
7.39.4.2	textPos	104
7.40	DynGenPar::TextByteTokenSource Class Reference	105
7.40.1	Detailed Description	105
7.40.2	Constructor & Destructor Documentation	106
7.40.2.1	TextByteTokenSource	106
7.40.2.2	TextByteTokenSource	106
7.40.2.3	~TextByteTokenSource	106
7.40.3	Member Function Documentation	106
7.40.3.1	readToken	106
7.40.3.2	reset	106

7.40.3.3	rewindTo	106
7.40.3.4	saveState	106
7.40.3.5	textPosition	107
7.41	DynGenPar::TextPosition Struct Reference	107
7.41.1	Detailed Description	107
7.41.2	Constructor & Destructor Documentation	107
7.41.2.1	TextPosition	107
7.41.2.2	TextPosition	108
7.41.3	Member Function Documentation	108
7.41.3.1	countCharacter	108
7.41.3.2	reset	108
7.41.4	Member Data Documentation	108
7.41.4.1	col	108
7.41.4.2	line	108
7.42	DynGenPar::TokenSource Class Reference	108
7.42.1	Detailed Description	109
7.42.2	Constructor & Destructor Documentation	109
7.42.2.1	TokenSource	109
7.42.2.2	~TokenSource	109
7.42.3	Member Function Documentation	110
7.42.3.1	currentPosition	110
7.42.3.2	matchParseTree	110
7.42.3.3	nextToken	110
7.42.3.4	parseTree	110
7.42.3.5	readToken	110
7.42.3.6	rewindTo	111
7.42.3.7	saveState	111
7.42.3.8	simpleRewind	111
7.42.4	Member Data Documentation	111

7.42.4.1	currPos	111
7.42.4.2	tree	112
7.43	DynGenPar::UnifiedStackItemData Class Reference	112
7.43.1	Detailed Description	113
7.43.2	Constructor & Destructor Documentation	113
7.43.2.1	UnifiedStackItemData	113
7.43.2.2	~UnifiedStackItemData	113
7.43.3	Member Function Documentation	113
7.43.3.1	derefUsage	113
7.43.3.2	refUsage	113
8	File Documentation	115
8.1	bytetokensource.h File Reference	115
8.1.1	Enumeration Type Documentation	115
8.1.1.1	ByteTokens	115
8.2	dyngenpar.cpp File Reference	116
8.2.1	Function Documentation	116
8.2.1.1	qHash	116
8.3	dyngenpar.h File Reference	116
8.3.1	Define Documentation	118
8.3.1.1	IS_EPSILON	118
8.3.2	Function Documentation	119
8.3.2.1	qHash	119
8.4	flexlexertokensource.h File Reference	119
8.5	pgf.cpp File Reference	119
8.5.1	Define Documentation	119
8.5.1.1	CHECK_STATUS	119
8.6	pgf.h File Reference	119
8.6.1	Define Documentation	120
8.6.1.1	STATIC	120

Chapter 1

Main Page

1.1 Introduction

DynGenPar is an innovative parser based on a new principle combining bottom-up and top-down features of traditional parsers. The most unique feature of the algorithm is the possibility to add rules to the grammar at almost any time, even during parsing. DynGenPar has the following characterizing properties:

Dynamic = The grammar is not hardcoded as in usual table-driven approaches, such as (Generalized) LR or Earley's algorithm. Instead, the algorithm works on a runtime representation of the grammar, which allows efficient handling of dynamic grammar changes. To decide when and how to shift or reduce, we use, instead of the usual LR tables, the initial graph, a graph which is easily updated as the grammar changes, along with some runtime top-down information.

Generalized = The algorithm exhaustively parses ambiguous grammars. In addition, epsilon productions are considered in order to support arbitrary CFGs. (Left recursion is naturally supported thanks to the bottom-up nature of the algorithm.) We use graph-structured (DAG-structured) stacks similar to the ones used in Tomita's Generalized LR algorithm. As additional generalizations, we also support Parallel Multiple Context-Free Grammars (PMCFGs) and next token constraints (useful, e.g. for scannerless parsing).

Due to the dynamic design, a parser generator is not needed. Instead, the parser can simply be used as a library.

DynGenPar supports dynamic grammar additions, incremental parsing, prediction, rule labels, custom parse actions, arbitrary token sources, hierarchical parsing, parallel multiple context-free grammars (PMCFGs), and next token constraints. See section [Features](#) for details.

1.2 Basic Usage

All the APIs provided by DynGenPar are in the [DynGenPar](#) namespace. You can use

```
using namespace DynGenPar;
```

to bring in the entire namespace.

The main class of DynGenPar is `DynGenPar::Parser`. To do any parsing, you will always have to instantiate at least one object of the `DynGenPar::Parser` class, or a subclass such as `DynGenPar::PgParser`. Parsing is done through the `DynGenPar::Parser::parse` methods, either the fine-grained version with several arguments or the binding-friendly version which operates on a `DynGenPar::ParseState` object grouping all the arguments.

The parser operates on a stream of tokens. Those tokens have to be provided by a token source, i.e. a class implementing the `DynGenPar::TokenSource` interface. DynGenPar provides several ready-made token sources. You can also implement your own: Your class only has to derive from `DynGenPar::TokenSource` and implement the needed virtual methods, at least the pure virtual `DynGenPar::TokenSource::readToken`. In the following example, we will use the simplest token source, which operates directly on a list of tokens: the `DynGenPar::ListTokenSource`.

This is a basic example for how to use `DynGenPar::Parser`:

```
DynGenPar::ListTokenSource tokenSource;
DynGenPar::Parser parser(&tokenSource);

// We have to specify the grammar.
// In this example, the grammar is hardcoded.
parser.tokens << "a" << "b" << "c";
parser.rules["S"] << (Rule() << "A")
                  << (Rule() << "A" << "S");
parser.rules["A"] << (Rule() << "a")
                  << (Rule() << "b")
                  << (Rule() << "c");
parser.startCat = "S";

// Whenever we set the grammar directly, we have to call this function.
parser.initCaches();

// And here, we specify the sample input, also hardcoded.
tokenSource.inputTokens << "b" << "a";

// Now we parse the input, using the default arguments for
// DynGenPar::Parser::parse.
QList<DynGenPar::Match> matches = parser.parse();
```

The `DynGenPar::Match` class contains the results of the parsing process; in particular, the parsed tree(s) in a packed, i.e. DAG (directed acyclic graph) -structured, representation, see the `DynGenPar::Match::tree` field.

The following example shows how to iterate over a parse tree:

```
static void printSubtree(const DynGenPar::Node &node, int level,
                       QTextStream &stream);

// prints the parse tree given as an argument to stdout
static void printParseTree(const DynGenPar::Node &tree) {
    QTextStream stream(stdout);
    printSubtree(tree, 0, stream);
}

// recursive helper function for the above, which does the actual work
static void printSubtree(const DynGenPar::Node &node, int level,
                       QTextStream &stream) {
    for (int i=0; i<level; i++) stream << ' ';
    stream << node.cat << endl;
    switch (node.children.size()) {
        case 0: // no alternatives = error node
```

```

    for (int i=0; i<=level; i++) stream << ' ';
    stream << "ERROR" << endl;
    break;
case 1: // 1 alternative, normal tree
    foreach (const DynGenPar::Node &child, node.children.first())
        printSubtree(child, level+1, stream);
    break;
default: // multiple alternatives
    foreach (const DynGenPar::Alternative &subtree, node.children) {
        for (int i=0; i<=level; i++) stream << ' ';
        stream << "ALTERNATIVE" << endl;
        foreach (const DynGenPar::Node &child, subtree)
            printSubtree(child, level+2, stream);
    }
    break;
}
}
}

```

You would call the above `printParseTree` function using e.g.

```
foreach (const DynGenPar::Match &m, matches) printParseTree(m.tree);
```

on the result of the previous example.

By default, category names, i.e. the `DynGenPar::Cat` typedef, are strings, which is convenient for debugging, but often not what is wanted in practice. For efficiency and/or interoperability, categories can be forced to be integers instead, by #defining `DYNGENPAR_INTEGER_CATEGORIES`. Some contexts such as the PGF support ([pgf.h](#)), the `DynGenPar::FlexLexerTokenSource` ([flexlexertokensource.h](#)) and the Java bindings always force this option.

1.3 Java Bindings

Java bindings for `DynGenPar` based on Qt Jambi are provided. Most of the `DynGenPar` API is available also to Java developers.

All the APIs provided by `DynGenPar` are in the `concise.DynGenPar` package. You can use

```
import concise.DynGenPar.*;
```

to import the entire package.

The following mappings and restrictions apply:

- The `DynGenPar` namespace is mapped to the `concise.DynGenPar` package.
- The global functions in the `DynGenPar` namespace are mapped to static methods of a `concise.DynGenPar.Util` class.
- The global constants and enumerations in the `DynGenPar` namespace are mapped as follows:
 - `DynGenPar::PgfToken*` to `concise.DynGenPar.Pgf.Token*`
 - `DynGenPar::PreludeBind` to `concise.DynGenPar.Pgf.PreludeBind`
 - `DynGenPar::ByteToken*` to `concise.DynGenPar.ByteTokenSource.*`

- `DYNGENPAR_INTEGER_CATEGORIES` is always defined, i.e. category identifiers are always integers, not strings. See `DynGenPar::Cat`.
- The internals of parse stacks cannot be accessed through Java. Parse stacks are only exposed in the form of opaque parse states (`concise.DynGenPar.ParseState`, i.e. `DynGenPar::ParseState`). The `DynGenPar::ParseState::incrementalStacks` member cannot be accessed from Java. In particular, only the overloads of `DynGenPar::Parser::parse` and of the `Prediction` methods in `DynGenPar::Parser` which take a `DynGenPar::ParseState` are available.
- The in-place `DynGenPar::PgfParser::filterCoercionsFromSyntaxTree(DynGenPar::Node &) const` method is not supported, but the `DynGenPar::PgfParser::filterCoercionsFromSyntaxTree(const DynGenPar::Node &) const` method which returns a `DynGenPar::Node` is available.
- Some other methods are currently not available due to technical limitations or bugs in the version of Qt Jambi we are currently using.
- Public class/structure member fields such as `DynGenPar::Parser::tokens` are mapped to getters, e.g. `concise.DynGenPar.Parser.tokens`, and setters, e.g. `concise.DynGenPar.Parser.setTokens`. This is because Java only supports public member fields for Java code, it does not support native fields nor a property mechanism that would allow hiding the required function calls to access the C++ class's member field.
- Qt lists, hash tables etc. are mapped to their Java equivalents.
- Multi-valued hash tables are not directly supported in Java, therefore they are mapped to list-valued hash tables.
- List classes (classes derived from `QList`) can be appended to using a Java-style `add` method, and converted to an iterable Java list using a `toList` method. (In C++, the `toList` method is just an alias for a trivial cast and returns a writable reference to a `QList`. In Java, due to the required conversion, the returned list can be used for reading only.)
- Java does not support typedef, so unfortunately you have to spell out the actual types of things like `DynGenPar::RuleSet`.

This documentation documents the C++ API. With the exception of the above changes, the same interfaces can also be used from Java. In particular, it is transparently possible to implement virtual methods of C++ classes in derived Java classes, and thus interfaces required by the C++ code can also be implemented in Java.

In Java, the basic example for how to use `DynGenPar::Parser` would look as follows:

```
concise.DynGenPar.ListTokenSource tokenSource
    = new concise.DynGenPar.ListTokenSource();
concise.DynGenPar.Parser parser = new concise.DynGenPar.Parser(tokenSource);

// We have to specify the grammar.
// In this example, the grammar is hardcoded.
// Category identifiers must be integers, but char promotes to int.
// Unfortunately, the syntax is not quite as pretty as the C++ one.
parser.addToken('a');
parser.addToken('b');
parser.addToken('c');
java.util.HashMap<Integer, java.util.List<concise.DynGenPar.Rule> > rules
    = new java.util.HashMap<Integer,
        java.util.List<concise.DynGenPar.Rule> > ();
java.util.List<concise.DynGenPar.Rule> sRules
    = new java.util.ArrayList<concise.DynGenPar.Rule> ();
concise.DynGenPar.Rule rule1 = new concise.DynGenPar.Rule();
rule1.add('A');
sRules.add(rule1);
```

```

concise.DynGenPar.Rule rule2 = new concise.DynGenPar.Rule();
rule2.add('A');
rule2.add('S');
sRules.add(rule2);
rules.put('S', sRules);
java.util.List<concise.DynGenPar.Rule> aRules
    = new java.util.ArrayList<concise.DynGenPar.Rule>();
concise.DynGenPar.Rule rule3 = new concise.DynGenPar.Rule();
rule3.add('a');
aRules.add(rule3);
concise.DynGenPar.Rule rule4 = new concise.DynGenPar.Rule();
rule4.add('b');
aRules.add(rule4);
concise.DynGenPar.Rule rule5 = new concise.DynGenPar.Rule();
rule5.add('c');
aRules.add(rule5);
rules.put('A', aRules);
parser.setRules(rules);
parser.setStartCat('S');

// Whenever we set the grammar directly, we have to call this function.
parser.initCaches();

// And here, we specify the sample input, also hardcoded.
java.util.List<Integer> inputTokens = new java.util.ArrayList<Integer>();
inputTokens.add('b');
inputTokens.add('a');
tokenSource.setInputTokens(inputTokens);

// Now we parse the input, using a default parse state.
concise.DynGenPar.ParseState parseState
    = new concise.DynGenPar.ParseState();
java.util.List<concise.DynGenPar.Match> matches = parser.parse(parseState);

```

The DynGenPar Java bindings are used in the [FMathL Concise](#) environment.

[<http://www.mat.univie.ac.at/~neum/FMathL.html#Concise>]

1.4 Features

This section lists the advanced features supported by DynGenPar.

1.4.1 Dynamic Grammar Additions

The most unique feature of DynGenPar is the possibility to add rules to the grammar at almost any time, even during parsing. See [DynGenPar::Parser::addRule](#).

1.4.2 Incremental Parsing

DynGenPar allows operating on its input incrementally, parsing interactively as input is produced and remembering its state. See [DynGenPar::Parser::parse](#) and [DynGenPar::ParseState](#).

It is also possible to go back by resuming at a previous saved state.

1.4.3 Prediction

Possible continuations, i.e. tokens which can come next, can be predicted at any saved [incremental parsing](#) state, see [DynGenPar::Parser::computePredictions](#) and [DynGenPar::Parser::computeConstrainedPredictions](#).

It is also possible to predict entire "literals", which are sequences of tokens appearing sequentially within a rule, see [DynGenPar::Parser::computeMultiPredictions](#) and [DynGenPar::Parser::computeConstrainedMultiPredictions](#). This is particularly useful for scannerless parsing.

1.4.4 Rule Labels

CFG rules can have labels, which are reproduced in the produced parse tree. Those labels can be anything which can be stored in a QVariant, including strings, integers and pointers to arbitrary objects. See [DynGenPar::Rule::Rule](#), [DynGenPar::Rule::label](#) and [DynGenPar::Rule::setLabel](#).

1.4.5 Custom Parse Actions

It is also possible to attach an action to a rule, which will be executed when the rule is matched. Note that actions will currently only be executed for nonempty matches (and in particular, actions on epsilon rules will always be ignored) and that an action will be executed even if the detected match only appears in some of the considered parses and is later discarded due to the input that follows. An action must be a subclass of [DynGenPar::Action](#) implementing the pure virtual [DynGenPar::Action::execute](#) method. See [DynGenPar::Action](#) and [DynGenPar::Rule::action](#).

1.4.6 Arbitrary Token Sources

DynGenPar accepts tokens from any source implementing the [DynGenPar::TokenSource](#) interface. Your class only has to derive from [DynGenPar::TokenSource](#) and implement the needed virtual methods, at least the pure virtual [DynGenPar::TokenSource::readToken](#).

A number of ready-made token sources are also provided. See [DynGenPar::ListTokenSource](#), [DynGenPar::ByteTokenSource](#) and [DynGenPar::TextByteTokenSource](#).

1.4.6.1 Flex Lexer Token Source

In particular, it is possible to use a lexer generated by Flex as a token source. See [DynGenPar::FlexLexerTokenSource](#).

1.4.7 Hierarchical Parsing

A token source can return, along with a token, a complete parse (sub)tree to use in lieu of a leaf node in the resulting parse tree, making it possible to call, from your token source, another parser, or another instance of DynGenPar (which is fully reentrant), and to return the result as a single token for the higher-level grammar. See [DynGenPar::TokenSource::tree](#).

1.4.8 Parallel Multiple Context-Free Grammars (PMCFGs)

PMCFGs (Parallel Multiple Context-Free Grammars) are an extension of context-free grammars used for natural language, e.g. by the Grammatical Framework GF. They are natively supported by DynGenPar. See [DynGenPar::Pmcfg](#), [DynGenPar::Parser::loadPmcfg](#), [DynGenPar::Parser::addPmcfgRule](#) and [DynGenPar::parseTreeToPmcfgSyntaxTree](#).

1.4.8.1 Grammatical Framework (GF) PGF Grammars

In particular, DynGenPar can import compiled grammars produced by the GF compiler, in the PGF (Portable Grammatical Framework) file format, automatically converting them to PMCFGs in standard form and providing a GF-compatible lexer. See [DynGenPar::Pgf](#) and [DynGenPar::PgfParser](#).

1.4.9 Next Token Constraints

Rules can have constraints attached that require (expect) or forbid (taboo) certain tokens following the rule. This is particularly useful for scannerless parsing, where it allows one to implement the usual context-sensitive scannerless parsing primitives such as maximal matches. It can also be used to enforce grammatically correct word sequences, e.g. for the singular indefinite article ("a" vs. "an") in English. See [DynGenPar::NextTokenConstraints](#) and [DynGenPar::Rule::nextTokenConstraints](#).

1.5 TODO

The following desirable features are currently not supported:

- some PMCFGs with infinitely ambiguous context-free approximations
- general context-sensitive constraints on rules (except PMCFG, next token)
- triggering parse actions on empty matches (including epsilon rules)
- error correction (we only have basic error detection)

(These features may or may not turn out to be required in practice.)

It is planned to add support for these features to the algorithm as needed, without changing the basic structure.

1.6 Copyright

DynGenPar: Dynamic Generalized Parser

Copyright (C) 2010-2012 Kevin Kofler <kevin.kofler@chello.at>

1.6.1 Acknowledgments

Support by the Austrian Science Fund FWF under contract numbers P20631 and P23554 is gratefully acknowledged.

1.6.2 Licensing and Warranty Disclaimer

This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program. If not, see <http://www.gnu.org/licenses/>.

Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

DynGenPar	17
---------------------------------	----

Chapter 3

Class Index

3.1 Class Hierarchy

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

DynGenPar::Action	24
DynGenPar::ActionInfo	25
DynGenPar::Cfg	31
DynGenPar::ConstrainedMultiPrediction	32
DynGenPar::FullRule	36
DynGenPar::LexerState	39
DynGenPar::Match	42
DynGenPar::MultiPrediction	44
DynGenPar::NextTokenConstraints	45
DynGenPar::Node	47
DynGenPar::Parser	48
DynGenPar::PgfParser	65
DynGenPar::ParseState	61
DynGenPar::Pgf	63
DynGenPar::Pmcfg	68
DynGenPar::PmcfgComponentInfo	70
DynGenPar::PseudoCatScope	71
QList	74
DynGenPar::Alternative	26
DynGenPar::Function	37
DynGenPar::Sequence	78
QSharedData	75
DynGenPar::AbstractLexerStateData	23
DynGenPar::TextByteLexerStateData	103
DynGenPar::PseudoCatScopeData	73
DynGenPar::StackItemData	83
DynGenPar::StackItemType0	84
DynGenPar::StackItemType1	87
DynGenPar::StackItemType3	91

DynGenPar::StackItemType5	97
DynGenPar::StackItemType6	99
DynGenPar::UnifiedStackItemData	112
DynGenPar::StackItemType2	89
DynGenPar::StackItemType4	94
DynGenPar::Rule	75
DynGenPar::StackItem	80
DynGenPar::Term	101
DynGenPar::TextPosition	107
DynGenPar::TokenSource	108
DynGenPar::ByteTokenSource	28
DynGenPar::TextByteTokenSource	105
DynGenPar::FlexLexerTokenSource	34
DynGenPar::ListTokenSource	40

Chapter 4

Class Index

4.1 Class List

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

DynGenPar::AbstractLexerStateData (API for stateful lexers to save their state for rewinding)	23
DynGenPar::Action (Interface for parser actions)	24
DynGenPar::ActionInfo (Data passed to parser actions)	25
DynGenPar::Alternative	26
DynGenPar::ByteTokenSource	28
DynGenPar::Cfg (An object representing a CFG (or a PMCFG in our internal representation))	31
DynGenPar::ConstrainedMultiPrediction (Multi-token predictions with next token constraints)	32
DynGenPar::FlexLexerTokenSource	34
DynGenPar::FullRule (Full rule as found in the initial graph)	36
DynGenPar::Function (PMCFG function)	37
DynGenPar::LexerState	39
DynGenPar::ListTokenSource	40
DynGenPar::Match ((possibly partial) match)	42
DynGenPar::MultiPrediction (Multi-token predictions)	44
DynGenPar::NextTokenConstraints (Rule constraints affecting the next token, for scannerless parsing)	45
DynGenPar::Node (Node in the parse tree)	47
DynGenPar::Parser (Main class)	48
DynGenPar::ParseState (Parse state struct, for bindings)	61
DynGenPar::Pgf (Representation of the information in .pgf files in a format we can process)	63
DynGenPar::PgfParser (Parser for PGF grammars)	65
DynGenPar::Pmcf (PMCFG)	68
DynGenPar::PmcfComponentInfo (Attached to the parse trees as rule labels to allow obtaining syntax trees)	70
DynGenPar::PseudoCatScope	71
DynGenPar::PseudoCatScopeData	73
QList	74
QSharedData	75
DynGenPar::Rule	75
DynGenPar::Sequence (Component of a PMCFG function, a sequence of terms)	78
DynGenPar::StackItem	80
DynGenPar::StackItemData	83

DynGenPar::StackItemType0 (Type 0 item: during match, we're waiting for a token to shift)	84
DynGenPar::StackItemType1 (Type 1 item: during type 0 item processing, we're executing a reduce)	87
DynGenPar::StackItemType2 (Type 2 item: during reduce, we're recursively executing another reduce)	89
DynGenPar::StackItemType3 (Type 3 item: during matchRemaining, we're executing a match)	91
DynGenPar::StackItemType4 (Type 4 item: during reduce, we're executing a matchRemaining)	94
DynGenPar::StackItemType5 (Type 5 item: during match (of an MCFG constraint), we're executing a matchRemaining)	97
DynGenPar::StackItemType6 (Type 6 item: during match, we're matching a P constraint)	99
DynGenPar::Term (<i>Term</i> in the expression of a component of a PMCFG function)	101
DynGenPar::TextByteLexerStateData (You should not have to use this class directly, ever)	103
DynGenPar::TextByteTokenSource	105
DynGenPar::TextPosition (Text position)	107
DynGenPar::TokenSource	108
DynGenPar::UnifiedStackItemData	112

Chapter 5

File Index

5.1 File List

Here is a list of all files with brief descriptions:

bytetokensource.h	115
dyngenpar.cpp	116
dyngenpar.h	116
flexlexertokensource.h	119
pgf.cpp	119
pgf.h	119

Chapter 6

Namespace Documentation

6.1 DynGenPar Namespace Reference

Classes

- class [ByteTokenSource](#)
- class [TextByteLexerStateData](#)
You should not have to use this class directly, ever.
- class [TextByteTokenSource](#)
- struct [NextTokenConstraints](#)
rule constraints affecting the next token, for scannerless parsing
- class [Rule](#)
- struct [Cfg](#)
An object representing a CFG (or a PMCFG in our internal representation)
- struct [MultiPrediction](#)
multi-token predictions
- struct [ConstrainedMultiPrediction](#)
multi-token predictions with next token constraints
- struct [FullRule](#)
full rule as found in the initial graph
- class [Alternative](#)
- struct [Node](#)
node in the parse tree
- class [PseudoCatScopeData](#)
- class [PseudoCatScope](#)
- struct [Match](#)
(possibly partial) match
- struct [ActionInfo](#)
data passed to parser actions
- class [Action](#)
interface for parser actions

- class [StackItemData](#)
- class [UnifiedStackItemData](#)
- class [StackItem](#)
- class [StackItemType0](#)
 - type 0 item: during match, we're waiting for a token to shift*
- class [StackItemType1](#)
 - type 1 item: during type 0 item processing, we're executing a reduce*
- class [StackItemType2](#)
 - type 2 item: during reduce, we're recursively executing another reduce*
- class [StackItemType3](#)
 - type 3 item: during matchRemaining, we're executing a match*
- class [StackItemType4](#)
 - type 4 item: during reduce, we're executing a matchRemaining*
- class [StackItemType5](#)
 - type 5 item: during match (of an MCFG constraint), we're executing a matchRemaining*
- class [StackItemType6](#)
 - type 6 item: during match, we're matching a P constraint*
- class [AbstractLexerStateData](#)
 - API for stateful lexers to save their state for rewinding.*
- class [LexerState](#)
- class [TokenSource](#)
- class [ListTokenSource](#)
- struct [TextPosition](#)
 - text position*
- struct [Term](#)
 - term in the expression of a component of a PMCFG function*
- class [Sequence](#)
 - component of a PMCFG function, a sequence of terms*
- class [Function](#)
 - PMCFG function.*
- struct [Pmcfg](#)
 - PMCFG.*
- struct [PmcfgComponentInfo](#)
 - attached to the parse trees as rule labels to allow obtaining syntax trees*
- struct [ParseState](#)
 - parse state struct, for bindings*
- class [Parser](#)
 - main class*
- class [FlexLexerTokenSource](#)
- struct [Pgf](#)
 - representation of the information in .pgf files in a format we can process*
- class [PgfParser](#)
 - parser for PGF grammars*

Typedefs

- typedef QString [Cat](#)
Category type: string or integer depending on DYNGENPAR_INTEGER_CATEGORIES.
- typedef const [Cat](#) & [CatArg](#)
Category type (string or integer) when passed as an argument.
- typedef QHash< [Cat](#), [QList](#)< [Rule](#) > > [RuleSet](#)
- typedef QSet< [Cat](#) > [TokenSet](#)
- typedef QSet< [Cat](#) > [Predictions](#)
- typedef QMultiHash< [QList](#)< [Cat](#) >, [MultiPrediction](#) > [MultiPredictions](#)
- typedef QMultiHash< [Cat](#), [NextTokenConstraints](#) > [ConstrainedPredictions](#)
- typedef QMultiHash< [QList](#)< [Cat](#) >, [ConstrainedMultiPrediction](#) > [ConstrainedMultiPredictions](#)

Enumerations

- enum [PgfReservedTokens](#) {
[PgfTokenEpsilon](#), [PgfTokenLexError](#), [PgfTokenVar](#), [PgfTokenFloat](#),
[PgfTokenInt](#), [PgfTokenString](#) }

Functions

- uint [qHash](#) (const [NextTokenConstraints](#) &nextTokenConstraints)
simple hash function for next token constraints
- [Node](#) [parseTreeToPmcfgyntaxTree](#) (const [Node](#) &parseTree)
converts a parse tree obtained from a PMCFG to a PMCFG syntax tree
- template<class [MultiHashType](#) >
[QHash](#)< typename [MultiHashType](#)::key_type, [QList](#)< typename [MultiHashType](#)::mapped_type > > [multi-HashToListHash](#) (const [MultiHashType](#) &multiHash)
mapping from QMultiHash to QHash with list value for the Java bindings
- uint [qHash](#) (const [PseudoCatScope](#) &scope)

Variables

- STATIC const char *const [PreludeBind](#) = "&+"

6.1.1 Typedef Documentation

6.1.1.1 typedef QString DynGenPar::Cat

Category type: string or integer depending on DYNGENPAR_INTEGER_CATEGORIES.

If DYNGENPAR_INTEGER_CATEGORIES is defined, this typedef is defined as:

```
typedef int Cat;
```

instead.

Definition at line 57 of file dyngenpar.h.

6.1.1.2 `typedef const Cat& DynGenPar::CatArg`

Category type (string or integer) when passed as an argument.

If `DYNGENPAR_INTEGER_CATEGORIES` is defined, this typedef is defined as:

```
typedef int CatArg;
```

instead.

This typedef is used for maximum efficiency, to pass strings by reference, but integers by value.

Definition at line 69 of file `dyngenpar.h`.

6.1.1.3 `typedef QMultiHash<QList<Cat>, ConstrainedMultiPrediction> DynGenPar::ConstrainedMultiPredictions`

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

6.1.1.4 `typedef QMultiHash<Cat, NextTokenConstraints> DynGenPar::ConstrainedPredictions`

Definition at line 192 of file `dyngenpar.h`.

6.1.1.5 `typedef QMultiHash<QList<Cat>, MultiPrediction> DynGenPar::MultiPredictions`

Definition at line 191 of file `dyngenpar.h`.

6.1.1.6 `typedef QSet<Cat> DynGenPar::Predictions`

Definition at line 175 of file `dyngenpar.h`.

6.1.1.7 `typedef QHash<Cat, QList<Rule>> DynGenPar::RuleSet`

Definition at line 156 of file `dyngenpar.h`.

6.1.1.8 `typedef QSet<Cat> DynGenPar::TokenSet`

Definition at line 157 of file `dyngenpar.h`.

6.1.2 Enumeration Type Documentation

6.1.2.1 enum DynGenPar::PgfReservedTokens

Enumerator:

PgfTokenEpsilon

PgfTokenLexError

PgfTokenVar

PgfTokenFloat

PgfTokenInt

PgfTokenString

Definition at line 39 of file pgf.h.

6.1.3 Function Documentation

6.1.3.1 template<class MultiHashType > QHash<typename MultiHashType::key_type, QList<typename MultiHashType::mapped_type> > DynGenPar::multiHashToListHash (const MultiHashType & multiHash)

mapping from QMultiHash to QHash with list value for the Java bindings

Definition at line 76 of file dyngenpar.h.

6.1.3.2 Node DynGenPar::parseTreeToPmcfgSyntaxTree (const Node & parseTree)

converts a parse tree obtained from a PMCFG to a PMCFG syntax tree

Definition at line 528 of file dyngenpar.cpp.

6.1.3.3 uint DynGenPar::qHash (const NextTokenConstraints & nextTokenConstraints)

simple hash function for next token constraints

Definition at line 441 of file dyngenpar.cpp.

6.1.3.4 uint DynGenPar::qHash (const PseudoCatScope & scope) [inline]

Definition at line 342 of file dyngenpar.h.

6.1.4 Variable Documentation

6.1.4.1 STATIC const char* const DynGenPar::PreludeBind = "&+"

Definition at line 45 of file pgf.h.

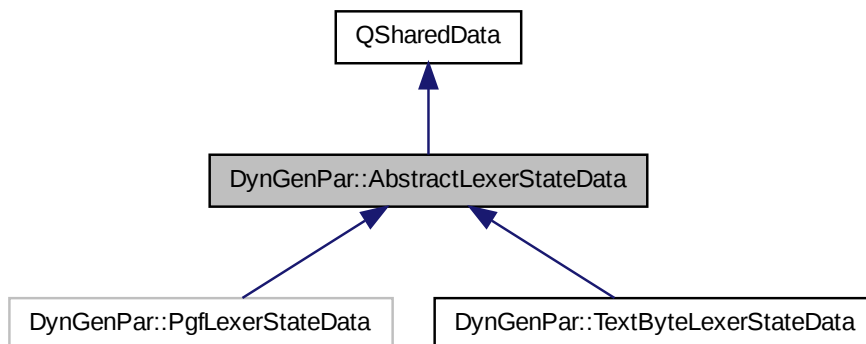
Chapter 7

Class Documentation

7.1 DynGenPar::AbstractLexerStateData Class Reference

API for stateful lexers to save their state for rewinding.

Inheritance diagram for DynGenPar::AbstractLexerStateData:



Public Member Functions

- virtual `~AbstractLexerStateData ()`
- virtual `AbstractLexerStateData * clone ()=0`

7.1.1 Detailed Description

API for stateful lexers to save their state for rewinding.

Definition at line 678 of file `dyngenpar.h`.

7.1.2 Constructor & Destructor Documentation

7.1.2.1 virtual DynGenPar::AbstractLexerStateData::~~AbstractLexerStateData () [inline, virtual]

Definition at line 680 of file dyngenpar.h.

7.1.3 Member Function Documentation

7.1.3.1 virtual AbstractLexerStateData* DynGenPar::AbstractLexerStateData::clone () [pure virtual]

Implemented in [DynGenPar::TextByteLexerStateData](#).

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

- [dyngenpar.h](#)

7.2 DynGenPar::Action Class Reference

interface for parser actions

Public Member Functions

- virtual [~Action](#) ()
- virtual void [execute](#) (const [ActionInfo](#) &info)=0

7.2.1 Detailed Description

interface for parser actions

Definition at line 371 of file dyngenpar.h.

7.2.2 Constructor & Destructor Documentation

7.2.2.1 virtual DynGenPar::Action::~~Action () [inline, virtual]

Definition at line 373 of file dyngenpar.h.

7.2.3 Member Function Documentation

7.2.3.1 virtual void DynGenPar::Action::execute (const [ActionInfo](#) & *info*) [pure virtual]

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

- [dyngenpar.h](#)

7.3 DynGenPar::ActionInfo Struct Reference

data passed to parser actions

Public Member Functions

- [ActionInfo](#) ()
dummy default constructor for bindings
- [ActionInfo](#) (const [Node](#) &t)

Public Attributes

- [Node tree](#)

7.3.1 Detailed Description

data passed to parser actions

Definition at line 363 of file dyngenpar.h.

7.3.2 Constructor & Destructor Documentation

7.3.2.1 DynGenPar::ActionInfo::ActionInfo () [[inline](#)]

dummy default constructor for bindings

Definition at line 365 of file dyngenpar.h.

7.3.2.2 DynGenPar::ActionInfo::ActionInfo (const **Node** & t) [[inline](#)]

Definition at line 366 of file dyngenpar.h.

7.3.3 Member Data Documentation

7.3.3.1 Node DynGenPar::ActionInfo::tree

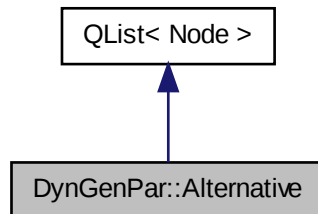
Definition at line 367 of file dyngenpar.h.

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

- [dyngenpar.h](#)

7.4 DynGenPar::Alternative Class Reference

Inheritance diagram for DynGenPar::Alternative:



Public Member Functions

- [Alternative](#) ()
- [Alternative](#) (const QVariant &label)
- [Alternative](#) (const QList< DynGenPar::Node > &list)
- [Alternative](#) (const QList< DynGenPar::Node > &list, const QVariant &label)
- QVariant [label](#) () const
- void [setLabel](#) (const QVariant &label)
- [Alternative](#) & [operator+=](#) (const QList< DynGenPar::Node > &other)
- [Alternative](#) & [operator+=](#) (const DynGenPar::Node &value)
- [Alternative](#) & [operator<<](#) (const QList< DynGenPar::Node > &other)
- [Alternative](#) & [operator<<](#) (const DynGenPar::Node &value)
- void [add](#) (const DynGenPar::Node &value)

Java-style + the binding generator doesn't detect the inherited append.

- QList< DynGenPar::Node > & [toList](#) ()

for bindings

7.4.1 Detailed Description

Definition at line 234 of file dyngenpar.h.

7.4.2 Constructor & Destructor Documentation

7.4.2.1 DynGenPar::Alternative::Alternative () [inline]

Definition at line 236 of file dyngenpar.h.

7.4.2.2 `DynGenPar::Alternative::Alternative (const QVariant & label) [inline, explicit]`

Definition at line 237 of file `dyngenpar.h`.

7.4.2.3 `DynGenPar::Alternative::Alternative (const QList< DynGenPar::Node > & list) [inline, explicit]`

Definition at line 239 of file `dyngenpar.h`.

7.4.2.4 `DynGenPar::Alternative::Alternative (const QList< DynGenPar::Node > & list, const QVariant & label) [inline]`

Definition at line 241 of file `dyngenpar.h`.

7.4.3 Member Function Documentation

7.4.3.1 `void DynGenPar::Alternative::add (const DynGenPar::Node & value) [inline]`

Java-style + the binding generator doesn't detect the inherited append.

Definition at line 262 of file `dyngenpar.h`.

7.4.3.2 `QVariant DynGenPar::Alternative::label () const [inline]`

Definition at line 243 of file `dyngenpar.h`.

7.4.3.3 `Alternative& DynGenPar::Alternative::operator+=(const DynGenPar::Node & value) [inline]`

Definition at line 249 of file `dyngenpar.h`.

7.4.3.4 `Alternative& DynGenPar::Alternative::operator+=(const QList< DynGenPar::Node > & other) [inline]`

Definition at line 245 of file `dyngenpar.h`.

7.4.3.5 `Alternative& DynGenPar::Alternative::operator<< (const QList< DynGenPar::Node > & other) [inline]`

Definition at line 253 of file `dyngenpar.h`.

7.4.3.6 `Alternative& DynGenPar::Alternative::operator<< (const DynGenPar::Node & value) [inline]`

Definition at line 257 of file `dyngenpar.h`.

7.4.3.7 `void DynGenPar::Alternative::setLabel (const QVariant & label) [inline]`

Definition at line 244 of file dyngenpar.h.

7.4.3.8 `QList<DynGenPar::Node>& DynGenPar::Alternative::toList () [inline]`

for bindings

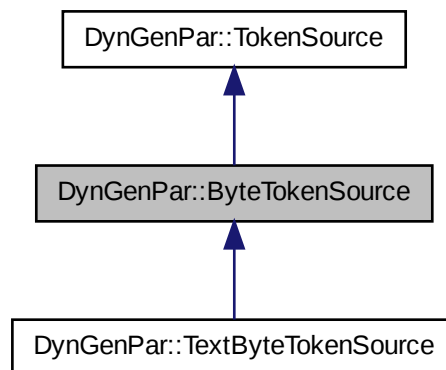
Definition at line 266 of file dyngenpar.h.

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

- [dyngenpar.h](#)

7.5 DynGenPar::ByteTokenSource Class Reference

Inheritance diagram for DynGenPar::ByteTokenSource:



Public Member Functions

- [ByteTokenSource](#) ()
- [ByteTokenSource](#) (const QString &fileName)
- virtual [~ByteTokenSource](#) ()
- virtual bool [rewindTo](#) (int pos, const [LexerState](#) &=[LexerState](#)())
rewind to an older position (requires buffering)
- void [setInputStdin](#) ()
- void [setInputFile](#) (const QString &fileName)
- void [setInputBytes](#) (const QByteArray &bytes)
- void [setInputString](#) (const QString &string)
- void [setInputBuffer](#) (QByteArray *buffer)

Protected Member Functions

- virtual [Cat readToken \(\)](#)
get the next token from the input, to be implemented by subclasses
- virtual void [reset \(\)](#)

Protected Attributes

- QIODevice * [stream](#)

7.5.1 Detailed Description

Definition at line 41 of file `bytetokensource.h`.

7.5.2 Constructor & Destructor Documentation

7.5.2.1 `DynGenPar::ByteTokenSource::ByteTokenSource ()` `[inline]`

Definition at line 43 of file `bytetokensource.h`.

7.5.2.2 `DynGenPar::ByteTokenSource::ByteTokenSource (const QString & fileName)` `[inline]`

Definition at line 46 of file `bytetokensource.h`.

7.5.2.3 `virtual DynGenPar::ByteTokenSource::~~ByteTokenSource ()` `[inline, virtual]`

Definition at line 50 of file `bytetokensource.h`.

7.5.3 Member Function Documentation

7.5.3.1 `virtual Cat DynGenPar::ByteTokenSource::readToken ()` `[inline, protected, virtual]`

get the next token from the input, to be implemented by subclasses

Implements [DynGenPar::TokenSource](#).

Reimplemented in [DynGenPar::TextByteTokenSource](#).

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

7.5.3.2 `virtual void DynGenPar::ByteTokenSource::reset ()` [inline, protected, virtual]

Reimplemented in [DynGenPar::TextByteTokenSource](#).

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

7.5.3.3 `virtual bool DynGenPar::ByteTokenSource::rewindTo (int pos, const LexerState & = LexerState ())` [inline, virtual]

rewind to an older position (requires buffering)

Returns

`true` if successful, `false` otherwise

in all cases, destroys the saved parse tree

By default, only succeeds if the position is the current one, otherwise always returns `false`. Can be overridden by subclasses.

Reimplemented from [DynGenPar::TokenSource](#).

Reimplemented in [DynGenPar::TextByteTokenSource](#).

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

7.5.3.4 `void DynGenPar::ByteTokenSource::setInputBuffer (QByteArray * buffer)` [inline]

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

7.5.3.5 `void DynGenPar::ByteTokenSource::setInputBytes (const QByteArray & bytes)` [inline]

Definition at line 73 of file `bytetokensource.h`.

7.5.3.6 `void DynGenPar::ByteTokenSource::setInputFile (const QString & fileName)` [inline]

Definition at line 67 of file `bytetokensource.h`.

7.5.3.7 `void DynGenPar::ByteTokenSource::setInputStdin ()` [inline]

Definition at line 61 of file `bytetokensource.h`.

7.5.3.8 `void DynGenPar::ByteTokenSource::setInputString (const QString & string)` [inline]

Definition at line 80 of file `bytetokensource.h`.

7.5.4 Member Data Documentation

7.5.4.1 QIODevice* DynGenPar::ByteTokenSource::stream [protected]

Definition at line 96 of file bytetokensource.h.

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

- [bytetokensource.h](#)

7.6 DynGenPar::Cfg Struct Reference

An object representing a CFG (or a PMCFG in our internal representation)

Public Member Functions

- [Cfg](#) ()
dummy default constructor for bindings
- [Cfg](#) (const [RuleSet](#) &r, const [TokenSet](#) &t, [CatArg](#) sc)
- bool [isToken](#) ([CatArg](#) cat) const
- void [addToken](#) ([CatArg](#) cat)

Public Attributes

- [RuleSet](#) rules
- [TokenSet](#) tokens
- [Cat](#) startCat

7.6.1 Detailed Description

An object representing a CFG (or a PMCFG in our internal representation)

This allows passing it around more easily and loading it into the parser in one step.

Definition at line 163 of file dyngenpar.h.

7.6.2 Constructor & Destructor Documentation

7.6.2.1 DynGenPar::Cfg::Cfg () [inline]

dummy default constructor for bindings

Definition at line 165 of file dyngenpar.h.

7.6.2.2 `DynGenPar::Cfg::Cfg (const RuleSet & r, const TokenSet & t, CatArg sc)` `[inline]`

Definition at line 166 of file `dyngenpar.h`.

7.6.3 Member Function Documentation

7.6.3.1 `void DynGenPar::Cfg::addToken (CatArg cat)` `[inline]`

Definition at line 172 of file `dyngenpar.h`.

7.6.3.2 `bool DynGenPar::Cfg::isToken (CatArg cat) const` `[inline]`

Definition at line 171 of file `dyngenpar.h`.

7.6.4 Member Data Documentation

7.6.4.1 `RuleSet DynGenPar::Cfg::rules`

Definition at line 168 of file `dyngenpar.h`.

7.6.4.2 `Cat DynGenPar::Cfg::startCat`

Definition at line 170 of file `dyngenpar.h`.

7.6.4.3 `TokenSet DynGenPar::Cfg::tokens`

Definition at line 169 of file `dyngenpar.h`.

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

- [dyngenpar.h](#)

7.7 DynGenPar::ConstrainedMultiPrediction Struct Reference

multi-token predictions with next token constraints

Public Member Functions

- [ConstrainedMultiPrediction](#) ()
dummy default constructor for bindings
- [ConstrainedMultiPrediction](#) (const [QList](#)< [Cat](#) > &fullLit, [CatArg](#) c)
- [ConstrainedMultiPrediction](#) (const [QList](#)< [Cat](#) > &fullLit, [CatArg](#) c, [NextTokenConstraints](#) ntc)
- bool [operator==](#) (const [ConstrainedMultiPrediction](#) &other) const
needed for [QList](#), [QMultiHash](#)

Public Attributes

- [QList](#)< [Cat](#) > fullLiteral
the entire literal completed by the prediction
- [Cat](#) cat
the nonterminal generating the literal / the nonterminal itself
- [NextTokenConstraints](#) nextTokenConstraints
only for nonterminals

7.7.1 Detailed Description

multi-token predictions with next token constraints

Definition at line 195 of file dyngenpar.h.

7.7.2 Constructor & Destructor Documentation

7.7.2.1 `DynGenPar::ConstrainedMultiPrediction::ConstrainedMultiPrediction ()` `[inline]`

dummy default constructor for bindings

Definition at line 197 of file dyngenpar.h.

7.7.2.2 `DynGenPar::ConstrainedMultiPrediction::ConstrainedMultiPrediction (const QList< Cat > & fullLit, CatArg c)` `[inline]`

Definition at line 199 of file dyngenpar.h.

7.7.2.3 `DynGenPar::ConstrainedMultiPrediction::ConstrainedMultiPrediction (const QList< Cat > & fullLit, CatArg c, NextTokenConstraints ntc)` `[inline]`

Definition at line 201 of file dyngenpar.h.

7.7.3 Member Function Documentation

7.7.3.1 `bool DynGenPar::ConstrainedMultiPrediction::operator==(const ConstrainedMultiPrediction & other) const`
`[inline]`

needed for [QList](#), [QMultiHash](#)

Definition at line 209 of file `dyngenpar.h`.

7.7.4 Member Data Documentation

7.7.4.1 `Cat DynGenPar::ConstrainedMultiPrediction::cat`

the nonterminal generating the literal / the nonterminal itself

Definition at line 205 of file `dyngenpar.h`.

7.7.4.2 `QList<Cat> DynGenPar::ConstrainedMultiPrediction::fullLiteral`

the entire literal completed by the prediction

Definition at line 204 of file `dyngenpar.h`.

7.7.4.3 `NextTokenConstraints DynGenPar::ConstrainedMultiPrediction::nextTokenConstraints`

only for nonterminals

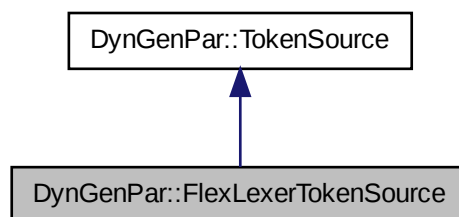
Definition at line 206 of file `dyngenpar.h`.

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

- [dyngenpar.h](#)

7.8 DynGenPar::FlexLexerTokenSource Class Reference

Inheritance diagram for `DynGenPar::FlexLexerTokenSource`:



Public Member Functions

- [FlexLexerTokenSource](#) (FlexLexer *lexer)
- virtual [~FlexLexerTokenSource](#) ()

Protected Member Functions

- virtual [Cat readToken](#) ()
get the next token from the input, to be implemented by subclasses

Protected Attributes

- FlexLexer * [flexLexer](#)

7.8.1 Detailed Description

Definition at line 29 of file flexlexertokensource.h.

7.8.2 Constructor & Destructor Documentation

7.8.2.1 `DynGenPar::FlexLexerTokenSource::FlexLexerTokenSource (FlexLexer * lexer)` [inline]

Definition at line 31 of file flexlexertokensource.h.

7.8.2.2 `virtual DynGenPar::FlexLexerTokenSource::~~FlexLexerTokenSource ()` [inline, virtual]

Definition at line 32 of file flexlexertokensource.h.

7.8.3 Member Function Documentation

7.8.3.1 `virtual Cat DynGenPar::FlexLexerTokenSource::readToken ()` [inline, protected, virtual]

get the next token from the input, to be implemented by subclasses

Implements [DynGenPar::TokenSource](#).

Definition at line 34 of file flexlexertokensource.h.

7.8.4 Member Data Documentation

7.8.4.1 FlexLexer* DynGenPar::FlexLexerTokenSource::flexLexer [protected]

Definition at line 37 of file flexlexertokensource.h.

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

- [flexlexertokensource.h](#)

7.9 DynGenPar::FullRule Struct Reference

full rule as found in the initial graph

Public Member Functions

- [FullRule](#) ()
dummy default constructor for bindings
- [FullRule](#) ([CatArg](#) c, const [Rule](#) &r, int epsSkipped, int n)

Public Attributes

- [Cat](#) cat
- [Rule](#) rule
- int [epsilonsSkipped](#)
- int [ruleno](#)
needed for PMCFGs (to match components of rules to each other)

7.9.1 Detailed Description

full rule as found in the initial graph

Definition at line 218 of file dyngenpar.h.

7.9.2 Constructor & Destructor Documentation

7.9.2.1 DynGenPar::FullRule::FullRule() [inline]

dummy default constructor for bindings

Definition at line 220 of file dyngenpar.h.

7.9.2.2 DynGenPar::FullRule::FullRule (CatArg c, const Rule & r, int epsSkipped, int n) [inline]

Definition at line 221 of file dyngenpar.h.

7.9.3 Member Data Documentation

7.9.3.1 Cat DynGenPar::FullRule::cat

Definition at line 223 of file dyngenpar.h.

7.9.3.2 int DynGenPar::FullRule::epsilonsSkipped

Definition at line 225 of file dyngenpar.h.

7.9.3.3 Rule DynGenPar::FullRule::rule

Definition at line 224 of file dyngenpar.h.

7.9.3.4 int DynGenPar::FullRule::ruleno

needed for PMCFGs (to match components of rules to each other)

always set to 0 for context-free (1-dimensional) categories

Definition at line 229 of file dyngenpar.h.

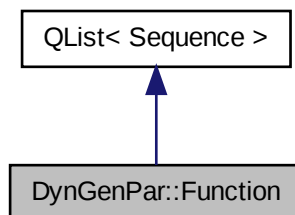
The documentation for this struct was generated from the following file:

- [dyngenpar.h](#)

7.10 DynGenPar::Function Class Reference

PMCFG function.

Inheritance diagram for DynGenPar::Function:



Public Member Functions

- [Function](#) ()
- [Function](#) (const [QList](#)< [Sequence](#) > &list)
- [Function](#) & [operator+=](#) (const [QList](#)< [Sequence](#) > &other)
- [Function](#) & [operator+=](#) (const [Sequence](#) &value)
- [Function](#) & [operator<<](#) (const [QList](#)< [Sequence](#) > &other)
- [Function](#) & [operator<<](#) (const [Sequence](#) &value)
- void [add](#) (const [Sequence](#) &value)
Java-style (for consistency, even though append is detected here)
- [QList](#)< [Sequence](#) > & [toList](#) ()
for bindings

7.10.1 Detailed Description

PMCFG function.

Definition at line 875 of file dyngenpar.h.

7.10.2 Constructor & Destructor Documentation

7.10.2.1 `DynGenPar::Function::Function ()` [`inline`]

Definition at line 877 of file dyngenpar.h.

7.10.2.2 `DynGenPar::Function::Function (const QList< Sequence > & list)` [`inline`, `explicit`]

Definition at line 878 of file dyngenpar.h.

7.10.3 Member Function Documentation

7.10.3.1 `void DynGenPar::Function::add (const Sequence & value)` [`inline`]

Java-style (for consistency, even though append is detected here)

Definition at line 896 of file dyngenpar.h.

7.10.3.2 `Function& DynGenPar::Function::operator+= (const QList< Sequence > & other)` [`inline`]

Definition at line 879 of file dyngenpar.h.

7.10.3.3 `Function& DynGenPar::Function::operator+=(const Sequence & value)` [inline]

Definition at line 883 of file `dyngenpar.h`.

7.10.3.4 `Function& DynGenPar::Function::operator<<(const Sequence & value)` [inline]

Definition at line 891 of file `dyngenpar.h`.

7.10.3.5 `Function& DynGenPar::Function::operator<<(const QList< Sequence > & other)` [inline]

Definition at line 887 of file `dyngenpar.h`.

7.10.3.6 `QList<Sequence>& DynGenPar::Function::toList()` [inline]

for bindings

Definition at line 900 of file `dyngenpar.h`.

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

- [dyngenpar.h](#)

7.11 DynGenPar::LexerState Class Reference

Public Member Functions

- [LexerState](#) ()
- [LexerState](#) ([AbstractLexerStateData](#) *data)
- void [clear](#) ()
- bool [isNull](#) () const
- const [AbstractLexerStateData](#) * [data](#) () const
- bool [operator==](#) (const [LexerState](#) &other) const

7.11.1 Detailed Description

Definition at line 684 of file `dyngenpar.h`.

7.11.2 Constructor & Destructor Documentation

7.11.2.1 `DynGenPar::LexerState::LexerState()` [inline]

Definition at line 686 of file `dyngenpar.h`.

7.11.2.2 DynGenPar::LexerState::LexerState (AbstractLexerStateData * data) [inline]

Definition at line 687 of file dyngenpar.h.

7.11.3 Member Function Documentation

7.11.3.1 void DynGenPar::LexerState::clear () [inline]

Definition at line 688 of file dyngenpar.h.

7.11.3.2 const AbstractLexerStateData* DynGenPar::LexerState::data () const [inline]

Definition at line 690 of file dyngenpar.h.

7.11.3.3 bool DynGenPar::LexerState::isNull () const [inline]

Definition at line 689 of file dyngenpar.h.

7.11.3.4 bool DynGenPar::LexerState::operator== (const LexerState & other) const [inline]

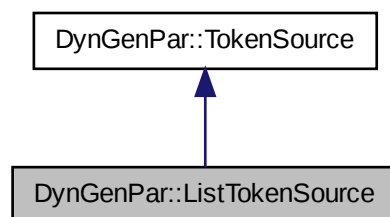
Definition at line 691 of file dyngenpar.h.

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

- [dyngenpar.h](#)

7.12 DynGenPar::ListTokenSource Class Reference

Inheritance diagram for DynGenPar::ListTokenSource:



Public Member Functions

- [ListTokenSource](#) ()
- virtual [~ListTokenSource](#) ()
- virtual bool [rewindTo](#) (int pos, const [LexerState](#) &=[LexerState](#)())
overridden because lists can be rewound

Public Attributes

- [QList< Cat >](#) inputTokens

Protected Member Functions

- virtual [Cat](#) [readToken](#) ()
just fetch the next token from the list

7.12.1 Detailed Description

Definition at line 770 of file `dyngenpar.h`.

7.12.2 Constructor & Destructor Documentation

7.12.2.1 `DynGenPar::ListTokenSource::ListTokenSource ()` [`inline`]

Definition at line 772 of file `dyngenpar.h`.

7.12.2.2 `virtual DynGenPar::ListTokenSource::~~ListTokenSource ()` [`inline`, `virtual`]

Definition at line 773 of file `dyngenpar.h`.

7.12.3 Member Function Documentation

7.12.3.1 `virtual Cat DynGenPar::ListTokenSource::readToken ()` [`inline`, `protected`, `virtual`]

just fetch the next token from the list

Implements [DynGenPar::TokenSource](#).

Definition at line 781 of file `dyngenpar.h`.

7.12.3.2 `virtual bool DynGenPar::ListTokenSource::rewindTo (int pos, const LexerState & = LexerState ()) [inline, virtual]`

overridden because lists can be rewound

Reimplemented from [DynGenPar::TokenSource](#).

Definition at line 776 of file `dyngenpar.h`.

7.12.4 Member Data Documentation

7.12.4.1 `QList<Cat> DynGenPar::ListTokenSource::inputTokens`

Definition at line 774 of file `dyngenpar.h`.

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

- [dyngenpar.h](#)

7.13 DynGenPar::Match Struct Reference

(possibly partial) match

Public Member Functions

- [Match \(\)](#)
dummy default constructor for bindings
- [Match \(int l, Node t, int n, PseudoCatScope s\)](#)
- [Match \(int l, Node t, int n, PseudoCatScope s, const NextTokenConstraints &nt\)](#)

Public Attributes

- `int len`
- `Node tree`
- `int ruleno`
used for PMCFGs
- `PseudoCatScope scope`
- `NextTokenConstraints nextTokenConstraints`

7.13.1 Detailed Description

(possibly partial) match

Definition at line 347 of file `dyngenpar.h`.

7.13.2 Constructor & Destructor Documentation

7.13.2.1 `DynGenPar::Match::Match()` `[inline]`

dummy default constructor for bindings

Definition at line 349 of file `dyngenpar.h`.

7.13.2.2 `DynGenPar::Match::Match(int l, Node t, int n, PseudoCatScope s)` `[inline]`

Definition at line 350 of file `dyngenpar.h`.

7.13.2.3 `DynGenPar::Match::Match(int l, Node t, int n, PseudoCatScope s, const NextTokenConstraints & nt)` `[inline]`

Definition at line 352 of file `dyngenpar.h`.

7.13.3 Member Data Documentation

7.13.3.1 `int DynGenPar::Match::len`

Definition at line 354 of file `dyngenpar.h`.

7.13.3.2 `NextTokenConstraints DynGenPar::Match::nextTokenConstraints`

Definition at line 359 of file `dyngenpar.h`.

7.13.3.3 `int DynGenPar::Match::ruleno`

used for PMCFGs

set to 0 where not needed to allow unification

Definition at line 356 of file `dyngenpar.h`.

7.13.3.4 `PseudoCatScope DynGenPar::Match::scope`

Definition at line 358 of file `dyngenpar.h`.

7.13.3.5 Node DynGenPar::Match::tree

Definition at line 355 of file dyngenpar.h.

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

- [dyngenpar.h](#)

7.14 DynGenPar::MultiPrediction Struct Reference

multi-token predictions

Public Member Functions

- [MultiPrediction](#) ()
dummy default constructor for bindings
- [MultiPrediction](#) (const [QList](#)< [Cat](#) > &fullLit, [CatArg](#) c)
- bool [operator==](#) (const [MultiPrediction](#) &other) const
needed for [QList](#), [QMultiHash](#)

Public Attributes

- [QList](#)< [Cat](#) > [fullLiteral](#)
the entire literal completed by the prediction
- [Cat](#) [cat](#)
the nonterminal generating the literal

7.14.1 Detailed Description

multi-token predictions

Definition at line 178 of file dyngenpar.h.

7.14.2 Constructor & Destructor Documentation

7.14.2.1 DynGenPar::MultiPrediction::MultiPrediction () [inline]

dummy default constructor for bindings

Definition at line 180 of file dyngenpar.h.

7.14.2.2 `DynGenPar::MultiPrediction::MultiPrediction (const QList< Cat > & fullLit, CatArg c) [inline]`

Definition at line 181 of file `dyngenpar.h`.

7.14.3 Member Function Documentation

7.14.3.1 `bool DynGenPar::MultiPrediction::operator==(const MultiPrediction & other) const [inline]`

needed for [QList](#), [QMultiHash](#)

Definition at line 187 of file `dyngenpar.h`.

7.14.4 Member Data Documentation

7.14.4.1 `Cat DynGenPar::MultiPrediction::cat`

the nonterminal generating the literal

Definition at line 184 of file `dyngenpar.h`.

7.14.4.2 `QList<Cat> DynGenPar::MultiPrediction::fullLiteral`

the entire literal completed by the prediction

Definition at line 183 of file `dyngenpar.h`.

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

- [dyngenpar.h](#)

7.15 `DynGenPar::NextTokenConstraints` Struct Reference

rule constraints affecting the next token, for scannerless parsing

Public Member Functions

- `bool operator==(const NextTokenConstraints &other) const`
needed for hash tables

Public Attributes

- [QList< Cat > expect](#)
list of context-free categories the next token MUST match
- [QList< Cat > taboo](#)
list of context-free categories the next token MUST NOT match

7.15.1 Detailed Description

rule constraints affecting the next token, for scannerless parsing

Definition at line 90 of file dyngenpar.h.

7.15.2 Member Function Documentation

7.15.2.1 `bool DynGenPar::NextTokenConstraints::operator==(const NextTokenConstraints & other) const` `[inline]`

needed for hash tables

Definition at line 108 of file dyngenpar.h.

7.15.3 Member Data Documentation

7.15.3.1 `QList<Cat> DynGenPar::NextTokenConstraints::expect`

list of context-free categories the next token MUST match

The categories in this list may be nonterminals or tokens. But they MUST be context-free. In other words, they must not be PMCFG pseudo-categories, and none of the rules used to derive them may contain any PMCFG pseudo-categories or next token constraints. (In particular, it is not possible to nest next token constraints.)

Definition at line 98 of file dyngenpar.h.

7.15.3.2 `QList<Cat> DynGenPar::NextTokenConstraints::taboo`

list of context-free categories the next token MUST NOT match

The categories in this list may be nonterminals or tokens. But they MUST be context-free. In other words, they must not be PMCFG pseudo-categories, and none of the rules used to derive them may contain any PMCFG pseudo-categories or next token constraints. (In particular, it is not possible to nest next token constraints.)

Definition at line 106 of file dyngenpar.h.

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

- [dyngenpar.h](#)

7.16 DynGenPar::Node Struct Reference

node in the parse tree

Public Member Functions

- [Node](#) ()
error node
- [Node](#) ([CatArg](#) c)
- [Node](#) ([CatArg](#) c, const [QVariant](#) &d)
- bool [operator==](#) (const [Node](#) &other) const
needed for [QList](#)

Public Attributes

- [Cat](#) cat
- [QVariant](#) data
- [QList](#)< [Alternative](#) > children

7.16.1 Detailed Description

node in the parse tree

Definition at line 275 of file [dyngenpar.h](#).

7.16.2 Constructor & Destructor Documentation

7.16.2.1 [DynGenPar::Node::Node](#) () [[inline](#)]

error node

Definition at line 276 of file [dyngenpar.h](#).

7.16.2.2 [DynGenPar::Node::Node](#) ([CatArg](#) c) [[inline](#)]

Definition at line 277 of file [dyngenpar.h](#).

7.16.2.3 [DynGenPar::Node::Node](#) ([CatArg](#) c, const [QVariant](#) & d) [[inline](#)]

Definition at line 278 of file [dyngenpar.h](#).

7.16.3 Member Function Documentation

7.16.3.1 `bool DynGenPar::Node::operator==(const Node & other) const` `[inline]`

needed for [QList](#)

Definition at line 285 of file `dyngenpar.h`.

7.16.4 Member Data Documentation

7.16.4.1 `Cat DynGenPar::Node::cat`

Definition at line 281 of file `dyngenpar.h`.

7.16.4.2 `QList<Alternative> DynGenPar::Node::children`

Definition at line 283 of file `dyngenpar.h`.

7.16.4.3 `QVariant DynGenPar::Node::data`

Definition at line 282 of file `dyngenpar.h`.

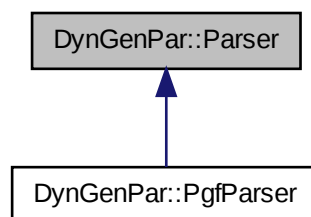
The documentation for this struct was generated from the following file:

- [dyngenpar.h](#)

7.17 DynGenPar::Parser Class Reference

main class

Inheritance diagram for `DynGenPar::Parser`:



Public Member Functions

- [Parser](#) ([TokenSource](#) *tokenSource)
- virtual [~Parser](#) ()
- bool [isToken](#) ([CatArg](#) cat) const
- void [addToken](#) ([CatArg](#) cat)
- bool [isLiteral](#) (const [QList](#)< [Cat](#) > &list) const
is a given list of categories a literal?
- void [initCaches](#) ()
clears all caches, then computes the nullable categories and the initial graph
- void [addRule](#) ([CatArg](#) cat, const [Rule](#) &rule)
adds a new rule to the grammar, updates the nullable categories and the initial graph and clears the other caches
- void [loadCfg](#) (const [Cfg](#) &cfg)
- bool [loadPmcf](#) (const [Pmcf](#) &pmcf)
loads a PMCFG in standard form, converting it to the internal representation
- bool [addPmcfRule](#) ([Pmcf](#) &pmcf, [CatArg](#) cat, const [Rule](#) &rule)
adds a new rule to the grammar (both the PMCFG and the internal representation), updates the nullable categories and the initial graph and clears the other caches
- [QList](#)< [Match](#) > [parse](#) (int *errorPos=0, [Cat](#) *errorToken=0, int *incrementalPos=0, [QList](#)< [StackItem](#) > *incrementalStacks=0, [QList](#)< [Match](#) > *incrementalMatches=0, [LexerState](#) *lexerState=0)
parse the input string
- [QList](#)< [Match](#) > [parse](#) ([ParseState](#) *parseState)
overloaded version using ParseState, for bindings
- [Predictions](#) [computePredictions](#) (const [QList](#)< [StackItem](#) > &stacks) const
compute a set of predictions from the stacks produced by an incremental parse
- [Predictions](#) [computePredictions](#) (const [ParseState](#) &parseState) const
overloaded version using ParseState, for bindings
- [QHash](#)< [Cat](#), [QSet](#)< [Cat](#) > > [expandNonterminalPrediction](#) ([CatArg](#) cat) const
expand a nonterminal prediction to the possible initial tokens and the nonterminals they immediately reduce to (for categorization), using recursive descent
- [QHash](#)< [Cat](#), [QSet](#)< [Cat](#) > > [expandNonterminalPredictionC](#) ([CatArg](#) cat)
expand a nonterminal prediction to the possible initial tokens and the nonterminals they immediately reduce to (for categorization), using recursive descent
- [MultiPredictions](#) [computeMultiPredictions](#) (const [QList](#)< [StackItem](#) > &stacks) const
compute a set of multi-token predictions from the stacks produced by an incremental parse
- [MultiPredictions](#) [computeMultiPredictions](#) (const [ParseState](#) &parseState) const
overloaded version using ParseState, for bindings
- [QHash](#)< [QList](#)< [Cat](#) >, [QList](#)< [MultiPrediction](#) > > [computeMultiPredictionsJava](#) (const [ParseState](#) &parseState) const
convert the QMultiHash for the Java binding
- [QHash](#)< [Cat](#), [QSet](#)< [QList](#)< [Cat](#) > > > [expandNonterminalPredictionMulti](#) ([CatArg](#) cat) const
expand a nonterminal prediction to the possible initial nonempty literals (strings of one or more tokens) and the nonterminals they immediately reduce to (for categorization), using recursive descent
- [QHash](#)< [Cat](#), [QSet](#)< [QList](#)< [Cat](#) > > > [expandNonterminalPredictionMultiC](#) ([CatArg](#) cat)
expand a nonterminal prediction to the possible initial nonempty literals (strings of one or more tokens) and the nonterminals they immediately reduce to (for categorization), using recursive descent

- [ConstrainedPredictions computeConstrainedPredictions](#) (const [QList](#)< [StackItem](#) > &stacks) const
compute a set of predictions from the stacks produced by an incremental parse
- [ConstrainedPredictions computeConstrainedPredictions](#) (const [ParseState](#) &parseState) const
overloaded version using [ParseState](#), for bindings
- [QHash](#)< [Cat](#), [QList](#)< [NextTokenConstraints](#) > > [computeConstrainedPredictionsJava](#) (const [ParseState](#) &parseState) const
convert the [QMultiHash](#) for the Java binding
- [QHash](#)< [Cat](#), [QSet](#)< [Cat](#) > > [expandNonterminalPredictionC](#) ([CatArg](#) cat, const [NextTokenConstraints](#) &nextTokenConstraints)
expand a nonterminal prediction to the possible initial tokens and the nonterminals they immediately reduce to (for categorization), using recursive descent
- [QHash](#)< [Cat](#), [QSet](#)< [Cat](#) > > [expandNonterminalPredictionC](#) ([CatArg](#) cat, const [QList](#)< [NextTokenConstraints](#) > &nextTokenConstraintsList)
expand a nonterminal prediction to the possible initial tokens and the nonterminals they immediately reduce to (for categorization), using recursive descent
- [ConstrainedMultiPredictions computeConstrainedMultiPredictions](#) (const [QList](#)< [StackItem](#) > &stacks) const
compute a set of multi-token predictions from the stacks produced by an incremental parse
- [ConstrainedMultiPredictions computeConstrainedMultiPredictions](#) (const [ParseState](#) &parseState) const
overloaded version using [ParseState](#), for bindings
- [QHash](#)< [QList](#)< [Cat](#) >, [QList](#)< [ConstrainedMultiPrediction](#) > > [computeConstrainedMultiPredictionsJava](#) (const [ParseState](#) &parseState) const
convert the [QMultiHash](#) for the Java binding
- [QHash](#)< [Cat](#), [QSet](#)< [QList](#)< [Cat](#) > > > [expandNonterminalPredictionMultiC](#) ([CatArg](#) cat, const [NextTokenConstraints](#) &nextTokenConstraints)
expand a nonterminal prediction to the possible initial nonempty literals (strings of one or more tokens) and the nonterminals they immediately reduce to (for categorization), using recursive descent
- [QHash](#)< [Cat](#), [QSet](#)< [QList](#)< [Cat](#) > > > [expandNonterminalPredictionMultiC](#) ([CatArg](#) cat, const [QList](#)< [NextTokenConstraints](#) > &nextTokenConstraintsList)
expand a nonterminal prediction to the possible initial nonempty literals (strings of one or more tokens) and the nonterminals they immediately reduce to (for categorization), using recursive descent

Public Attributes

grammar

- [RuleSet rules](#)
grammar rules
- [TokenSet tokens](#)
tokens
- [Cat startCat](#)
start category

additional information needed for PMCFGs

- [QHash](#)< [Cat](#), [QPair](#)< [Cat](#), [QList](#)< [Cat](#) > > > [pseudoCats](#)
pseudo-categories, used to represent PMCFGs internally
- [QHash](#)< [Cat](#), [QPair](#)< [Cat](#), int > > > [componentCats](#)
maps categories which represent components of a multi-component category to the category and component index they represent
- [QHash](#)< [Cat](#), [QList](#)< [Cat](#) > > [catComponents](#)
maps multi-component categories to the list of their components

Protected Attributes

- [TokenSource](#) * [inputSource](#)

input source

7.17.1 Detailed Description

main class

Definition at line 1014 of file `dyngenpar.h`.

7.17.2 Constructor & Destructor Documentation

7.17.2.1 `DynGenPar::Parser::Parser (TokenSource * tokenSource)` [inline]

Definition at line 1017 of file `dyngenpar.h`.

7.17.2.2 `virtual DynGenPar::Parser::~~Parser ()` [inline, virtual]

Definition at line 1019 of file `dyngenpar.h`.

7.17.3 Member Function Documentation

7.17.3.1 `bool DynGenPar::Parser::addPmcfgRule (Pmcfg & pmcfg, CatArg cat, const Rule & rule)`

adds a new rule to the grammar (both the PMCFG and the internal representation), updates the nullable categories and the initial graph and clears the other caches

Returns

`true` on success, `false` on failure

Note

Functions can be added by simply calling [Pmcfg::addFunction](#) on the `pmcfg` object (for named functions) or appending to the `pmcfg` object's [Pmcfg::functions](#) member (for unnamed functions).

Definition at line 955 of file `dyngenpar.cpp`.

7.17.3.2 `void DynGenPar::Parser::addRule (CatArg cat, const Rule & rule)`

adds a new rule to the grammar, updates the nullable categories and the initial graph and clears the other caches

Definition at line 675 of file `dyngenpar.cpp`.

7.17.3.3 void DynGenPar::Parser::addToken (CatArg cat) [inline]

Definition at line 1021 of file dyngenpar.h.

7.17.3.4 ConstrainedMultiPredictions DynGenPar::Parser::computeConstrainedMultiPredictions (const QList< StackItem > & stacks) const

compute a set of multi-token predictions from the stacks produced by an incremental parse

Returns

a table of extended categories which are valid continuations for the current input

An extended category is either a nonterminal or a "literal", meaning a nonempty list of tokens appearing in sequence in a rule.

The table is represented as a (possibly multi-valued) hash table mapping a list of categories (containing either exactly one nonterminal or at least one terminal) to

1. another list of categories representing the completed literal in case of a literal (e.g. if the complete literal is "abc" and the user already entered "a", the entry will have key "bc" and value "abc"), and reproducing the key otherwise,
2. the nonterminal the literal appears in (or the nonterminal itself if the predicted category is a nonterminal) and
3. for nonterminals, associated next token constraints.

For terminal/literal predictions, the constraints are immediately validated. For nonterminal predictions, this must be done during expansion.

Definition at line 3101 of file dyngenpar.cpp.

7.17.3.5 ConstrainedMultiPredictions DynGenPar::Parser::computeConstrainedMultiPredictions (const ParseState & parseState) const [inline]

overloaded version using [ParseState](#), for bindings

Definition at line 1100 of file dyngenpar.h.

7.17.3.6 QHash<QList<Cat>, QList<ConstrainedMultiPrediction> > DynGenPar::Parser::computeConstrainedMultiPredictionsJava (const ParseState & parseState) const [inline]

convert the QMultiHash for the Java binding

This method is renamed to `computeConstrainedMultiPredictions` in the Java binding.

DO NOT USE THIS METHOD IN C++ CODE, use [computeConstrainedMultiPredictions](#) instead.

Definition at line 1113 of file dyngenpar.h.

7.17.3.7 **ConstrainedPredictions** DynGenPar::Parser::computeConstrainedPredictions (const QList< StackItem > & stacks) const

compute a set of predictions from the stacks produced by an incremental parse

Returns

a table of (terminal or nonterminal) categories which are valid continuations for the current input and associated next token constraints

For terminal predictions, the constraints are immediately validated. For nonterminal predictions, this must be done during expansion.

Warning

This prediction method does not support multi-token literals.

Definition at line 2949 of file dyngenpar.cpp.

7.17.3.8 **ConstrainedPredictions** DynGenPar::Parser::computeConstrainedPredictions (const ParseState & parseState) const [inline]

overloaded version using [ParseState](#), for bindings

Definition at line 1077 of file dyngenpar.h.

7.17.3.9 **QHash<Cat, QList<NextTokenConstraints>>** DynGenPar::Parser::computeConstrainedPredictionsJava (const ParseState & parseState) const [inline]

convert the QMultiHash for the Java binding

This method is renamed to `computeConstrainedPredictions` in the Java binding.

DO NOT USE THIS METHOD IN C++ CODE, use [computeConstrainedPredictions](#) instead.

Definition at line 1089 of file dyngenpar.h.

7.17.3.10 **MultiPredictions** DynGenPar::Parser::computeMultiPredictions (const ParseState & parseState) const [inline]

overloaded version using [ParseState](#), for bindings

Definition at line 1054 of file dyngenpar.h.

7.17.3.11 MultiPredictions `DynGenPar::Parser::computeMultiPredictions (const QList< StackItem > & stacks) const`

compute a set of multi-token predictions from the stacks produced by an incremental parse

Returns

a table of extended categories which are valid continuations for the current input.

An extended category is either a nonterminal or a "literal", meaning a nonempty list of tokens appearing in sequence in a rule.

The table is represented as a (possibly multi-valued) hash table mapping a list of categories (containing either exactly one nonterminal or at least one terminal) to

1. another list of categories representing the completed literal in case of a literal (e.g. if the complete literal is "abc" and the user already entered "a", the entry will have key "bc" and value "abc"), and reproducing the key otherwise, and
2. the nonterminal the literal appears in (or the nonterminal itself if the predicted category is a nonterminal).

Warning

This prediction method does not support next token constraints.

Definition at line 2608 of file dyngenpar.cpp.

7.17.3.12 QHash<QList<Cat>, QList<MultiPrediction>> `DynGenPar::Parser::computeMultiPredictionsJava (const ParseState & parseState) const` `[inline]`

convert the QMultiHash for the Java binding

This method is renamed to `computeMultiPredictions` in the Java binding.

DO NOT USE THIS METHOD IN C++ CODE, use [computeMultiPredictions](#) instead.

Definition at line 1066 of file dyngenpar.h.

7.17.3.13 Predictions `DynGenPar::Parser::computePredictions (const QList< StackItem > & stacks) const`

compute a set of predictions from the stacks produced by an incremental parse

Returns

a set of (terminal or nonterminal) categories which are valid continuations for the current input

Warning

This prediction method does not support multi-token literals nor next token constraints.

Definition at line 2317 of file dyngenpar.cpp.

7.17.3.14 Predictions DynGenPar::Parser::computePredictions (const ParseState & parseState) const [inline]

overloaded version using [ParseState](#), for bindings

Definition at line 1046 of file dyngenpar.h.

7.17.3.15 QHash< Cat, QSet< Cat > > DynGenPar::Parser::expandNonterminalPrediction (CatArg cat) const

expand a nonterminal prediction to the possible initial tokens and the nonterminals they immediately reduce to (for categorization), using recursive descent

only follow the leftmost branch and ignore left recursion because it does not affect the starting tokens

also expand each category only once because we do not need the full parse trees, only the last category

Warning

This expansion method does not honor context-sensitive constraints (PMCFG constraints, next token constraints) attached to the skipped epsilon matches.

Definition at line 2377 of file dyngenpar.cpp.

7.17.3.16 QHash< Cat, QSet< Cat > > DynGenPar::Parser::expandNonterminalPredictionC (CatArg cat, const NextTokenConstraints & nextTokenConstraints)

expand a nonterminal prediction to the possible initial tokens and the nonterminals they immediately reduce to (for categorization), using recursive descent

only follow the leftmost branch and ignore left recursion because it does not affect the starting tokens

also match all the nullable categories encountered to epsilon, and collect and enforce any context-sensitive constraints

This overload also enforces the next token constraints passed as a second argument.

Definition at line 3017 of file dyngenpar.cpp.

7.17.3.17 QHash< Cat, QSet< Cat > > DynGenPar::Parser::expandNonterminalPredictionC (CatArg cat, const QList< NextTokenConstraints > & nextTokenConstraintsList)

expand a nonterminal prediction to the possible initial tokens and the nonterminals they immediately reduce to (for categorization), using recursive descent

only follow the leftmost branch and ignore left recursion because it does not affect the starting tokens

also match all the nullable categories encountered to epsilon, and collect and enforce any context-sensitive constraints

This overload also enforces the disjunctive (inclusive OR) list of next token constraints passed as a second argument, i.e. if any of the next token constraint sets in *nextTokenConstraintsList* matches, the prediction is accepted.

Definition at line 3044 of file dyngenpar.cpp.

7.17.3.18 QHash< Cat, QSet< Cat > > DynGenPar::Parser::expandNonterminalPredictionC (CatArg cat)

expand a nonterminal prediction to the possible initial tokens and the nonterminals they immediately reduce to (for categorization), using recursive descent

only follow the leftmost branch and ignore left recursion because it does not affect the starting tokens

also match all the nullable categories encountered to epsilon, and collect and enforce any context-sensitive constraints

Definition at line 2576 of file dyngenpar.cpp.

7.17.3.19 QHash< Cat, QSet< QList< Cat > > > DynGenPar::Parser::expandNonterminalPredictionMulti (CatArg cat) const

expand a nonterminal prediction to the possible initial nonempty literals (strings of one or more tokens) and the nonterminals they immediately reduce to (for categorization), using recursive descent

only follow the leftmost branch and ignore left recursion because it does not affect the starting tokens

also expand each category only once because we do not need the full parse trees, only the last category

Warning

This expansion method does not honor context-sensitive constraints (PMCFG constraints, next token constraints) attached to the skipped epsilon matches.

Definition at line 2720 of file dyngenpar.cpp.

7.17.3.20 QHash< Cat, QSet< QList< Cat > > > DynGenPar::Parser::expandNonterminalPredictionMultiC (CatArg cat, const QList< NextTokenConstraints > & nextTokenConstraintsList)

expand a nonterminal prediction to the possible initial nonempty literals (strings of one or more tokens) and the nonterminals they immediately reduce to (for categorization), using recursive descent

only follow the leftmost branch and ignore left recursion because it does not affect the starting tokens

also match all the nullable categories encountered to epsilon, and collect and enforce any context-sensitive constraints

This overload also enforces the disjunctive (inclusive OR) list of next token constraints passed as a second argument, i.e. if any of the next token constraint sets in *nextTokenConstraintsList* matches, the prediction is accepted.

Definition at line 3227 of file dyngenpar.cpp.

7.17.3.21 QHash< Cat, QSet< QList< Cat > > > DynGenPar::Parser::expandNonterminalPredictionMultiC (CatArg cat)

expand a nonterminal prediction to the possible initial nonempty literals (strings of one or more tokens) and the nonterminals they immediately reduce to (for categorization), using recursive descent

only follow the leftmost branch and ignore left recursion because it does not affect the starting tokens

also match all the nullable categories encountered to epsilon, and collect and enforce any context-sensitive constraints

Definition at line 2926 of file dyngenpar.cpp.

7.17.3.22 `QHash< Cat, QSet< QList< Cat > > > DynGenPar::Parser::expandNonterminalPredictionMultiC (CatArg cat, const NextTokenConstraints & nextTokenConstraints)`

expand a nonterminal prediction to the possible initial nonempty literals (strings of one or more tokens) and the nonterminals they immediately reduce to (for categorization), using recursive descent

only follow the leftmost branch and ignore left recursion because it does not affect the starting tokens

also match all the nullable categories encountered to epsilon, and collect and enforce any context-sensitive constraints

This overload also enforces the next token constraints passed as a second argument.

Definition at line 3199 of file `dyngenpar.cpp`.

7.17.3.23 `void DynGenPar::Parser::initCaches (void)`

clears all caches, then computes the nullable categories and the initial graph

should be called after each direct grammar modification (`addRule` takes care of updating the caches, which is more efficient than clearing.)

Definition at line 564 of file `dyngenpar.cpp`.

7.17.3.24 `bool DynGenPar::Parser::isLiteral (const QList< Cat > & list) const`

is a given list of categories a literal?

Returns

`true` if the given `list` of categories is a literal, i.e. contains only tokens, `false` otherwise

Definition at line 539 of file `dyngenpar.cpp`.

7.17.3.25 `bool DynGenPar::Parser::isToken (CatArg cat) const [inline]`

Definition at line 1020 of file `dyngenpar.h`.

7.17.3.26 `void DynGenPar::Parser::loadCfg (const Cfg & cfg) [inline]`

Definition at line 1025 of file `dyngenpar.h`.

7.17.3.27 bool DynGenPar::Parser::loadPmcf (const Pmcf & pmcf)

loads a PMCFG in standard form, converting it to the internal representation

Rules containing categories which are neither tokens nor have rules are discarded, as they're unreachable and as we cannot transform them without knowing the dimension of the unused categories.

Returns

true on success, false on failure

Warning

The parser rules may be in an inconsistent state if the loading failed.

Definition at line 913 of file dyngenpar.cpp.

7.17.3.28 QList<Match> DynGenPar::Parser::parse (ParseState * parseState) [inline]

overloaded version using [ParseState](#), for bindings

Definition at line 1039 of file dyngenpar.h.

7.17.3.29 QList< Match > DynGenPar::Parser::parse (int * errorPos = 0, Cat * errorToken = 0, int * incrementalPos = 0, QList< StackItem > * incrementalStacks = 0, QList< Match > * incrementalMatches = 0, LexerState * lexerState = 0)

parse the input string

Returns

the list of matches

Parameters

out	<i>errorPos</i>	if non-NULL, is filled with -1 on success and with the number of accepted tokens before the error occurred on error (Caution: This might not be a good position indicator to show to the end user. For some token sources, lexerState contains a more user-centric position indicator which can be obtained through that token source's API.)
out	<i>errorToken</i>	if non-NULL, is filled with 0 (epsilon) on success and with the token triggering the error on error.
in, out	<i>incrementalPos</i>	should be NULL for a non-incremental parse, a pointer to a negative integer to start an incremental parse or a pointer to a nonnegative integer to continue an incremental parse. It will be set to the current end of input if non-NULL.

in, out	<i>incremental-Stacks</i>	should be <code>NULL</code> for a non-incremental, non-predictive parse or a pointer to QList<StackItem> (initially empty) for an incremental parse or when needed for prediction. It represents the internal parser states. Normally, this will be the list of stacks at the end of the parsing process. However, if an error occurred, that list would always be empty, so instead, we return the list of stacks before the token triggering the error, thus allowing to use the prediction functionality to report what token would have been expected instead of the faulty one. (An empty list of stacks means that the input was expected to end before the faulty token.)
in, out	<i>incremental-Matches</i>	should be <code>NULL</code> for a non-incremental parse. For an incremental parse, it can be <code>NULL</code> if you do not need to get your matches back a second time when there is no new input. If set to non- <code>NULL</code> , it will be returned as is if there is no new input in an incremental parse, or updated to the current return value otherwise.
in, out	<i>lexerState</i>	if non- <code>NULL</code> , is filled with the lexer state at the end of the (incremental) parsing process. (In case of an error, it is filled with the lexer state where the error occurred, i.e. before shifting the faulty token, to allow reporting error positions accurately.) It is useful to allow rewinding to a previous position with a stateful lexer. It can be <code>NULL</code> for a non-incremental or a sequential incremental parse (i.e. if rewinding is not needed) or if a stateless token source (stateless lexer, token buffer etc.) is used. When starting a new incremental parse, the lexer state pointed to should be a null (default-constructed) LexerState . If the <i>lexerState</i> pointer is <code>NULL</code> , all rewind operations for the lexer will be passed a null (default-constructed) LexerState ; stateful lexers will then fail any rewind operations.

Note

An "error" is defined as a place at which it is no longer possible to continue parsing. This does not include incomplete input, i.e. input which can be continued to valid input, but does not form valid input by itself. If an empty list of matches is returned without an error being flagged, this means that the input was incomplete. Use one of the prediction functions (e.g. [computePredictions](#)) to list possible continuations ("expected ..."). Incremental parsing can be used to process additional input given by the user.

Definition at line 2239 of file `dyngenpar.cpp`.

7.17.4 Member Data Documentation**7.17.4.1 `QHash<Cat, QList<Cat>>` `DynGenPar::Parser::catComponents`**

maps multi-component categories to the list of their components

used during the import of PMCFG rules in the standard representation

can be left out if the internal representation is used

Definition at line 1173 of file `dyngenpar.h`.

7.17.4.2 `QHash<Cat, QPair<Cat, int> >` `DynGenPar::Parser::componentCats`

maps categories which represent components of a multi-component category to the category and component index they represent

also used to look up whether a category is a component of a multi-component category

Definition at line 1167 of file `dyngenpar.h`.

7.17.4.3 `TokenSource*` `DynGenPar::Parser::inputSource` `[protected]`

input source

Definition at line 1178 of file `dyngenpar.h`.

7.17.4.4 `QHash<Cat, QPair<Cat, QList<Cat> > >` `DynGenPar::Parser::pseudoCats`

pseudo-categories, used to represent PMCFGs internally

maps a pseudo-category to:

1. the actual component of the multidimensional category this pseudo-category stands for - just substituting this for the pseudo-category results in the context-free approximation of the PMCFG
2. the list of pseudo-categories resulting from the use of the SAME argument (not just the same category) in the same context - all those pseudo-categories, when used in the same context, have to be expanded using matching rules; we use top-down expansion for all except the first encountered one to guarantee the same expansion as the one obtained while reducing the first one

Note that, when parsing a PMCFG, the initial graph and the set of nullable categories are the ones for the context-free approximation. The PMCFG constraints are only evaluated during the matching resp. reducing steps.

Also note that tokens are always 1-dimensional, so tokens may not be pseudo-categories nor the actual component for a pseudo-category.

Definition at line 1161 of file `dyngenpar.h`.

7.17.4.5 `RuleSet` `DynGenPar::Parser::rules`

grammar rules

Warning

Modifying the grammar directly requires [initCaches](#), use [addRule](#) if possible.

Definition at line 1131 of file `dyngenpar.h`.

7.17.4.6 Cat DynGenPar::Parser::startCat

start category

Definition at line 1135 of file dyngenpar.h.

7.17.4.7 TokenSet DynGenPar::Parser::tokens

tokens

Definition at line 1133 of file dyngenpar.h.

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

- [dyngenpar.h](#)
- [dyngenpar.cpp](#)

7.18 DynGenPar::ParseState Struct Reference

parse state struct, for bindings

Public Member Functions

- [ParseState](#) ()
- [ParseState](#) (const [ParseState](#) &other)
- void [reset](#) ()

Public Attributes

- int [errorPos](#)
- Cat [errorToken](#)
- int [incrementalPos](#)
- [QList](#)< [StackItem](#) > [incrementalStacks](#)
- [QList](#)< [Match](#) > [incrementalMatches](#)
- [LexerState](#) [lexerState](#)

7.18.1 Detailed Description

parse state struct, for bindings

Definition at line 988 of file dyngenpar.h.

7.18.2 Constructor & Destructor Documentation

7.18.2.1 `DynGenPar::ParseState::ParseState ()` [inline]

Definition at line 989 of file `dyngenpar.h`.

7.18.2.2 `DynGenPar::ParseState::ParseState (const ParseState & other)` [inline]

Definition at line 993 of file `dyngenpar.h`.

7.18.3 Member Function Documentation

7.18.3.1 `void DynGenPar::ParseState::reset ()` [inline]

Definition at line 1007 of file `dyngenpar.h`.

7.18.4 Member Data Documentation

7.18.4.1 `int DynGenPar::ParseState::errorPos`

Definition at line 1000 of file `dyngenpar.h`.

7.18.4.2 `Cat DynGenPar::ParseState::errorToken`

Definition at line 1001 of file `dyngenpar.h`.

7.18.4.3 `QList<Match> DynGenPar::ParseState::incrementalMatches`

Definition at line 1004 of file `dyngenpar.h`.

7.18.4.4 `int DynGenPar::ParseState::incrementalPos`

Definition at line 1002 of file `dyngenpar.h`.

7.18.4.5 `QList<StackItem> DynGenPar::ParseState::incrementalStacks`

Definition at line 1003 of file `dyngenpar.h`.

7.18.4.6 LexerState DynGenPar::ParseState::lexerState

Definition at line 1005 of file dyngenpar.h.

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

- [dyngenpar.h](#)

7.19 DynGenPar::Pgf Struct Reference

representation of the information in .pgf files in a format we can process

Public Member Functions

- [Pgf \(\)](#)
dummy default constructor for bindings
- [Pgf \(const QString &fileName, const QString &concreteName=QString\(\)\)](#)
constructor, loads concrete syntax from .pgf file

Public Attributes

- [Pmcfq pmcfq](#)
the PMCFG (in (almost) standard form)
- [QStringList catNames](#)
names of categories, in general not unique
- [QStringList functionNames](#)
names of functions, in general not unique
- [QHash< QString, int > tokenHash](#)
hash table for quick token lexing
- [QList< QPair< QString, int > > suffixes](#)
list of &+ suffixes with their IDs
- [QHash< QString, QStringList > componentNames](#)
names of category components
- [int firstFunction](#)
the function ID of the first non-coercion function

7.19.1 Detailed Description

representation of the information in .pgf files in a format we can process

Definition at line 49 of file pgf.h.

7.19.2 Constructor & Destructor Documentation

7.19.2.1 DynGenPar::Pgf::Pgf () [inline]

dummy default constructor for bindings

Definition at line 51 of file pgf.h.

7.19.2.2 DynGenPar::Pgf::Pgf (const QString & fileName, const QString & concreteName = QString())

constructor, loads concrete syntax from .pgf file

Loads a .pgf file from disk and imports it into a format we can process.

Only one concrete grammar is loaded. If the .pgf file contains more than one concrete grammar, the name of the concrete grammar to load must be specified (otherwise a fatal error is raised).

Definition at line 513 of file pgf.cpp.

7.19.3 Member Data Documentation

7.19.3.1 QStringList DynGenPar::Pgf::catNames

names of categories, in general not unique

Definition at line 56 of file pgf.h.

7.19.3.2 QHash<QString, QStringList> DynGenPar::Pgf::componentNames

names of category components

Definition at line 61 of file pgf.h.

7.19.3.3 int DynGenPar::Pgf::firstFunction

the function ID of the first non-coercion function

Definition at line 62 of file pgf.h.

7.19.3.4 QStringList DynGenPar::Pgf::functionNames

names of functions, in general not unique

Definition at line 57 of file pgf.h.

7.19.3.5 Pmcfg DynGenPar::Pgf::pmcfg

the PMCFG (in (almost) standard form)

Definition at line 55 of file pgf.h.

7.19.3.6 QList<QPair<QString, int> > DynGenPar::Pgf::suffixes

list of &+ suffixes with their IDs

Definition at line 59 of file pgf.h.

7.19.3.7 QHash<QString, int> DynGenPar::Pgf::tokenHash

hash table for quick token lexing

Definition at line 58 of file pgf.h.

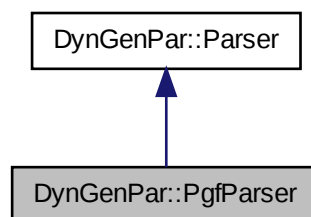
The documentation for this struct was generated from the following files:

- [pgf.h](#)
- [pgf.cpp](#)

7.20 DynGenPar::PgfParser Class Reference

parser for PGF grammars

Inheritance diagram for DynGenPar::PgfParser:



Public Member Functions

- [PgfParser](#) (const [Pgf](#) &p)
- [PgfParser](#) (const [QString](#) &fileName, const [QString](#) &concreteName=[QString](#)())
- virtual [~PgfParser](#) ()
- void [setInputStdin](#) ()
- void [setInputFile](#) (const [QString](#) &fileName)
- void [setInputBytes](#) (const [QByteArray](#) &bytes)
- void [setInputString](#) (const [QString](#) &string)
- void [setInputBuffer](#) ([QByteArray](#) *buffer)
- [QString](#) [catName](#) (int cat) const
- [QString](#) [functionName](#) (int id) const
- void [filterCoercionsFromSyntaxTree](#) ([Node](#) &tree) const
- [Node](#) [filterCoercionsFromSyntaxTree](#) (const [Node](#) &tree) const
for bindings

Public Attributes

- [Pgf](#) pgf

7.20.1 Detailed Description

parser for PGF grammars

Definition at line 66 of file pgf.h.

7.20.2 Constructor & Destructor Documentation

7.20.2.1 [DynGenPar::PgfParser::PgfParser](#) (const [Pgf](#) & p)

Definition at line 1163 of file pgf.cpp.

7.20.2.2 [DynGenPar::PgfParser::PgfParser](#) (const [QString](#) & fileName, const [QString](#) & concreteName =[QString](#)())

Definition at line 1168 of file pgf.cpp.

7.20.2.3 virtual [DynGenPar::PgfParser::~~PgfParser](#) () [inline, virtual]

Definition at line 70 of file pgf.h.

7.20.3 Member Function Documentation

7.20.3.1 [QString](#) [DynGenPar::PgfParser::catName](#) (int cat) const

Definition at line 1199 of file pgf.cpp.

7.20.3.2 void DynGenPar::PgParser::filterCoercionsFromSyntaxTree (Node & tree) const

Definition at line 1219 of file pgf.cpp.

7.20.3.3 Node DynGenPar::PgParser::filterCoercionsFromSyntaxTree (const Node & tree) const [inline]

for bindings

Definition at line 80 of file pgf.h.

7.20.3.4 QString DynGenPar::PgParser::functionName (int id) const [inline]

Definition at line 77 of file pgf.h.

7.20.3.5 void DynGenPar::PgParser::setInputBuffer (QByteArray * buffer)

Definition at line 1194 of file pgf.cpp.

7.20.3.6 void DynGenPar::PgParser::setInputBytes (const QByteArray & bytes)

Definition at line 1184 of file pgf.cpp.

7.20.3.7 void DynGenPar::PgParser::setInputFile (const QString & fileName)

Definition at line 1179 of file pgf.cpp.

7.20.3.8 void DynGenPar::PgParser::setInputStdin ()

Definition at line 1174 of file pgf.cpp.

7.20.3.9 void DynGenPar::PgParser::setInputString (const QString & string)

Definition at line 1189 of file pgf.cpp.

7.20.4 Member Data Documentation

7.20.4.1 Pgf DynGenPar::PgParser::pgf

Definition at line 85 of file pgf.h.

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

- [pgf.h](#)
- [pgf.cpp](#)

7.21 DynGenPar::PmcfG Struct Reference

PMCFG.

Public Attributes

grammar

- [QList< Function > functions](#)
list of PMCFG functions
- [RuleSet rules](#)
set of PMCFG rules
- [TokenSet tokens](#)
set of true tokens
- [Cat startCat](#)
start category
- [RuleSet cfRules](#)
optional context-free rules

function name tables (optional)

- [QHash< int, QString > functionNames](#)
- [QHash< QString, int > functionIndices](#)
- void [addFunction](#) (const QString &name, const [Function](#) &function)
- [Function lookupFunction](#) (const QVariant &id) const

7.21.1 Detailed Description

PMCFG.

Definition at line 906 of file dyngenpar.h.

7.21.2 Member Function Documentation

7.21.2.1 void DynGenPar::PmcfG::addFunction (const QString & name, const Function & function) [\[inline\]](#)

Definition at line 954 of file dyngenpar.h.

7.21.2.2 Function DynGenPar::PmcfG::lookupFunction (const QVariant & id) const [\[inline\]](#)

Definition at line 960 of file dyngenpar.h.

7.21.3 Member Data Documentation

7.21.3.1 RuleSet DynGenPar::Pmcfg::cfRules

optional context-free rules

allows specifying rules for context-free categories which can be used as "token" terms in PMCFG functions, e.g. A -> "a" | "an"

Warning

The exact expansion used will be reflected only in the parse tree, not in the PMCFG syntax tree. Do not use this feature if you need to know which exact expression was used.

Definition at line 947 of file dyngenpar.h.

7.21.3.2 QHash<QString, int> DynGenPar::Pmcfg::functionIndices

Definition at line 953 of file dyngenpar.h.

7.21.3.3 QHash<int, QString> DynGenPar::Pmcfg::functionNames

Definition at line 952 of file dyngenpar.h.

7.21.3.4 QList<Function> DynGenPar::Pmcfg::functions

list of PMCFG functions

This list does not store function names. They can be optionally used and are stored in [functionNames](#) and [functionIndices](#).

Definition at line 913 of file dyngenpar.h.

7.21.3.5 RuleSet DynGenPar::Pmcfg::rules

set of PMCFG rules

Rules should be labeled with the index of the function in [functions](#) or its name as found in [functionNames](#) and [functionIndices](#). The expression of the rule is interpreted as the argument list for the function. For example:

```
Rule(1) << "A" << "B"; // calls function #1 with (A, B) as parameters
Rule("f") << "A" << "B"; // calls f(A, B)
```

In a standard PMCFG, the argument list may contain only PMCFG nonterminals. The syntax tree then only contains the function and the syntax trees for each argument. This implementation also allows tokens and context-free nonterminals as function arguments, in which case the syntax tree will contain the parse tree for the context-free argument. In particular, in the case of a token, the data attached to the token is retained.

Definition at line 932 of file dyngenpar.h.

7.21.3.6 Cat DynGenPar::PmcfG::startCat

start category

The start category must be 1-dimensional.

Definition at line 938 of file dyngenpar.h.

7.21.3.7 TokenSet DynGenPar::PmcfG::tokens

set of true tokens

(must NOT contain context-free nonterminals)

Definition at line 935 of file dyngenpar.h.

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

- [dyngenpar.h](#)

7.22 DynGenPar::PmcfGComponentInfo Struct Reference

attached to the parse trees as rule labels to allow obtaining syntax trees

Public Member Functions

- [PmcfGComponentInfo](#) ()
- [PmcfGComponentInfo](#) (const [Rule](#) &rule)

Public Attributes

- [Rule](#) [pmcfGRule](#)
- [QVector](#)< [QVector](#)< int > > [argPositions](#)

7.22.1 Detailed Description

attached to the parse trees as rule labels to allow obtaining syntax trees

Definition at line 971 of file dyngenpar.h.

7.22.2 Constructor & Destructor Documentation

7.22.2.1 DynGenPar::PmcfGComponentInfo::PmcfGComponentInfo () [inline]

Definition at line 972 of file dyngenpar.h.

7.22.2.2 DynGenPar::PmcfComponentInfo::PmcfComponentInfo (const Rule & rule) [inline]

Definition at line 973 of file dyngenpar.h.

7.22.3 Member Data Documentation

7.22.3.1 QVector<QVector<int> > DynGenPar::PmcfComponentInfo::argPositions

must be the same size as [pmcfRule](#) (even if the last arguments are not used)

Definition at line 976 of file dyngenpar.h.

7.22.3.2 Rule DynGenPar::PmcfComponentInfo::pmcfRule

Definition at line 975 of file dyngenpar.h.

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

- [dyngenpar.h](#)

7.23 DynGenPar::PseudoCatScope Class Reference

Public Member Functions

- [PseudoCatScope](#) ()
- [QHash](#)< [Cat](#), [QPair](#)< [QPair](#)< [Node](#), [NextTokenConstraints](#) >, [int](#) > > & [pConstraints](#) ()
- [QHash](#)< [Cat](#), [QPair](#)< [int](#), [PseudoCatScope](#) > > & [mcfConstraints](#) ()
- [bool](#) [hasPConstraint](#) ([CatArg](#) cat) const
- [bool](#) [hasMcfConstraint](#) ([CatArg](#) cat) const
- [QPair](#)< [QPair](#)< [Node](#), [NextTokenConstraints](#) >, [int](#) > [pConstraint](#) ([CatArg](#) cat) const
- [QPair](#)< [int](#), [PseudoCatScope](#) > [mcfConstraint](#) ([CatArg](#) cat) const
- [bool](#) [isNull](#) () const
- [const](#) [PseudoCatScopeData](#) * [data](#) () const
needed for hash tables
- [bool](#) [operator==](#) (const [PseudoCatScope](#) &other) const

7.23.1 Detailed Description

Definition at line 307 of file dyngenpar.h.

7.23.2 Constructor & Destructor Documentation

7.23.2.1 DynGenPar::PseudoCatScope::PseudoCatScope () [inline]

Definition at line 309 of file dyngenpar.h.

7.23.3 Member Function Documentation

7.23.3.1 const PseudoCatScopeData* DynGenPar::PseudoCatScope::data () const [inline]

needed for hash tables

Definition at line 334 of file dyngenpar.h.

7.23.3.2 bool DynGenPar::PseudoCatScope::hasMcfgConstraint (CatArg cat) const [inline]

Definition at line 321 of file dyngenpar.h.

7.23.3.3 bool DynGenPar::PseudoCatScope::hasPConstraint (CatArg cat) const [inline]

Definition at line 318 of file dyngenpar.h.

7.23.3.4 bool DynGenPar::PseudoCatScope::isNull () const [inline]

Definition at line 332 of file dyngenpar.h.

7.23.3.5 QPair<int, PseudoCatScope> DynGenPar::PseudoCatScope::mcfgConstraint (CatArg cat) const [inline]

Definition at line 329 of file dyngenpar.h.

7.23.3.6 QHash<Cat, QPair<int, PseudoCatScope> >& DynGenPar::PseudoCatScope::mcfgConstraints () [inline]

Definition at line 314 of file dyngenpar.h.

7.23.3.7 bool DynGenPar::PseudoCatScope::operator==(const PseudoCatScope & other) const [inline]

Definition at line 335 of file dyngenpar.h.

7.23.3.8 `QPair<QPair<Node, NextTokenConstraints>, int> DynGenPar::PseudoCatScope::pConstraint (CatArg cat)`
`const [inline]`

Definition at line 324 of file dyngenpar.h.

7.23.3.9 `QHash<Cat, QPair<QPair<Node, NextTokenConstraints>, int> > & DynGenPar::PseudoCatScope::pConstraints`
`() [inline]`

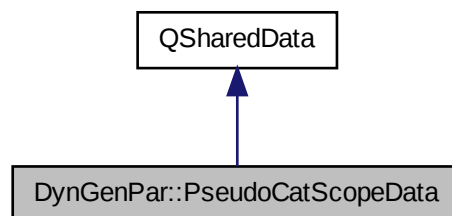
Definition at line 310 of file dyngenpar.h.

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

- [dyngenpar.h](#)

7.24 DynGenPar::PseudoCatScopeData Class Reference

Inheritance diagram for DynGenPar::PseudoCatScopeData:



Public Attributes

- `QHash< Cat, QPair< QPair< Node, NextTokenConstraints >, int > > pConstraints`
hash table recording parallel constraints
- `QHash< Cat, QPair< int, PseudoCatScope > > mcfgConstraints`
hash table recording MCFG constraints

7.24.1 Detailed Description

Definition at line 292 of file dyngenpar.h.

7.24.2 Member Data Documentation

7.24.2.1 `QHash<Cat, QPair<int, PseudoCatScope>> > DynGenPar::PseudoCatScopeData::mcfgConstraints`

hash table recording MCFG constraints

record the rule number to use for a pseudo-category to match the one used for the first encountered pseudo-category, and the scope to reuse

Definition at line 305 of file `dyngenpar.h`.

7.24.2.2 `QHash<Cat, QPair<QPair<Node, NextTokenConstraints>, int>> > DynGenPar::PseudoCatScopeData::pConstraints`

hash table recording parallel constraints

record the tree for each pseudo-category and, for efficiency, the length of the matched string so we can match the exact same token string if the exact same pseudo-category is used again: this is the "parallel" in PMCFGs; also record the next token constraints

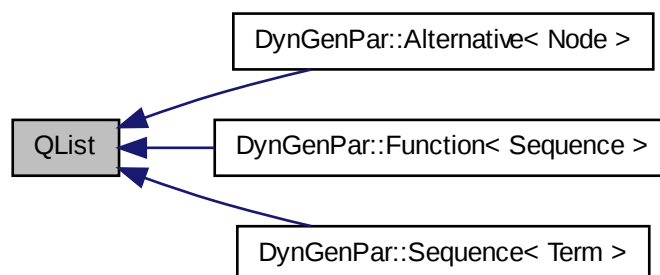
Definition at line 300 of file `dyngenpar.h`.

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

- [dyngenpar.h](#)

7.25 QList Class Reference

Inheritance diagram for QList:

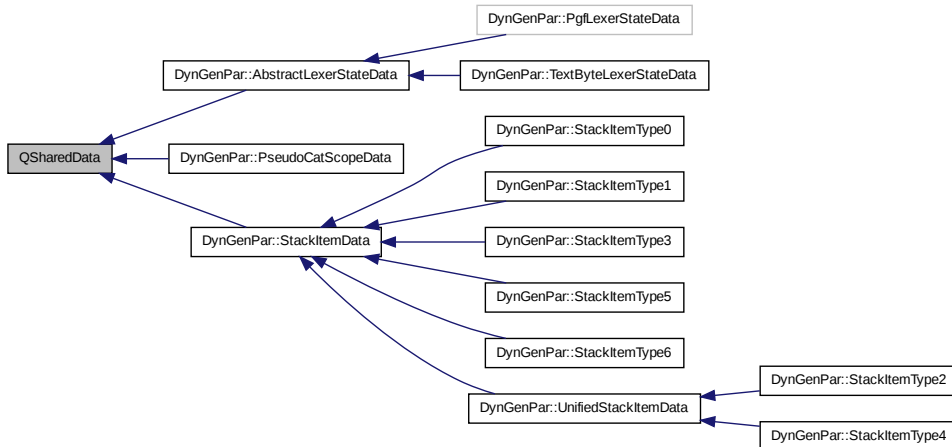


The documentation for this class was generated from the following file:

- [dyngenpar.h](#)

7.26 QSharedData Class Reference

Inheritance diagram for QSharedData:

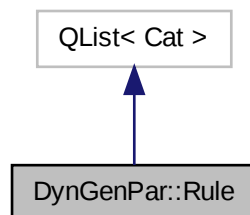


The documentation for this class was generated from the following file:

- [dyngenpar.h](#)

7.27 DynGenPar::Rule Class Reference

Inheritance diagram for DynGenPar::Rule:



Public Member Functions

- [Rule](#) ()
 - [Rule](#) (const QVariant &label)
 - [Rule](#) (const QList< [Cat](#) > &list)
 - [Rule](#) (const QList< [Cat](#) > &list, const QVariant &label)
 - QVariant [label](#) () const
 - void [setLabel](#) (const QVariant &label)
 - [Rule](#) & [operator+=](#) (const QList< [Cat](#) > &other)
 - [Rule](#) & [operator+=](#) (const [Cat](#) &value)
 - [Rule](#) & [operator<<](#) (const QList< [Cat](#) > &other)
 - [Rule](#) & [operator<<](#) (const [Cat](#) &value)
 - void [add](#) (const [Cat](#) &value)
- Java-style + the binding generator doesn't detect the inherited append.*
- QList< [Cat](#) > & [toList](#) ()
for bindings

Public Attributes

- NextTokenConstraints [nextTokenConstraints](#)
- Action * [action](#)

7.27.1 Detailed Description

Definition at line 116 of file dyngenpar.h.

7.27.2 Constructor & Destructor Documentation

7.27.2.1 DynGenPar::Rule::Rule () [inline]

Definition at line 118 of file dyngenpar.h.

7.27.2.2 DynGenPar::Rule::Rule (const QVariant & *label*) [inline, explicit]

Definition at line 119 of file dyngenpar.h.

7.27.2.3 DynGenPar::Rule::Rule (const QList< [Cat](#) > & *list*) [inline, explicit]

Definition at line 121 of file dyngenpar.h.

7.27.2.4 DynGenPar::Rule::Rule (const QList< [Cat](#) > & *list*, const QVariant & *label*) [inline]

Definition at line 123 of file dyngenpar.h.

7.27.3 Member Function Documentation

7.27.3.1 `void DynGenPar::Rule::add (const Cat & value) [inline]`

Java-style + the binding generator doesn't detect the inherited append.

Definition at line 146 of file dyngenpar.h.

7.27.3.2 `QVariant DynGenPar::Rule::label () const [inline]`

Definition at line 125 of file dyngenpar.h.

7.27.3.3 `Rule& DynGenPar::Rule::operator+= (const Cat & value) [inline]`

Definition at line 133 of file dyngenpar.h.

7.27.3.4 `Rule& DynGenPar::Rule::operator+= (const QList< Cat > & other) [inline]`

Definition at line 129 of file dyngenpar.h.

7.27.3.5 `Rule& DynGenPar::Rule::operator<< (const QList< Cat > & other) [inline]`

Definition at line 137 of file dyngenpar.h.

7.27.3.6 `Rule& DynGenPar::Rule::operator<< (const Cat & value) [inline]`

Definition at line 141 of file dyngenpar.h.

7.27.3.7 `void DynGenPar::Rule::setLabel (const QVariant & label) [inline]`

Definition at line 126 of file dyngenpar.h.

7.27.3.8 `QList<Cat>& DynGenPar::Rule::toList () [inline]`

for bindings

Definition at line 150 of file dyngenpar.h.

7.27.4 Member Data Documentation

7.27.4.1 `Action* DynGenPar::Rule::action`

Definition at line 128 of file dyngenpar.h.

7.27.4.2 NextTokenConstraints DynGenPar::Rule::nextTokenConstraints

Definition at line 127 of file dyngenpar.h.

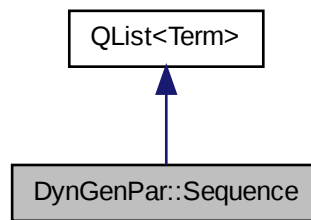
The documentation for this class was generated from the following file:

- [dyngenpar.h](#)

7.28 DynGenPar::Sequence Class Reference

component of a PMCFG function, a sequence of terms

Inheritance diagram for DynGenPar::Sequence:



Public Member Functions

- [Sequence](#) ()
- [Sequence](#) (const [NextTokenConstraints](#) &ntc)
- [Sequence](#) (const [QList](#)< [Term](#) > &list)
- [Sequence](#) (const [QList](#)< [Term](#) > &list, const [NextTokenConstraints](#) &ntc)
- [Sequence](#) & [operator+=](#) (const [QList](#)< [Term](#) > &other)
- [Sequence](#) & [operator+=](#) (const [Term](#) &value)
- [Sequence](#) & [operator<<](#) (const [QList](#)< [Term](#) > &other)
- [Sequence](#) & [operator<<](#) (const [Term](#) &value)
- void [add](#) (const [Term](#) &value)
Java-style (for consistency, even though append is detected here)
- [QList](#)< [Term](#) > & [toList](#) ()
for bindings

Public Attributes

- [NextTokenConstraints](#) [nextTokenConstraints](#)

7.28.1 Detailed Description

component of a PMCFG function, a sequence of terms

Definition at line 837 of file dyngenpar.h.

7.28.2 Constructor & Destructor Documentation

7.28.2.1 DynGenPar::Sequence::Sequence () [inline]

Definition at line 841 of file dyngenpar.h.

7.28.2.2 DynGenPar::Sequence::Sequence (const NextTokenConstraints & ntc) [inline, explicit]

Definition at line 842 of file dyngenpar.h.

7.28.2.3 DynGenPar::Sequence::Sequence (const QList< Term > & list) [inline, explicit]

Definition at line 844 of file dyngenpar.h.

7.28.2.4 DynGenPar::Sequence::Sequence (const QList< Term > & list, const NextTokenConstraints & ntc) [inline]

Definition at line 846 of file dyngenpar.h.

7.28.3 Member Function Documentation

7.28.3.1 void DynGenPar::Sequence::add (const Term & value) [inline]

Java-style (for consistency, even though append is detected here)

Definition at line 865 of file dyngenpar.h.

7.28.3.2 Sequence& DynGenPar::Sequence::operator+= (const QList< Term > & other) [inline]

Definition at line 848 of file dyngenpar.h.

7.28.3.3 Sequence& DynGenPar::Sequence::operator+= (const Term & value) [inline]

Definition at line 852 of file dyngenpar.h.

7.28.3.4 Sequence& DynGenPar::Sequence::operator<< (const Term & value) [inline]

Definition at line 860 of file dyngenpar.h.

7.28.3.5 Sequence& DynGenPar::Sequence::operator<< (const QList< Term > & other) [inline]

Definition at line 856 of file dyngenpar.h.

7.28.3.6 QList<Term>& DynGenPar::Sequence::toList () [inline]

for bindings

Definition at line 869 of file dyngenpar.h.

7.28.4 Member Data Documentation

7.28.4.1 NextTokenConstraints DynGenPar::Sequence::nextTokenConstraints

Definition at line 839 of file dyngenpar.h.

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

- [dyngenpar.h](#)

7.29 DynGenPar::StackItem Class Reference

Public Member Functions

- [StackItem](#) ()
invalid type
- [StackItem](#) (const QList< [StackItem](#) > &parents, [CatArg](#) cat, [CatArg](#) effCat, int pos, const [PseudoCatScope](#) &scope)
type 0
- [StackItem](#) (const QList< [StackItem](#) > &parents, [CatArg](#) cat, [CatArg](#) effCat, const [PseudoCatScope](#) &scope)
type 1
- [StackItem](#) (const [StackItem](#) &parent, int dummy)
type 2
- [StackItem](#) (const [StackItem](#) &parent, const [Rule](#) &rule, int len, int curr, int i, const [Node](#) &tree, const [PseudoCatScope](#) &scope, int ruleno, const [NextTokenConstraints](#) &nextTokenConstraints)
type 3
- [StackItem](#) (const [StackItem](#) &parent, [CatArg](#) target, int pos, int len)
type 4
- [StackItem](#) (const [StackItem](#) &parent, [CatArg](#) cat, const [PseudoCatScope](#) &scope)

type 5

- `StackItem` (const `StackItem` &parent, const `QList`< `Node` > &leaves, int i, const `Node` &tree, const `PseudoCatScope` &scope, const `NextTokenConstraints` &nextTokenConstraints)

type 6

- int `type` () const
- void `addParent` (const `StackItem` &parent)
- void `setParents` (const `QList`< `StackItem` > &parents)
- const `StackItemData` * `data` () const

7.29.1 Detailed Description

Definition at line 410 of file `dyngenpar.h`.

7.29.2 Constructor & Destructor Documentation

7.29.2.1 `DynGenPar::StackItem::StackItem ()` `[inline]`

invalid type

Definition at line 412 of file `dyngenpar.h`.

7.29.2.2 `DynGenPar::StackItem::StackItem (const QList< StackItem > & parents, CatArg cat, CatArg effCat, int pos, const PseudoCatScope & scope)` `[inline]`

type 0

Definition at line 627 of file `dyngenpar.h`.

7.29.2.3 `DynGenPar::StackItem::StackItem (const QList< StackItem > & parents, CatArg cat, CatArg effCat, const PseudoCatScope & scope)` `[inline]`

type 1

Definition at line 634 of file `dyngenpar.h`.

7.29.2.4 `DynGenPar::StackItem::StackItem (const StackItem & parent, int dummy)` `[inline]`

type 2

Definition at line 640 of file `dyngenpar.h`.

7.29.2.5 `DynGenPar::StackItem::StackItem (const StackItem & parent, const Rule & rule, int len, int curr, int i, const Node & tree, const PseudoCatScope & scope, int ruleno, const NextTokenConstraints & nextTokenConstraints)` `[inline]`

type 3

Definition at line 645 of file dyngenpar.h.

7.29.2.6 `DynGenPar::StackItem::StackItem (const StackItem & parent, CatArg target, int pos, int len)` `[inline]`

type 4

Definition at line 654 of file dyngenpar.h.

7.29.2.7 `DynGenPar::StackItem::StackItem (const StackItem & parent, CatArg cat, const PseudoCatScope & scope)` `[inline]`

type 5

Definition at line 660 of file dyngenpar.h.

7.29.2.8 `DynGenPar::StackItem::StackItem (const StackItem & parent, const QList< Node > & leaves, int i, const Node & tree, const PseudoCatScope & scope, const NextTokenConstraints & nextTokenConstraints)` `[inline]`

type 6

Definition at line 666 of file dyngenpar.h.

7.29.3 Member Function Documentation

7.29.3.1 `void DynGenPar::StackItem::addParent (const StackItem & parent)` `[inline]`

Definition at line 429 of file dyngenpar.h.

7.29.3.2 `const StackItemData* DynGenPar::StackItem::data () const` `[inline]`

Definition at line 431 of file dyngenpar.h.

7.29.3.3 `void DynGenPar::StackItem::setParents (const QList< StackItem > & parents)` `[inline]`

Definition at line 430 of file dyngenpar.h.

7.29.3.4 `int DynGenPar::StackItem::type () const [inline]`

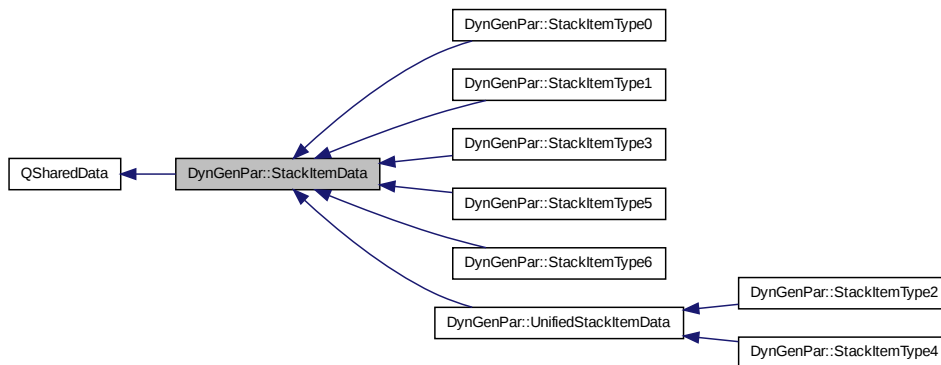
Definition at line 428 of file `dyngenpar.h`.

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

- [dyngenpar.h](#)

7.30 DynGenPar::StackItemData Class Reference

Inheritance diagram for `DynGenPar::StackItemData`:



Public Member Functions

- virtual `~StackItemData ()`
- virtual `StackItemData * clone ()=0`
- virtual `int type () const =0`
- virtual `void addParent (const StackItem &parent)=0`
- virtual `void setParents (const QList< StackItem > &parents)=0`

7.30.1 Detailed Description

Definition at line 382 of file `dyngenpar.h`.

7.30.2 Constructor & Destructor Documentation

7.30.2.1 `virtual DynGenPar::StackItemData::~~StackItemData () [inline, virtual]`

Definition at line 384 of file `dyngenpar.h`.

7.30.3 Member Function Documentation

7.30.3.1 `virtual void DynGenPar::StackItemData::addParent (const StackItem & parent) [pure virtual]`

Implemented in [DynGenPar::StackItemType0](#), [DynGenPar::StackItemType1](#), [DynGenPar::StackItemType2](#), [DynGenPar::StackItemType3](#), [DynGenPar::StackItemType4](#), [DynGenPar::StackItemType5](#), and [DynGenPar::StackItemType6](#).

7.30.3.2 `virtual StackItemData* DynGenPar::StackItemData::clone () [pure virtual]`

Implemented in [DynGenPar::StackItemType0](#), [DynGenPar::StackItemType1](#), [DynGenPar::StackItemType2](#), [DynGenPar::StackItemType3](#), [DynGenPar::StackItemType4](#), [DynGenPar::StackItemType5](#), and [DynGenPar::StackItemType6](#).

7.30.3.3 `virtual void DynGenPar::StackItemData::setParents (const QList< StackItem > & parents) [pure virtual]`

Implemented in [DynGenPar::StackItemType0](#), [DynGenPar::StackItemType1](#), [DynGenPar::StackItemType2](#), [DynGenPar::StackItemType3](#), [DynGenPar::StackItemType4](#), [DynGenPar::StackItemType5](#), and [DynGenPar::StackItemType6](#).

7.30.3.4 `virtual int DynGenPar::StackItemData::type () const [pure virtual]`

Implemented in [DynGenPar::StackItemType0](#), [DynGenPar::StackItemType1](#), [DynGenPar::StackItemType2](#), [DynGenPar::StackItemType3](#), [DynGenPar::StackItemType4](#), [DynGenPar::StackItemType5](#), and [DynGenPar::StackItemType6](#).

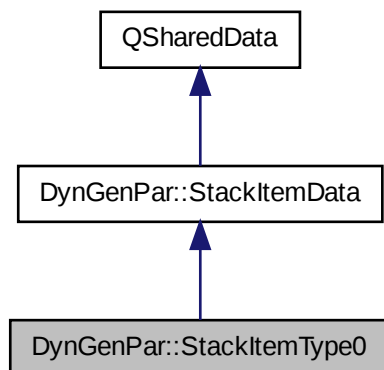
The documentation for this class was generated from the following file:

- [dyngenpar.h](#)

7.31 DynGenPar::StackItemType0 Class Reference

type 0 item: during match, we're waiting for a token to shift

Inheritance diagram for `DynGenPar::StackItemType0`:



Public Member Functions

- [StackItemType0](#) (const [QList](#)< [StackItem](#) > &parents, [CatArg](#) cat, [CatArg](#) effCat, int pos, const [PseudoCatScope](#) &scope)
- virtual [~StackItemType0](#) ()
- virtual [StackItemData](#) * [clone](#) ()
- virtual int [type](#) () const
- virtual void [addParent](#) (const [StackItem](#) &parent)
- virtual void [setParents](#) (const [QList](#)< [StackItem](#) > &parents)
- const [QList](#)< [StackItem](#) > & [parents](#) () const
- [Cat](#) cat () const
- [Cat](#) effCat () const
- int [pos](#) () const
- [PseudoCatScope](#) scope () const

7.31.1 Detailed Description

type 0 item: during match, we're waiting for a token to shift

Definition at line 437 of file `dyngenpar.h`.

7.31.2 Constructor & Destructor Documentation

7.31.2.1 `DynGenPar::StackItemType0::StackItemType0 (const QList< StackItem > & parents, CatArg cat, CatArg effCat, int pos, const PseudoCatScope & scope) [inline]`

Definition at line 439 of file `dyngenpar.h`.

7.31.2.2 `virtual DynGenPar::StackItemType0::~~StackItemType0 () [inline, virtual]`

Definition at line 443 of file `dyngenpar.h`.

7.31.3 Member Function Documentation

7.31.3.1 `virtual void DynGenPar::StackItemType0::addParent (const StackItem & parent) [inline, virtual]`

Implements [DynGenPar::StackItemData](#).

Definition at line 446 of file `dyngenpar.h`.

7.31.3.2 `Cat DynGenPar::StackItemType0::cat () const [inline]`

Definition at line 451 of file `dyngenpar.h`.

7.31.3.3 `virtual StackItemData* DynGenPar::StackItemType0::clone() [inline, virtual]`

Implements [DynGenPar::StackItemData](#).

Definition at line 444 of file `dyngenpar.h`.

7.31.3.4 `Cat DynGenPar::StackItemType0::effCat() const [inline]`

Definition at line 452 of file `dyngenpar.h`.

7.31.3.5 `const QList<StackItem>& DynGenPar::StackItemType0::parents() const [inline]`

Definition at line 450 of file `dyngenpar.h`.

7.31.3.6 `int DynGenPar::StackItemType0::pos() const [inline]`

Definition at line 453 of file `dyngenpar.h`.

7.31.3.7 `PseudoCatScope DynGenPar::StackItemType0::scope() const [inline]`

Definition at line 454 of file `dyngenpar.h`.

7.31.3.8 `virtual void DynGenPar::StackItemType0::setParents(const QList< StackItem > & parents) [inline, virtual]`

Implements [DynGenPar::StackItemData](#).

Definition at line 447 of file `dyngenpar.h`.

7.31.3.9 `virtual int DynGenPar::StackItemType0::type() const [inline, virtual]`

Implements [DynGenPar::StackItemData](#).

Definition at line 445 of file `dyngenpar.h`.

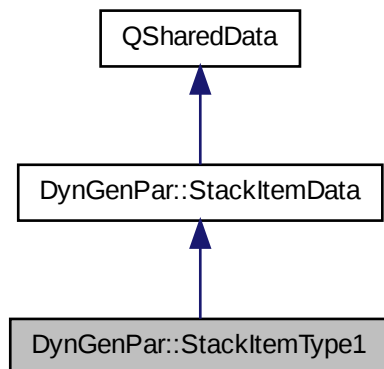
The documentation for this class was generated from the following file:

- [dyngenpar.h](#)

7.32 DynGenPar::StackItemType1 Class Reference

type 1 item: during type 0 item processing, we're executing a reduce

Inheritance diagram for DynGenPar::StackItemType1:



Public Member Functions

- [StackItemType1](#) (const [QList](#)< [StackItem](#) > &parents, [CatArg](#) cat, [CatArg](#) effCat, const [PseudoCatScope](#) &scope)
- virtual [~StackItemType1](#) ()
- virtual [StackItemData](#) * [clone](#) ()
- virtual int [type](#) () const
- virtual void [addParent](#) (const [StackItem](#) &parent)
- virtual void [setParents](#) (const [QList](#)< [StackItem](#) > &parents)
- const [QList](#)< [StackItem](#) > & [parents](#) () const
- [Cat](#) cat () const
- [Cat](#) effCat () const
- [PseudoCatScope](#) scope () const

7.32.1 Detailed Description

type 1 item: during type 0 item processing, we're executing a reduce

Definition at line 463 of file dyngenpar.h.

7.32.2 Constructor & Destructor Documentation

7.32.2.1 `DynGenPar::StackItemType1::StackItemType1 (const QList< StackItem > & parents, CatArg cat, CatArg effCat, const PseudoCatScope & scope) [inline]`

Definition at line 465 of file dyngenpar.h.

7.32.2.2 `virtual DynGenPar::StackItemType1::~~StackItemType1 () [inline, virtual]`

Definition at line 468 of file dyngenpar.h.

7.32.3 Member Function Documentation

7.32.3.1 `virtual void DynGenPar::StackItemType1::addParent (const StackItem & parent) [inline, virtual]`

Implements [DynGenPar::StackItemData](#).

Definition at line 471 of file dyngenpar.h.

7.32.3.2 `Cat DynGenPar::StackItemType1::cat () const [inline]`

Definition at line 476 of file dyngenpar.h.

7.32.3.3 `virtual StackItemData* DynGenPar::StackItemType1::clone () [inline, virtual]`

Implements [DynGenPar::StackItemData](#).

Definition at line 469 of file dyngenpar.h.

7.32.3.4 `Cat DynGenPar::StackItemType1::effCat () const [inline]`

Definition at line 477 of file dyngenpar.h.

7.32.3.5 `const QList< StackItem > & DynGenPar::StackItemType1::parents () const [inline]`

Definition at line 475 of file dyngenpar.h.

7.32.3.6 `PseudoCatScope DynGenPar::StackItemType1::scope () const [inline]`

Definition at line 478 of file dyngenpar.h.

7.32.3.7 `virtual void DynGenPar::StackItemType1::setParents (const QList< StackItem > & parents) [inline, virtual]`

Implements [DynGenPar::StackItemData](#).

Definition at line 472 of file `dyngenpar.h`.

7.32.3.8 `virtual int DynGenPar::StackItemType1::type () const [inline, virtual]`

Implements [DynGenPar::StackItemData](#).

Definition at line 470 of file `dyngenpar.h`.

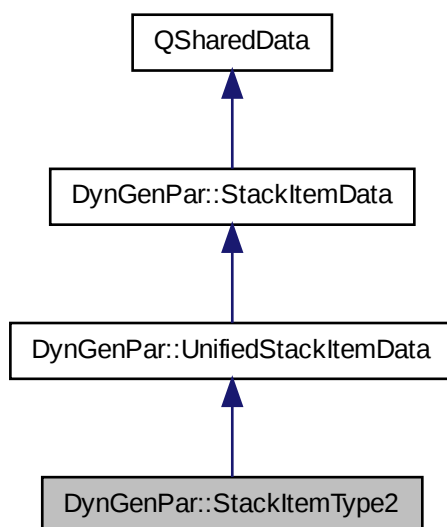
The documentation for this class was generated from the following file:

- [dyngenpar.h](#)

7.33 DynGenPar::StackItemType2 Class Reference

type 2 item: during reduce, we're recursively executing another reduce

Inheritance diagram for `DynGenPar::StackItemType2`:



Public Member Functions

- [StackItemType2](#) (const [StackItem](#) &parent)
- virtual [~StackItemType2](#) ()
- virtual [StackItemData](#) * [clone](#) ()
- virtual int [type](#) () const
- virtual void [addParent](#) (const [StackItem](#) &)
- virtual void [setParents](#) (const [QList](#)< [StackItem](#) > &)
- const [StackItem](#) & [parent](#) () const

7.33.1 Detailed Description

type 2 item: during reduce, we're recursively executing another reduce

This item forms a unification point.

Definition at line 487 of file dyngenpar.h.

7.33.2 Constructor & Destructor Documentation

7.33.2.1 `DynGenPar::StackItemType2::StackItemType2 (const StackItem & parent) [inline]`

Definition at line 489 of file dyngenpar.h.

7.33.2.2 `virtual DynGenPar::StackItemType2::~~StackItemType2 () [inline, virtual]`

Definition at line 490 of file dyngenpar.h.

7.33.3 Member Function Documentation

7.33.3.1 `virtual void DynGenPar::StackItemType2::addParent (const StackItem &) [inline, virtual]`

Implements [DynGenPar::StackItemData](#).

Definition at line 493 of file dyngenpar.h.

7.33.3.2 `virtual StackItemData* DynGenPar::StackItemType2::clone () [inline, virtual]`

Implements [DynGenPar::StackItemData](#).

Definition at line 491 of file dyngenpar.h.

7.33.3.3 `const StackItem& DynGenPar::StackItemType2::parent () const` [inline]

Definition at line 499 of file dyngenpar.h.

7.33.3.4 `virtual void DynGenPar::StackItemType2::setParents (const QList< StackItem > &)` [inline, virtual]

Implements [DynGenPar::StackItemData](#).

Definition at line 496 of file dyngenpar.h.

7.33.3.5 `virtual int DynGenPar::StackItemType2::type () const` [inline, virtual]

Implements [DynGenPar::StackItemData](#).

Definition at line 492 of file dyngenpar.h.

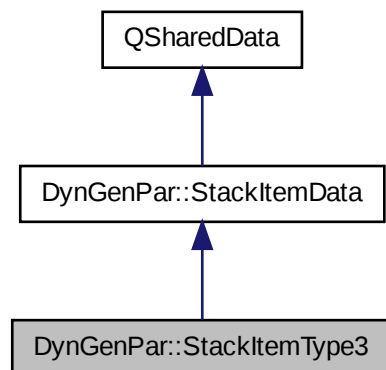
The documentation for this class was generated from the following file:

- [dyngenpar.h](#)

7.34 DynGenPar::StackItemType3 Class Reference

type 3 item: during matchRemaining, we're executing a match

Inheritance diagram for DynGenPar::StackItemType3:



Public Member Functions

- [StackItemType3](#) (const [StackItem](#) &parent, const [Rule](#) &rule, int len, int curr, int i, const [Node](#) &tree, const [PseudoCatScope](#) &scope, int ruleno, const [NextTokenConstraints](#) &nextTokenConstraints)
- virtual [~StackItemType3](#) ()
- virtual [StackItemData](#) * [clone](#) ()
- virtual int [type](#) () const
- virtual void [addParent](#) (const [StackItem](#) &)
- virtual void [setParents](#) (const [QList](#)< [StackItem](#) > &)
- const [StackItem](#) & [parent](#) () const
- [Rule](#) [rule](#) () const
- int [len](#) () const
- int [curr](#) () const
- int [i](#) () const
- [Node](#) [tree](#) () const
- [PseudoCatScope](#) [scope](#) () const
- int [ruleno](#) () const
- [NextTokenConstraints](#) [nextTokenConstraints](#) () const

7.34.1 Detailed Description

type 3 item: during matchRemaining, we're executing a match

Definition at line 505 of file dyngenpar.h.

7.34.2 Constructor & Destructor Documentation

7.34.2.1 `DynGenPar::StackItemType3::StackItemType3 (const StackItem & parent, const Rule & rule, int len, int curr, int i, const Node & tree, const PseudoCatScope & scope, int ruleno, const NextTokenConstraints & nextTokenConstraints) [inline]`

Definition at line 507 of file dyngenpar.h.

7.34.2.2 `virtual DynGenPar::StackItemType3::~~StackItemType3 () [inline, virtual]`

Definition at line 513 of file dyngenpar.h.

7.34.3 Member Function Documentation

7.34.3.1 `virtual void DynGenPar::StackItemType3::addParent (const StackItem &) [inline, virtual]`

Implements [DynGenPar::StackItemData](#).

Definition at line 516 of file dyngenpar.h.

7.34.3.2 `virtual StackItemData* DynGenPar::StackItemType3::clone ()` `[inline, virtual]`

Implements [DynGenPar::StackItemData](#).

Definition at line 514 of file `dyngenpar.h`.

7.34.3.3 `int DynGenPar::StackItemType3::curr () const` `[inline]`

Definition at line 525 of file `dyngenpar.h`.

7.34.3.4 `int DynGenPar::StackItemType3::i () const` `[inline]`

Definition at line 526 of file `dyngenpar.h`.

7.34.3.5 `int DynGenPar::StackItemType3::len () const` `[inline]`

Definition at line 524 of file `dyngenpar.h`.

7.34.3.6 `NextTokenConstraints DynGenPar::StackItemType3::nextTokenConstraints () const` `[inline]`

Definition at line 530 of file `dyngenpar.h`.

7.34.3.7 `const StackItem& DynGenPar::StackItemType3::parent () const` `[inline]`

Definition at line 522 of file `dyngenpar.h`.

7.34.3.8 `Rule DynGenPar::StackItemType3::rule () const` `[inline]`

Definition at line 523 of file `dyngenpar.h`.

7.34.3.9 `int DynGenPar::StackItemType3::ruleno () const` `[inline]`

Definition at line 529 of file `dyngenpar.h`.

7.34.3.10 `PseudoCatScope DynGenPar::StackItemType3::scope () const` `[inline]`

Definition at line 528 of file `dyngenpar.h`.

7.34.3.11 `virtual void DynGenPar::StackItemType3::setParents (const QList< StackItem > &) [inline, virtual]`

Implements [DynGenPar::StackItemData](#).

Definition at line 519 of file `dyngenpar.h`.

7.34.3.12 `Node DynGenPar::StackItemType3::tree () const [inline]`

Definition at line 527 of file `dyngenpar.h`.

7.34.3.13 `virtual int DynGenPar::StackItemType3::type () const [inline, virtual]`

Implements [DynGenPar::StackItemData](#).

Definition at line 515 of file `dyngenpar.h`.

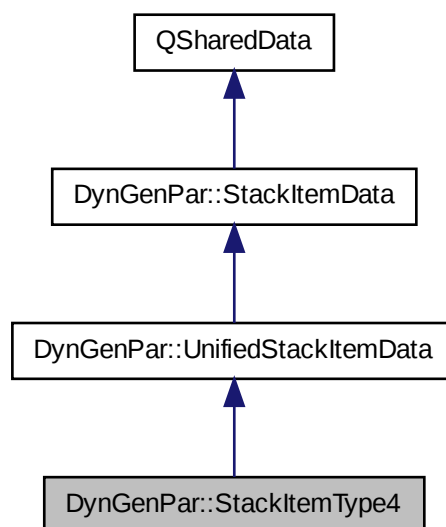
The documentation for this class was generated from the following file:

- [dyngenpar.h](#)

7.35 DynGenPar::StackItemType4 Class Reference

type 4 item: during reduce, we're executing a `matchRemaining`

Inheritance diagram for `DynGenPar::StackItemType4`:



Public Member Functions

- [StackItemType4](#) (const [StackItem](#) &parent, [CatArg](#) target, int pos, int len)
- virtual [~StackItemType4](#) ()
- virtual [StackItemData](#) * [clone](#) ()
- virtual int [type](#) () const
- virtual void [addParent](#) (const [StackItem](#) &)
- virtual void [setParents](#) (const [QList](#)< [StackItem](#) > &)
- const [StackItem](#) & [parent](#) () const
- [Cat](#) [target](#) () const
- int [pos](#) () const
- int [len](#) () const

7.35.1 Detailed Description

type 4 item: during reduce, we're executing a matchRemaining

This item forms a unification point.

Definition at line 545 of file dyngenpar.h.

7.35.2 Constructor & Destructor Documentation

7.35.2.1 `DynGenPar::StackItemType4::StackItemType4 (const StackItem & parent, CatArg target, int pos, int len)`
`[inline]`

Definition at line 547 of file dyngenpar.h.

7.35.2.2 `virtual DynGenPar::StackItemType4::~~StackItemType4 ()` `[inline, virtual]`

Definition at line 549 of file dyngenpar.h.

7.35.3 Member Function Documentation

7.35.3.1 `virtual void DynGenPar::StackItemType4::addParent (const StackItem &)` `[inline, virtual]`

Implements [DynGenPar::StackItemData](#).

Definition at line 552 of file dyngenpar.h.

7.35.3.2 `virtual StackItemData* DynGenPar::StackItemType4::clone ()` `[inline, virtual]`

Implements [DynGenPar::StackItemData](#).

Definition at line 550 of file dyngenpar.h.

7.35.3.3 `int DynGenPar::StackItemType4::len () const [inline]`

Definition at line 561 of file `dyngenpar.h`.

7.35.3.4 `const StackItem& DynGenPar::StackItemType4::parent () const [inline]`

Definition at line 558 of file `dyngenpar.h`.

7.35.3.5 `int DynGenPar::StackItemType4::pos () const [inline]`

Definition at line 560 of file `dyngenpar.h`.

7.35.3.6 `virtual void DynGenPar::StackItemType4::setParents (const QList< StackItem > &) [inline, virtual]`

Implements [DynGenPar::StackItemData](#).

Definition at line 555 of file `dyngenpar.h`.

7.35.3.7 `const DynGenPar::StackItemType4::target () const [inline]`

Definition at line 559 of file `dyngenpar.h`.

7.35.3.8 `virtual int DynGenPar::StackItemType4::type () const [inline, virtual]`

Implements [DynGenPar::StackItemData](#).

Definition at line 551 of file `dyngenpar.h`.

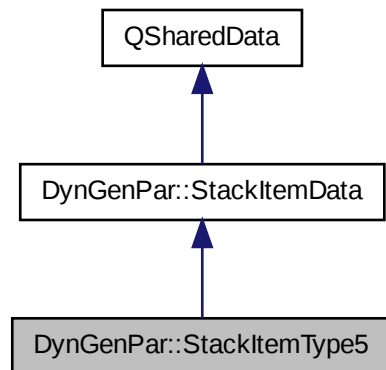
The documentation for this class was generated from the following file:

- [dyngenpar.h](#)

7.36 DynGenPar::StackItemType5 Class Reference

type 5 item: during match (of an MCFG constraint), we're executing a matchRemaining

Inheritance diagram for DynGenPar::StackItemType5:



Public Member Functions

- [StackItemType5](#) (const [StackItem](#) &parent, [CatArg](#) cat, const [PseudoCatScope](#) &scope)
- virtual [~StackItemType5](#) ()
- virtual [StackItemData](#) * [clone](#) ()
- virtual int [type](#) () const
- virtual void [addParent](#) (const [StackItem](#) &)
- virtual void [setParents](#) (const [QList](#)< [StackItem](#) > &)
- const [StackItem](#) & [parent](#) () const
- [Cat](#) cat () const
- [PseudoCatScope](#) scope () const

7.36.1 Detailed Description

type 5 item: during match (of an MCFG constraint), we're executing a matchRemaining

Definition at line 570 of file dyngenpar.h.

7.36.2 Constructor & Destructor Documentation

7.36.2.1 `DynGenPar::StackItemType5::StackItemType5 (const StackItem & parent, CatArg cat, const PseudoCatScope & scope) [inline]`

Definition at line 572 of file dyngenpar.h.

7.36.2.2 virtual `DynGenPar::StackItemType5::~~StackItemType5 ()` [`inline`, `virtual`]

Definition at line 575 of file `dyngenpar.h`.

7.36.3 Member Function Documentation

7.36.3.1 virtual void `DynGenPar::StackItemType5::addParent (const StackItem &)` [`inline`, `virtual`]

Implements [DynGenPar::StackItemData](#).

Definition at line 578 of file `dyngenpar.h`.

7.36.3.2 Cat `DynGenPar::StackItemType5::cat () const` [`inline`]

Definition at line 585 of file `dyngenpar.h`.

7.36.3.3 virtual `StackItemData* DynGenPar::StackItemType5::clone ()` [`inline`, `virtual`]

Implements [DynGenPar::StackItemData](#).

Definition at line 576 of file `dyngenpar.h`.

7.36.3.4 const `StackItem& DynGenPar::StackItemType5::parent () const` [`inline`]

Definition at line 584 of file `dyngenpar.h`.

7.36.3.5 PseudoCatScope `DynGenPar::StackItemType5::scope () const` [`inline`]

Definition at line 586 of file `dyngenpar.h`.

7.36.3.6 virtual void `DynGenPar::StackItemType5::setParents (const QList< StackItem > &)` [`inline`, `virtual`]

Implements [DynGenPar::StackItemData](#).

Definition at line 581 of file `dyngenpar.h`.

7.36.3.7 virtual int `DynGenPar::StackItemType5::type () const` [`inline`, `virtual`]

Implements [DynGenPar::StackItemData](#).

Definition at line 577 of file `dyngenpar.h`.

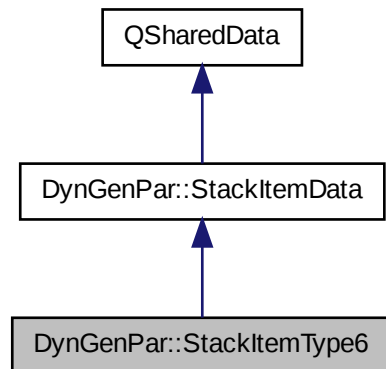
The documentation for this class was generated from the following file:

- [dyngenpar.h](#)

7.37 DynGenPar::StackItemType6 Class Reference

type 6 item: during match, we're matching a P constraint

Inheritance diagram for DynGenPar::StackItemType6:



Public Member Functions

- [StackItemType6](#) (const [StackItem](#) &parent, const [QList](#)< [Node](#) > &leaves, int i, const [Node](#) &tree, const [PseudoCatScope](#) &scope, const [NextTokenConstraints](#) &nextTokenConstraints)
- virtual [~StackItemType6](#) ()
- virtual [StackItemData](#) * [clone](#) ()
- virtual int [type](#) () const
- virtual void [addParent](#) (const [StackItem](#) &)
- virtual void [setParents](#) (const [QList](#)< [StackItem](#) > &)
- const [StackItem](#) & [parent](#) () const
- [QList](#)< [Node](#) > [leaves](#) () const
- int [i](#) () const
- [Node](#) [tree](#) () const
- [PseudoCatScope](#) [scope](#) () const
- [NextTokenConstraints](#) [nextTokenConstraints](#) () const

7.37.1 Detailed Description

type 6 item: during match, we're matching a P constraint

Definition at line 594 of file dyngenpar.h.

7.37.2 Constructor & Destructor Documentation

7.37.2.1 `DynGenPar::StackItemType6::StackItemType6 (const StackItem & parent, const QList< Node > & leaves, int i, const Node & tree, const PseudoCatScope & scope, const NextTokenConstraints & nextTokenConstraints)` `[inline]`

Definition at line 596 of file dyngenpar.h.

7.37.2.2 `virtual DynGenPar::StackItemType6::~~StackItemType6 ()` `[inline, virtual]`

Definition at line 601 of file dyngenpar.h.

7.37.3 Member Function Documentation

7.37.3.1 `virtual void DynGenPar::StackItemType6::addParent (const StackItem &)` `[inline, virtual]`

Implements [DynGenPar::StackItemData](#).

Definition at line 604 of file dyngenpar.h.

7.37.3.2 `virtual StackItemData* DynGenPar::StackItemType6::clone ()` `[inline, virtual]`

Implements [DynGenPar::StackItemData](#).

Definition at line 602 of file dyngenpar.h.

7.37.3.3 `int DynGenPar::StackItemType6::i () const` `[inline]`

Definition at line 612 of file dyngenpar.h.

7.37.3.4 `QList<Node> DynGenPar::StackItemType6::leaves () const` `[inline]`

Definition at line 611 of file dyngenpar.h.

7.37.3.5 `NextTokenConstraints DynGenPar::StackItemType6::nextTokenConstraints () const` `[inline]`

Definition at line 615 of file dyngenpar.h.

7.37.3.6 `const StackItem& DynGenPar::StackItemType6::parent () const` `[inline]`

Definition at line 610 of file dyngenpar.h.

7.37.3.7 PseudoCatScope DynGenPar::StackItemType6::scope () const [inline]

Definition at line 614 of file dyngenpar.h.

7.37.3.8 virtual void DynGenPar::StackItemType6::setParents (const QList< StackItem > &) [inline, virtual]

Implements [DynGenPar::StackItemData](#).

Definition at line 607 of file dyngenpar.h.

7.37.3.9 Node DynGenPar::StackItemType6::tree () const [inline]

Definition at line 613 of file dyngenpar.h.

7.37.3.10 virtual int DynGenPar::StackItemType6::type () const [inline, virtual]

Implements [DynGenPar::StackItemData](#).

Definition at line 603 of file dyngenpar.h.

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

- [dyngenpar.h](#)

7.38 DynGenPar::Term Struct Reference

term in the expression of a component of a PMCFG function

Public Member Functions

- [Term](#) ()
dummy default constructor for bindings
- [Term](#) (int a, int c)
- [Term](#) (CatArg t)
- bool [isComponent](#) () const
- bool [isToken](#) () const
- bool [operator==](#) (const [Term](#) &other) const
needed for QList

Public Attributes

- int [arg](#)
- int [component](#)
- [Cat](#) token

7.38.1 Detailed Description

term in the expression of a component of a PMCFG function

Definition at line 817 of file dyngenpar.h.

7.38.2 Constructor & Destructor Documentation

7.38.2.1 DynGenPar::Term::Term () [inline]

dummy default constructor for bindings

Definition at line 819 of file dyngenpar.h.

7.38.2.2 DynGenPar::Term::Term (int *a*, int *c*) [inline]

Definition at line 820 of file dyngenpar.h.

7.38.2.3 DynGenPar::Term::Term (CatArg *t*) [inline]

Definition at line 821 of file dyngenpar.h.

7.38.3 Member Function Documentation

7.38.3.1 bool DynGenPar::Term::isComponent () const [inline]

Definition at line 827 of file dyngenpar.h.

7.38.3.2 bool DynGenPar::Term::isToken () const [inline]

Definition at line 828 of file dyngenpar.h.

7.38.3.3 bool DynGenPar::Term::operator==(const Term & *other*) const [inline]

needed for [QList](#)

Definition at line 830 of file dyngenpar.h.

7.38.4 Member Data Documentation

7.38.4.1 int DynGenPar::Term::arg

Definition at line 822 of file dyngenpar.h.

7.38.4.2 int DynGenPar::Term::component

Definition at line 822 of file dyngenpar.h.

7.38.4.3 Cat DynGenPar::Term::token

This is supposed to be a token in a standard PMCFG, though in the implementation, any context-free category will work.

Definition at line 826 of file dyngenpar.h.

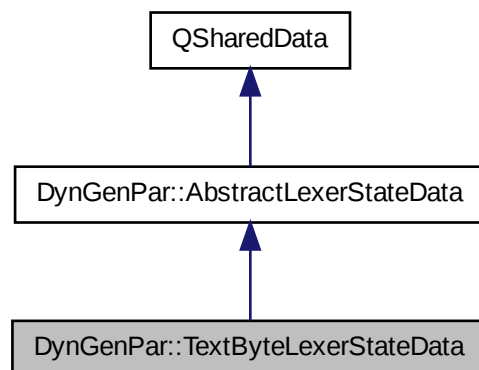
The documentation for this struct was generated from the following file:

- [dyngenpar.h](#)

7.39 DynGenPar::TextByteLexerStateData Class Reference

You should not have to use this class directly, ever.

Inheritance diagram for DynGenPar::TextByteLexerStateData:



Public Member Functions

- `TextByteLexerStateData` (qint64 streamPosition, `TextPosition` textPosition)
- virtual `AbstractLexerStateData * clone ()`

Public Attributes

- qint64 [streamPos](#)
- [TextPosition](#) *textPos*

7.39.1 Detailed Description

You should not have to use this class directly, ever.

[TextByteTokenSource](#) needs lexer states to store the true position, since our token positions don't count the stripped CRs.

This also stores the text position, use [TextByteTokenSource::textPosition](#) to retrieve it.

Definition at line 110 of file `bytetokensource.h`.

7.39.2 Constructor & Destructor Documentation

7.39.2.1 `DynGenPar::TextByteLexerStateData::TextByteLexerStateData (qint64 streamPosition, TextPosition textPosition)`
[`inline`]

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

7.39.3 Member Function Documentation

7.39.3.1 `virtual AbstractLexerStateData* DynGenPar::TextByteLexerStateData::clone ()` [`inline`, `virtual`]

Implements [DynGenPar::AbstractLexerStateData](#).

Definition at line 115 of file `bytetokensource.h`.

7.39.4 Member Data Documentation

7.39.4.1 `qint64 DynGenPar::TextByteLexerStateData::streamPos`

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

7.39.4.2 `TextPosition DynGenPar::TextByteLexerStateData::textPos`

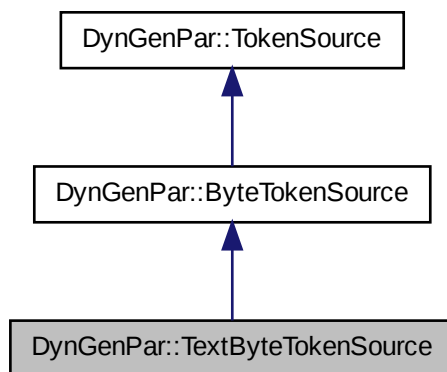
Definition at line 119 of file `bytetokensource.h`.

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

- [bytetokensource.h](#)

7.40 DynGenPar::TextByteTokenSource Class Reference

Inheritance diagram for DynGenPar::TextByteTokenSource:



Public Member Functions

- [TextByteTokenSource](#) ()
- [TextByteTokenSource](#) (const QString &fileName)
- virtual [~TextByteTokenSource](#) ()
- virtual bool [rewindTo](#) (int pos, const [LexerState](#) &lexerState=[LexerState](#)())
We can only rewind if we have a lexer state with the true position.
- virtual [LexerState](#) [saveState](#) ()
Saves the true stream position (including CRs) and the text position in lines and columns into a lexer state.

Static Public Member Functions

- static [TextPosition](#) [textPosition](#) (const [LexerState](#) &lexerState)
Retrieves the text position (line and column) stored in the lexer state.

Protected Member Functions

- virtual [Cat](#) [readToken](#) ()
Overrides readToken to strip CRs.
- virtual void [reset](#) ()

7.40.1 Detailed Description

Definition at line 122 of file `bytetokensource.h`.

7.40.2 Constructor & Destructor Documentation

7.40.2.1 `DynGenPar::TextByteTokenSource::TextByteTokenSource ()` [inline]

Definition at line 124 of file `bytetokensource.h`.

7.40.2.2 `DynGenPar::TextByteTokenSource::TextByteTokenSource (const QString & fileName)` [inline]

Definition at line 125 of file `bytetokensource.h`.

7.40.2.3 `virtual DynGenPar::TextByteTokenSource::~~TextByteTokenSource ()` [inline, virtual]

Definition at line 126 of file `bytetokensource.h`.

7.40.3 Member Function Documentation

7.40.3.1 `virtual Cat DynGenPar::TextByteTokenSource::readToken ()` [inline, protected, virtual]

Overrides `readToken` to strip CRs.

Reimplemented from [DynGenPar::ByteTokenSource](#).

Definition at line 164 of file `bytetokensource.h`.

7.40.3.2 `virtual void DynGenPar::TextByteTokenSource::reset ()` [inline, protected, virtual]

Reimplemented from [DynGenPar::ByteTokenSource](#).

Definition at line 173 of file `bytetokensource.h`.

7.40.3.3 `virtual bool DynGenPar::TextByteTokenSource::rewindTo (int pos, const LexerState & lexerState = LexerState ())`
[inline, virtual]

We can only rewind if we have a lexer state with the true position.

Reimplemented from [DynGenPar::ByteTokenSource](#).

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

7.40.3.4 `virtual LexerState DynGenPar::TextByteTokenSource::saveState ()` [inline, virtual]

Saves the true stream position (including CRs) and the text position in lines and columns into a lexer state.

Reimplemented from [DynGenPar::TokenSource](#).

Definition at line 151 of file `bytetokensource.h`.

7.40.3.5 `static TextPosition DynGenPar::TextByteTokenSource::textPosition (const LexerState & lexerState) [inline, static]`

Retrieves the text position (line and column) stored in the lexer state.

Definition at line 157 of file `bytetokensource.h`.

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

- [bytetokensource.h](#)

7.41 DynGenPar::TextPosition Struct Reference

text position

Public Member Functions

- [TextPosition](#) ()
- [TextPosition](#) (int l, int c)
- void [reset](#) ()
- void [countCharacter](#) (unsigned char c)
convenience method to count a character

Public Attributes

- int [line](#)
line, zero-based
- int [col](#)
column, zero-based

7.41.1 Detailed Description

text position

stored in the lexer state by some text-oriented token sources to allow correct error reporting

Definition at line 792 of file `dyngenpar.h`.

7.41.2 Constructor & Destructor Documentation

7.41.2.1 `DynGenPar::TextPosition::TextPosition () [inline]`

Definition at line 793 of file `dyngenpar.h`.

7.41.2.2 DynGenPar::TextPosition::TextPosition (int *l*, int *c*) [inline]

Definition at line 794 of file dyngenpar.h.

7.41.3 Member Function Documentation

7.41.3.1 void DynGenPar::TextPosition::countCharacter (unsigned char *c*) [inline]

convenience method to count a character

Definition at line 799 of file dyngenpar.h.

7.41.3.2 void DynGenPar::TextPosition::reset () [inline]

Definition at line 796 of file dyngenpar.h.

7.41.4 Member Data Documentation

7.41.4.1 int DynGenPar::TextPosition::col

column, zero-based

Definition at line 810 of file dyngenpar.h.

7.41.4.2 int DynGenPar::TextPosition::line

line, zero-based

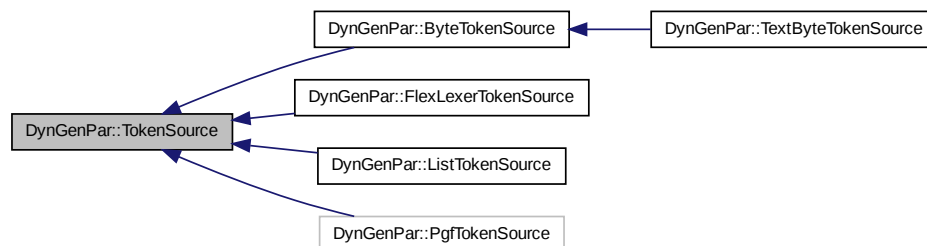
Definition at line 809 of file dyngenpar.h.

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

- [dyngenpar.h](#)

7.42 DynGenPar::TokenSource Class Reference

Inheritance diagram for DynGenPar::TokenSource:



Public Member Functions

- [TokenSource](#) ()
- virtual [~TokenSource](#) ()
- [Cat nextToken](#) ()
get the next token from the input, increment current position, save parse tree
- [Node parseTree](#) ()
get the parse tree for the last shifted token
- virtual bool [matchParseTree](#) (const [Node](#) &treeToMatch)
match the parse tree for the last shifted token against the given tree
- int [currentPosition](#) ()
get the current input position
- virtual bool [rewindTo](#) (int pos, const [LexerState](#) &=[LexerState](#)())
rewind to an older position (requires buffering)
- virtual [LexerState saveState](#) ()
saves the current lexer state, by default a null [LexerState](#)

Protected Member Functions

- virtual [Cat readToken](#) ()=0
get the next token from the input, to be implemented by subclasses
- bool [simpleRewind](#) (int pos)
basic implementation of [rewindTo](#) for subclasses which support it

Protected Attributes

- int [currPos](#)
- [Node tree](#)
sub-parse-tree for hierarchical parsing

7.42.1 Detailed Description

Definition at line 696 of file `dyngenpar.h`.

7.42.2 Constructor & Destructor Documentation

7.42.2.1 `DynGenPar::TokenSource::TokenSource ()` [`inline`]

Definition at line 698 of file `dyngenpar.h`.

7.42.2.2 `virtual DynGenPar::TokenSource::~~TokenSource ()` [`inline`, `virtual`]

Definition at line 699 of file `dyngenpar.h`.

7.42.3 Member Function Documentation

7.42.3.1 `int DynGenPar::TokenSource::currentPosition () [inline]`

get the current input position

Definition at line 732 of file `dyngenpar.h`.

7.42.3.2 `virtual bool DynGenPar::TokenSource::matchParseTree (const Node & treeToMatch) [inline, virtual]`

match the parse tree for the last shifted token against the given tree

Returns

`true` if they should be considered identical for the purposes of a PMCFG rule using the same variable twice,
`false` otherwise

The default implementation compares only the category. Other token sources may want to also look at the data and children fields.

Definition at line 728 of file `dyngenpar.h`.

7.42.3.3 `Cat DynGenPar::TokenSource::nextToken () [inline]`

get the next token from the input, increment current position, save parse tree

An epsilon token is returned and `currPos` is not incremented if the end of input was reached.

Definition at line 705 of file `dyngenpar.h`.

7.42.3.4 `Node DynGenPar::TokenSource::parseTree () [inline]`

get the parse tree for the last shifted token

Definition at line 717 of file `dyngenpar.h`.

7.42.3.5 `virtual Cat DynGenPar::TokenSource::readToken () [protected, pure virtual]`

get the next token from the input, to be implemented by subclasses

Implemented in [DynGenPar::ByteTokenSource](#), [DynGenPar::TextByteTokenSource](#), [DynGenPar::ListTokenSource](#), and [DynGenPar::FlexLexerTokenSource](#).

7.42.3.6 `virtual bool DynGenPar::TokenSource::rewindTo (int pos, const LexerState & =LexerState()) [inline, virtual]`

rewind to an older position (requires buffering)

Returns

`true` if successful, `false` otherwise

in all cases, destroys the saved parse tree

By default, only succeeds if the position is the current one, otherwise always returns `false`. Can be overridden by subclasses.

Reimplemented in [DynGenPar::ByteTokenSource](#), [DynGenPar::TextByteTokenSource](#), and [DynGenPar::ListTokenSource](#).

Definition at line 741 of file `dyngenpar.h`.

7.42.3.7 `virtual LexerState DynGenPar::TokenSource::saveState () [inline, virtual]`

saves the current lexer state, by default a null [LexerState](#)

Reimplemented in [DynGenPar::TextByteTokenSource](#).

Definition at line 746 of file `dyngenpar.h`.

7.42.3.8 `bool DynGenPar::TokenSource::simpleRewind (int pos) [inline, protected]`

basic implementation of `rewindTo` for subclasses which support it

some subclasses may need additional processing

Definition at line 752 of file `dyngenpar.h`.

7.42.4 Member Data Documentation

7.42.4.1 `int DynGenPar::TokenSource::currPos [protected]`

Definition at line 759 of file `dyngenpar.h`.

7.42.4.2 Node DynGenPar::TokenSource::tree [protected]

sub-parse-tree for hierarchical parsing

This variable can be set by [readToken](#) to a subtree produced by a hierarchical parser, which will be attached to the resulting parse tree in lieu of a leaf node.

If this variable is left unset, a leaf node is automatically produced.

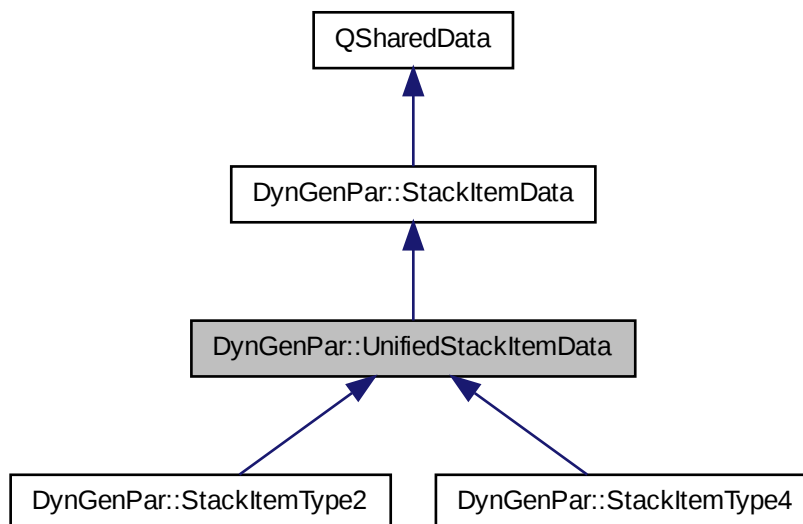
Definition at line 767 of file dyngenpar.h.

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

- [dyngenpar.h](#)

7.43 DynGenPar::UnifiedStackItemData Class Reference

Inheritance diagram for DynGenPar::UnifiedStackItemData:



Public Member Functions

- [UnifiedStackItemData \(\)](#)
- [virtual ~UnifiedStackItemData \(\)](#)

reference-counting functions

These must be const (and thus the private `usageCount` mutable) because calling them MUST NOT unshare the stack item, which would defeat the whole purpose of the unification feature.

- unsigned [refUsage](#) () const
- unsigned [derefUsage](#) () const

7.43.1 Detailed Description

Definition at line 391 of file `dyngenpar.h`.

7.43.2 Constructor & Destructor Documentation

7.43.2.1 `DynGenPar::UnifiedStackItemData::UnifiedStackItemData ()` [inline]

Definition at line 393 of file `dyngenpar.h`.

7.43.2.2 `virtual DynGenPar::UnifiedStackItemData::~~UnifiedStackItemData ()` [inline, virtual]

Definition at line 394 of file `dyngenpar.h`.

7.43.3 Member Function Documentation

7.43.3.1 `unsigned DynGenPar::UnifiedStackItemData::derefUsage () const` [inline]

Definition at line 403 of file `dyngenpar.h`.

7.43.3.2 `unsigned DynGenPar::UnifiedStackItemData::refUsage () const` [inline]

Definition at line 402 of file `dyngenpar.h`.

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

- [dyngenpar.h](#)

Chapter 8

File Documentation

8.1 bytetokensource.h File Reference

Classes

- class [DynGenPar::ByteTokenSource](#)
- class [DynGenPar::TextByteLexerStateData](#)
You should not have to use this class directly, ever.
- class [DynGenPar::TextByteTokenSource](#)

Namespaces

- namespace [DynGenPar](#)

Enumerations

- enum [ByteTokens](#) { [ByteTokenEpsilon](#) = 0, [ByteTokenNul](#) = 256, [ByteTokenError](#) = 257 }

8.1.1 Enumeration Type Documentation

8.1.1.1 enum [ByteTokens](#)

Enumerator:

ByteTokenEpsilon

ByteTokenNul we have to remap this because 0 is epsilon

ByteTokenError

Definition at line 33 of file bytetokensource.h.

8.2 dyngenpar.cpp File Reference

Namespaces

- namespace [DynGenPar](#)

Functions

- uint [qHash](#) (const [QList](#)< [DynGenPar::Cat](#) > &list)
simple hash function for lists of categories
- uint [DynGenPar::qHash](#) (const [NextTokenConstraints](#) &nextTokenConstraints)
simple hash function for next token constraints
- Node [DynGenPar::parseTreeToPmcfgsyntaxTree](#) (const Node &parseTree)
converts a parse tree obtained from a PMCFG to a PMCFG syntax tree

8.2.1 Function Documentation

8.2.1.1 uint qHash (const [QList](#)< [DynGenPar::Cat](#) > & list)

simple hash function for lists of categories

Definition at line 431 of file dyngenpar.cpp.

8.3 dyngenpar.h File Reference

Classes

- struct [DynGenPar::NextTokenConstraints](#)
rule constraints affecting the next token, for scannerless parsing
- class [DynGenPar::Rule](#)
- struct [DynGenPar::Cfg](#)
An object representing a CFG (or a PMCFG in our internal representation)
- struct [DynGenPar::MultiPrediction](#)
multi-token predictions
- struct [DynGenPar::ConstrainedMultiPrediction](#)
multi-token predictions with next token constraints
- struct [DynGenPar::FullRule](#)
full rule as found in the initial graph
- class [DynGenPar::Alternative](#)
- struct [DynGenPar::Node](#)
node in the parse tree
- class [DynGenPar::PseudoCatScopeData](#)
- class [DynGenPar::PseudoCatScope](#)

- struct `DynGenPar::Match`
(possibly partial) match
- struct `DynGenPar::ActionInfo`
data passed to parser actions
- class `DynGenPar::Action`
interface for parser actions
- class `DynGenPar::StackItemData`
- class `DynGenPar::UnifiedStackItemData`
- class `DynGenPar::StackItem`
- class `DynGenPar::StackItemType0`
type 0 item: during match, we're waiting for a token to shift
- class `DynGenPar::StackItemType1`
type 1 item: during type 0 item processing, we're executing a reduce
- class `DynGenPar::StackItemType2`
type 2 item: during reduce, we're recursively executing another reduce
- class `DynGenPar::StackItemType3`
type 3 item: during matchRemaining, we're executing a match
- class `DynGenPar::StackItemType4`
type 4 item: during reduce, we're executing a matchRemaining
- class `DynGenPar::StackItemType5`
type 5 item: during match (of an MCFG constraint), we're executing a matchRemaining
- class `DynGenPar::StackItemType6`
type 6 item: during match, we're matching a P constraint
- class `DynGenPar::AbstractLexerStateData`
API for stateful lexers to save their state for rewinding.
- class `DynGenPar::LexerState`
- class `DynGenPar::TokenSource`
- class `DynGenPar::ListTokenSource`
- struct `DynGenPar::TextPosition`
text position
- struct `DynGenPar::Term`
term in the expression of a component of a PMCFG function
- class `DynGenPar::Sequence`
component of a PMCFG function, a sequence of terms
- class `DynGenPar::Function`
PMCFG function.
- struct `DynGenPar::Pmcfg`
PMCFG.
- struct `DynGenPar::PmcfgComponentInfo`
attached to the parse trees as rule labels to allow obtaining syntax trees
- struct `DynGenPar::ParseState`
parse state struct, for bindings
- class `DynGenPar::Parser`
main class

Namespaces

- namespace [DynGenPar](#)

Defines

- `#define IS_EPSILON(cat) ((cat).isNull())`

Typedefs

- typedef QString [DynGenPar::Cat](#)
Category type: string or integer depending on DYNGENPAR_INTEGER_CATEGORIES.
- typedef const Cat & [DynGenPar::CatArg](#)
Category type (string or integer) when passed as an argument.
- typedef QHash< Cat, [QList< Rule >](#) > [DynGenPar::RuleSet](#)
- typedef QSet< Cat > [DynGenPar::TokenSet](#)
- typedef QSet< Cat > [DynGenPar::Predictions](#)
- typedef QMultiHash< [QList< Cat >](#), [MultiPrediction](#) > [DynGenPar::MultiPredictions](#)
- typedef QMultiHash< Cat, [NextTokenConstraints](#) > [DynGenPar::ConstrainedPredictions](#)
- typedef QMultiHash< [QList< Cat >](#), [ConstrainedMultiPrediction](#) > [DynGenPar::ConstrainedMultiPredictions](#)

Functions

- `template<class MultiHashType >`
`QHash< typename MultiHashType::key_type, QList< typename MultiHashType::mapped_type > > > DynGenPar::multiHashToListHash (const MultiHashType &multiHash)`
mapping from QMultiHash to QHash with list value for the Java bindings
- `uint DynGenPar::qHash (const NextTokenConstraints &nextTokenConstraints)`
simple hash function for next token constraints
- `uint DynGenPar::qHash (const PseudoCatScope &scope)`
- `Node DynGenPar::parseTreeToPmcfgSyntaxTree (const Node &parseTree)`
converts a parse tree obtained from a PMCFG to a PMCFG syntax tree
- `uint qHash (const QList< DynGenPar::Cat > &list)`
simple hash function for lists of categories

8.3.1 Define Documentation

8.3.1.1 `#define IS_EPSILON(cat) ((cat).isNull())`

Definition at line 58 of file `dyngenpar.h`.

8.3.2 Function Documentation

8.3.2.1 `uint qHash (const QList< DynGenPar::Cat > & list)`

simple hash function for lists of categories

Definition at line 431 of file `dyngenpar.cpp`.

8.4 flexlexertokensource.h File Reference

Classes

- class [DynGenPar::FlexLexerTokenSource](#)

Namespaces

- namespace [DynGenPar](#)

8.5 pgf.cpp File Reference

Namespaces

- namespace [DynGenPar](#)

Defines

- `#define` [CHECK_STATUS\(\)](#)

8.5.1 Define Documentation

8.5.1.1 `#define CHECK_STATUS()`

Value:

```
if (stream.status() != HaskellDataStream::Ok) \  
    qFatal("invalid PGF file or wrong version of GF")
```

8.6 pgf.h File Reference

Classes

- struct [DynGenPar::Pgf](#)
representation of the information in .pgf files in a format we can process
- class [DynGenPar::PgfParser](#)
parser for PGF grammars

Namespaces

- namespace [DynGenPar](#)

Defines

- #define [STATIC](#) static

Enumerations

- enum [DynGenPar::PgfReservedTokens](#) {
 [DynGenPar::PgfTokenEpsilon](#), [DynGenPar::PgfTokenLexError](#), [DynGenPar::PgfTokenVar](#), [DynGenPar::PgfTokenFloat](#),
 [DynGenPar::PgfTokenInt](#), [DynGenPar::PgfTokenString](#) }

Variables

- [STATIC](#) const char *const [DynGenPar::PreludeBind](#) = "&+"

8.6.1 Define Documentation

8.6.1.1 #define [STATIC](#) static

Definition at line 44 of file pgf.h.

Index

- ~AbstractLexerStateData
 - DynGenPar::AbstractLexerStateData, [24](#)
- ~Action
 - DynGenPar::Action, [24](#)
- ~ByteTokenSource
 - DynGenPar::ByteTokenSource, [29](#)
- ~FlexLexerTokenSource
 - DynGenPar::FlexLexerTokenSource, [35](#)
- ~ListTokenSource
 - DynGenPar::ListTokenSource, [41](#)
- ~Parser
 - DynGenPar::Parser, [51](#)
- ~PgfParser
 - DynGenPar::PgfParser, [66](#)
- ~StackItemData
 - DynGenPar::StackItemData, [83](#)
- ~StackItemType0
 - DynGenPar::StackItemType0, [85](#)
- ~StackItemType1
 - DynGenPar::StackItemType1, [88](#)
- ~StackItemType2
 - DynGenPar::StackItemType2, [90](#)
- ~StackItemType3
 - DynGenPar::StackItemType3, [92](#)
- ~StackItemType4
 - DynGenPar::StackItemType4, [95](#)
- ~StackItemType5
 - DynGenPar::StackItemType5, [97](#)
- ~StackItemType6
 - DynGenPar::StackItemType6, [100](#)
- ~TextByteTokenSource
 - DynGenPar::TextByteTokenSource, [106](#)
- ~TokenSource
 - DynGenPar::TokenSource, [109](#)
- ~UnifiedStackItemData
 - DynGenPar::UnifiedStackItemData, [113](#)
- action
 - DynGenPar::Rule, [77](#)
- ActionInfo
 - DynGenPar::ActionInfo, [25](#)
- add
 - DynGenPar::Alternative, [27](#)
 - DynGenPar::Function, [38](#)
 - DynGenPar::Rule, [77](#)
 - DynGenPar::Sequence, [79](#)
- addFunction
 - DynGenPar::Pmcfg, [68](#)
- addParent
 - DynGenPar::StackItem, [82](#)
 - DynGenPar::StackItemData, [84](#)
 - DynGenPar::StackItemType0, [85](#)
 - DynGenPar::StackItemType1, [88](#)
 - DynGenPar::StackItemType2, [90](#)
 - DynGenPar::StackItemType3, [92](#)
 - DynGenPar::StackItemType4, [95](#)
 - DynGenPar::StackItemType5, [98](#)
 - DynGenPar::StackItemType6, [100](#)
- addPmcfgRule
 - DynGenPar::Parser, [51](#)
- addRule
 - DynGenPar::Parser, [51](#)
- addToken
 - DynGenPar::Cfg, [32](#)
 - DynGenPar::Parser, [51](#)
- Alternative
 - DynGenPar::Alternative, [26, 27](#)
- arg
 - DynGenPar::Term, [102](#)
- argPositions
 - DynGenPar::PmcfgComponentInfo, [71](#)
- ByteTokenEpsilon
 - bytetokensource.h, [115](#)
- ByteTokenError
 - bytetokensource.h, [115](#)
- ByteTokenNul
 - bytetokensource.h, [115](#)
- ByteTokens
 - bytetokensource.h, [115](#)
- ByteTokenSource
 - DynGenPar::ByteTokenSource, [29](#)
- bytetokensource.h, [115](#)
 - ByteTokenEpsilon, [115](#)
 - ByteTokenError, [115](#)
 - ByteTokenNul, [115](#)

- ByteTokens, 115
- Cat
 - DynGenPar, 19
- cat
 - DynGenPar::ConstrainedMultiPrediction, 34
 - DynGenPar::FullRule, 37
 - DynGenPar::MultiPrediction, 45
 - DynGenPar::Node, 48
 - DynGenPar::StackItemType0, 85
 - DynGenPar::StackItemType1, 88
 - DynGenPar::StackItemType5, 98
- CatArg
 - DynGenPar, 19
- catComponents
 - DynGenPar::Parser, 59
- catName
 - DynGenPar::PgfParser, 66
- catNames
 - DynGenPar::Pgf, 64
- Cfg
 - DynGenPar::Cfg, 31
- cfRules
 - DynGenPar::Pmcf, 69
- CHECK_STATUS
 - pgf.cpp, 119
- children
 - DynGenPar::Node, 48
- clear
 - DynGenPar::LexerState, 40
- clone
 - DynGenPar::AbstractLexerStateData, 24
 - DynGenPar::StackItemData, 84
 - DynGenPar::StackItemType0, 85
 - DynGenPar::StackItemType1, 88
 - DynGenPar::StackItemType2, 90
 - DynGenPar::StackItemType3, 92
 - DynGenPar::StackItemType4, 95
 - DynGenPar::StackItemType5, 98
 - DynGenPar::StackItemType6, 100
 - DynGenPar::TextByteLexerStateData, 104
- col
 - DynGenPar::TextPosition, 108
- component
 - DynGenPar::Term, 102
- componentCats
 - DynGenPar::Parser, 59
- componentNames
 - DynGenPar::Pgf, 64
- computeConstrainedMultiPredictions
 - DynGenPar::Parser, 52
- computeConstrainedMultiPredictionsJava
 - DynGenPar::Parser, 52
- computeConstrainedPredictions
 - DynGenPar::Parser, 52, 53
- computeConstrainedPredictionsJava
 - DynGenPar::Parser, 53
- computeMultiPredictions
 - DynGenPar::Parser, 53
- computeMultiPredictionsJava
 - DynGenPar::Parser, 54
- computePredictions
 - DynGenPar::Parser, 54
- ConstrainedMultiPrediction
 - DynGenPar::ConstrainedMultiPrediction, 33
- ConstrainedMultiPredictions
 - DynGenPar, 20
- ConstrainedPredictions
 - DynGenPar, 20
- countCharacter
 - DynGenPar::TextPosition, 108
- curr
 - DynGenPar::StackItemType3, 93
- currentPosition
 - DynGenPar::TokenSource, 110
- currPos
 - DynGenPar::TokenSource, 111
- data
 - DynGenPar::LexerState, 40
 - DynGenPar::Node, 48
 - DynGenPar::PseudoCatScope, 72
 - DynGenPar::StackItem, 82
- derefUsage
 - DynGenPar::UnifiedStackItemData, 113
- DynGenPar, 17
 - Cat, 19
 - CatArg, 19
 - ConstrainedMultiPredictions, 20
 - ConstrainedPredictions, 20
 - multiHashToListHash, 21
 - MultiPredictions, 20
 - parseTreeToPmcfSyntaxTree, 21
 - PgfReservedTokens, 21
 - PgfTokenEpsilon, 21
 - PgfTokenFloat, 21
 - PgfTokenInt, 21
 - PgfTokenLexError, 21
 - PgfTokenString, 21
 - PgfTokenVar, 21
 - Predictions, 20
 - PreludeBind, 21

- qHash, 21
- RuleSet, 20
- TokenSet, 20
- dyngenpar.cpp, 116
 - qHash, 116
- dyngenpar.h, 116
 - IS_EPSILON, 118
 - qHash, 119
- DynGenPar::AbstractLexerStateData, 23
 - ~AbstractLexerStateData, 24
 - clone, 24
- DynGenPar::Action, 24
 - ~Action, 24
 - execute, 24
- DynGenPar::ActionInfo, 25
 - ActionInfo, 25
 - tree, 25
- DynGenPar::Alternative, 26
 - add, 27
 - Alternative, 26, 27
 - label, 27
 - operator<<, 27
 - operator+=", 27
 - setLabel, 27
 - toList, 28
- DynGenPar::ByteTokenSource, 28
 - ~ByteTokenSource, 29
 - ByteTokenSource, 29
 - readToken, 29
 - reset, 29
 - rewindTo, 30
 - setInputBuffer, 30
 - setInputBytes, 30
 - setInputFile, 30
 - setInputStdin, 30
 - setInputString, 30
 - stream, 31
- DynGenPar::Cfg, 31
 - addToken, 32
 - Cfg, 31
 - isToken, 32
 - rules, 32
 - startCat, 32
 - tokens, 32
- DynGenPar::ConstrainedMultiPrediction, 32
 - cat, 34
 - ConstrainedMultiPrediction, 33
 - fullLiteral, 34
 - nextTokenConstraints, 34
 - operator==, 34
- DynGenPar::FlexLexerTokenSource, 34
 - ~FlexLexerTokenSource, 35
 - flexLexer, 36
 - FlexLexerTokenSource, 35
 - readToken, 35
- DynGenPar::FullRule, 36
 - cat, 37
 - epsilonsSkipped, 37
 - FullRule, 36
 - rule, 37
 - rueno, 37
- DynGenPar::Function, 37
 - add, 38
 - Function, 38
 - operator<<, 39
 - operator+=", 38
 - toList, 39
- DynGenPar::LexerState, 39
 - clear, 40
 - data, 40
 - isNull, 40
 - LexerState, 39
 - operator==, 40
- DynGenPar::ListTokenSource, 40
 - ~ListTokenSource, 41
 - inputTokens, 42
 - ListTokenSource, 41
 - readToken, 41
 - rewindTo, 41
- DynGenPar::Match, 42
 - len, 43
 - Match, 43
 - nextTokenConstraints, 43
 - rueno, 43
 - scope, 43
 - tree, 43
- DynGenPar::MultiPrediction, 44
 - cat, 45
 - fullLiteral, 45
 - MultiPrediction, 44
 - operator==, 45
- DynGenPar::NextTokenConstraints, 45
 - expect, 46
 - operator==, 46
 - taboo, 46
- DynGenPar::Node, 47
 - cat, 48
 - children, 48
 - data, 48
 - Node, 47

- operator==, 48
- DynGenPar::Parser, 48
 - ~Parser, 51
 - addPmcfgrule, 51
 - addRule, 51
 - addToken, 51
 - catComponents, 59
 - componentCats, 59
 - computeConstrainedMultiPredictions, 52
 - computeConstrainedMultiPredictionsJava, 52
 - computeConstrainedPredictions, 52, 53
 - computeConstrainedPredictionsJava, 53
 - computeMultiPredictions, 53
 - computeMultiPredictionsJava, 54
 - computePredictions, 54
 - expandNonterminalPrediction, 55
 - expandNonterminalPredictionC, 55
 - expandNonterminalPredictionMulti, 56
 - expandNonterminalPredictionMultiC, 56
 - initCaches, 57
 - inputSource, 60
 - isLiteral, 57
 - isToken, 57
 - loadCfg, 57
 - loadPmcfgrule, 57
 - parse, 58
 - Parser, 51
 - pseudoCats, 60
 - rules, 60
 - startCat, 60
 - tokens, 61
- DynGenPar::ParseState, 61
 - errorPos, 62
 - errorToken, 62
 - incrementalMatches, 62
 - incrementalPos, 62
 - incrementalStacks, 62
 - lexerState, 62
 - ParseState, 62
 - reset, 62
- DynGenPar::Pgf, 63
 - catNames, 64
 - componentNames, 64
 - firstFunction, 64
 - functionNames, 64
 - Pgf, 64
 - pmcfgrule, 64
 - suffixes, 65
 - tokenHash, 65
- DynGenPar::PgfParser, 65
 - ~PgfParser, 66
 - catName, 66
 - filterCoercionsFromSyntaxTree, 66, 67
 - functionName, 67
 - pgf, 67
 - PgfParser, 66
 - setInputBuffer, 67
 - setInputBytes, 67
 - setInputFile, 67
 - setInputStdin, 67
 - setInputString, 67
- DynGenPar::Pmcfgrule, 68
 - addFunction, 68
 - cfRules, 69
 - functionIndices, 69
 - functionNames, 69
 - functions, 69
 - lookupFunction, 68
 - rules, 69
 - startCat, 69
 - tokens, 70
- DynGenPar::PmcfgruleComponentInfo, 70
 - argPositions, 71
 - PmcfgruleComponentInfo, 70
 - pmcfgrule, 71
- DynGenPar::PseudoCatScope, 71
 - data, 72
 - hasMcfgruleConstraint, 72
 - hasPConstraint, 72
 - isNull, 72
 - mcfgruleConstraint, 72
 - mcfgruleConstraints, 72
 - operator==, 72
 - pConstraint, 72
 - pConstraints, 73
 - PseudoCatScope, 72
- DynGenPar::PseudoCatScopeData, 73
 - mcfgruleConstraints, 74
 - pConstraints, 74
- DynGenPar::Rule, 75
 - action, 77
 - add, 77
 - label, 77
 - nextTokenConstraints, 77
 - operator<<, 77
 - operator+&, 77
 - Rule, 76
 - setLabel, 77
 - toList, 77
- DynGenPar::Sequence, 78

- add, 79
- nextTokenConstraints, 80
- operator<<, 79, 80
- operator+=", 79
- Sequence, 79
- toList, 80
- DynGenPar::StackItem, 80
 - addParent, 82
 - data, 82
 - setParents, 82
 - StackItem, 81, 82
 - type, 82
- DynGenPar::StackItemData, 83
 - ~StackItemData, 83
 - addParent, 84
 - clone, 84
 - setParents, 84
 - type, 84
- DynGenPar::StackItemType0, 84
 - ~StackItemType0, 85
 - addParent, 85
 - cat, 85
 - clone, 85
 - effCat, 86
 - parents, 86
 - pos, 86
 - scope, 86
 - setParents, 86
 - StackItemType0, 85
 - type, 86
- DynGenPar::StackItemType1, 87
 - ~StackItemType1, 88
 - addParent, 88
 - cat, 88
 - clone, 88
 - effCat, 88
 - parents, 88
 - scope, 88
 - setParents, 88
 - StackItemType1, 88
 - type, 89
- DynGenPar::StackItemType2, 89
 - ~StackItemType2, 90
 - addParent, 90
 - clone, 90
 - parent, 90
 - setParents, 91
 - StackItemType2, 90
 - type, 91
- DynGenPar::StackItemType3, 91
 - ~StackItemType3, 92
 - addParent, 92
 - clone, 92
 - curr, 93
 - i, 93
 - len, 93
 - nextTokenConstraints, 93
 - parent, 93
 - rule, 93
 - ruleno, 93
 - scope, 93
 - setParents, 93
 - StackItemType3, 92
 - tree, 94
 - type, 94
- DynGenPar::StackItemType4, 94
 - ~StackItemType4, 95
 - addParent, 95
 - clone, 95
 - len, 95
 - parent, 96
 - pos, 96
 - setParents, 96
 - StackItemType4, 95
 - target, 96
 - type, 96
- DynGenPar::StackItemType5, 97
 - ~StackItemType5, 97
 - addParent, 98
 - cat, 98
 - clone, 98
 - parent, 98
 - scope, 98
 - setParents, 98
 - StackItemType5, 97
 - type, 98
- DynGenPar::StackItemType6, 99
 - ~StackItemType6, 100
 - addParent, 100
 - clone, 100
 - i, 100
 - leaves, 100
 - nextTokenConstraints, 100
 - parent, 100
 - scope, 100
 - setParents, 101
 - StackItemType6, 100
 - tree, 101
 - type, 101
- DynGenPar::Term, 101

- arg, 102
- component, 102
- isComponent, 102
- isToken, 102
- operator==, 102
- Term, 102
- token, 103
- DynGenPar::TextByteLexerStateData, 103
 - clone, 104
 - streamPos, 104
 - TextByteLexerStateData, 104
 - textPos, 104
- DynGenPar::TextByteTokenSource, 105
 - ~TextByteTokenSource, 106
 - readToken, 106
 - reset, 106
 - rewindTo, 106
 - saveState, 106
 - TextByteTokenSource, 106
 - textPosition, 106
- DynGenPar::TextPosition, 107
 - col, 108
 - countCharacter, 108
 - line, 108
 - reset, 108
 - TextPosition, 107
- DynGenPar::TokenSource, 108
 - ~TokenSource, 109
 - currentPosition, 110
 - currPos, 111
 - matchParseTree, 110
 - nextToken, 110
 - parseTree, 110
 - readToken, 110
 - rewindTo, 110
 - saveState, 111
 - simpleRewind, 111
 - TokenSource, 109
 - tree, 111
- DynGenPar::UnifiedStackItemData, 112
 - ~UnifiedStackItemData, 113
 - derefUsage, 113
 - refUsage, 113
 - UnifiedStackItemData, 113
- effCat
 - DynGenPar::StackItemType0, 86
 - DynGenPar::StackItemType1, 88
- epsilonSkipped
 - DynGenPar::FullRule, 37
- errorPos
 - DynGenPar::ParseState, 62
- errorToken
 - DynGenPar::ParseState, 62
- execute
 - DynGenPar::Action, 24
- expandNonterminalPrediction
 - DynGenPar::Parser, 55
- expandNonterminalPredictionC
 - DynGenPar::Parser, 55
- expandNonterminalPredictionMulti
 - DynGenPar::Parser, 56
- expandNonterminalPredictionMultiC
 - DynGenPar::Parser, 56
- expect
 - DynGenPar::NextTokenConstraints, 46
- filterCoercionsFromSyntaxTree
 - DynGenPar::PgfParser, 66, 67
- firstFunction
 - DynGenPar::Pgf, 64
- flexLexer
 - DynGenPar::FlexLexerTokenSource, 36
- FlexLexerTokenSource
 - DynGenPar::FlexLexerTokenSource, 35
- flexlexertokensource.h, 119
- fullLiteral
 - DynGenPar::ConstrainedMultiPrediction, 34
 - DynGenPar::MultiPrediction, 45
- FullRule
 - DynGenPar::FullRule, 36
- Function
 - DynGenPar::Function, 38
- functionIndices
 - DynGenPar::Pmcfg, 69
- functionName
 - DynGenPar::PgfParser, 67
- functionNames
 - DynGenPar::Pgf, 64
 - DynGenPar::Pmcfg, 69
- functions
 - DynGenPar::Pmcfg, 69
- hasMcfgConstraint
 - DynGenPar::PseudoCatScope, 72
- hasPCConstraint
 - DynGenPar::PseudoCatScope, 72
- i
 - DynGenPar::StackItemType3, 93
 - DynGenPar::StackItemType6, 100
- incrementalMatches

- DynGenPar::ParseState, 62
- incrementalPos
 - DynGenPar::ParseState, 62
- incrementalStacks
 - DynGenPar::ParseState, 62
- initCaches
 - DynGenPar::Parser, 57
- inputSource
 - DynGenPar::Parser, 60
- inputTokens
 - DynGenPar::ListTokenSource, 42
- IS_EPSILON
 - dynngenpar.h, 118
- isComponent
 - DynGenPar::Term, 102
- isLiteral
 - DynGenPar::Parser, 57
- isNull
 - DynGenPar::LexerState, 40
 - DynGenPar::PseudoCatScope, 72
- isToken
 - DynGenPar::Cfg, 32
 - DynGenPar::Parser, 57
 - DynGenPar::Term, 102
- label
 - DynGenPar::Alternative, 27
 - DynGenPar::Rule, 77
- leaves
 - DynGenPar::StackItemType6, 100
- len
 - DynGenPar::Match, 43
 - DynGenPar::StackItemType3, 93
 - DynGenPar::StackItemType4, 95
- LexerState
 - DynGenPar::LexerState, 39
- lexerState
 - DynGenPar::ParseState, 62
- line
 - DynGenPar::TextPosition, 108
- ListTokenSource
 - DynGenPar::ListTokenSource, 41
- loadCfg
 - DynGenPar::Parser, 57
- loadPmcfg
 - DynGenPar::Parser, 57
- lookupFunction
 - DynGenPar::Pmcfg, 68
- Match
 - DynGenPar::Match, 43
- matchParseTree
 - DynGenPar::TokenSource, 110
- mcfgConstraint
 - DynGenPar::PseudoCatScope, 72
- mcfgConstraints
 - DynGenPar::PseudoCatScope, 72
 - DynGenPar::PseudoCatScopeData, 74
- multiHashToListHash
 - DynGenPar, 21
- MultiPrediction
 - DynGenPar::MultiPrediction, 44
- MultiPredictions
 - DynGenPar, 20
- nextToken
 - DynGenPar::TokenSource, 110
- nextTokenConstraints
 - DynGenPar::ConstrainedMultiPrediction, 34
 - DynGenPar::Match, 43
 - DynGenPar::Rule, 77
 - DynGenPar::Sequence, 80
 - DynGenPar::StackItemType3, 93
 - DynGenPar::StackItemType6, 100
- Node
 - DynGenPar::Node, 47
- operator<<
 - DynGenPar::Alternative, 27
 - DynGenPar::Function, 39
 - DynGenPar::Rule, 77
 - DynGenPar::Sequence, 79, 80
- operator+=
 - DynGenPar::Alternative, 27
 - DynGenPar::Function, 38
 - DynGenPar::Rule, 77
 - DynGenPar::Sequence, 79
- operator==
 - DynGenPar::ConstrainedMultiPrediction, 34
 - DynGenPar::LexerState, 40
 - DynGenPar::MultiPrediction, 45
 - DynGenPar::NextTokenConstraints, 46
 - DynGenPar::Node, 48
 - DynGenPar::PseudoCatScope, 72
 - DynGenPar::Term, 102
- parent
 - DynGenPar::StackItemType2, 90
 - DynGenPar::StackItemType3, 93
 - DynGenPar::StackItemType4, 96
 - DynGenPar::StackItemType5, 98
 - DynGenPar::StackItemType6, 100

- parents
 - DynGenPar::StackItemType0, [86](#)
 - DynGenPar::StackItemType1, [88](#)
- parse
 - DynGenPar::Parser, [58](#)
- Parser
 - DynGenPar::Parser, [51](#)
- ParseState
 - DynGenPar::ParseState, [62](#)
- parseTree
 - DynGenPar::TokenSource, [110](#)
- parseTreeToPmcfgSyntaxTree
 - DynGenPar, [21](#)
- pConstraint
 - DynGenPar::PseudoCatScope, [72](#)
- pConstraints
 - DynGenPar::PseudoCatScope, [73](#)
 - DynGenPar::PseudoCatScopeData, [74](#)
- Pgf
 - DynGenPar::Pgf, [64](#)
- pgf
 - DynGenPar::PgfParser, [67](#)
- pgf.cpp, [119](#)
 - CHECK_STATUS, [119](#)
- pgf.h, [119](#)
 - STATIC, [120](#)
- PgfParser
 - DynGenPar::PgfParser, [66](#)
- PgfReservedTokens
 - DynGenPar, [21](#)
- PgfTokenEpsilon
 - DynGenPar, [21](#)
- PgfTokenFloat
 - DynGenPar, [21](#)
- PgfTokenInt
 - DynGenPar, [21](#)
- PgfTokenLexError
 - DynGenPar, [21](#)
- PgfTokenString
 - DynGenPar, [21](#)
- PgfTokenVar
 - DynGenPar, [21](#)
- pmcfg
 - DynGenPar::Pgf, [64](#)
- PmcfgComponentInfo
 - DynGenPar::PmcfgComponentInfo, [70](#)
- pmcfgRule
 - DynGenPar::PmcfgComponentInfo, [71](#)
- pos
 - DynGenPar::StackItemType0, [86](#)
 - DynGenPar::StackItemType4, [96](#)
- Predictions
 - DynGenPar, [20](#)
- PreludeBind
 - DynGenPar, [21](#)
- pseudoCats
 - DynGenPar::Parser, [60](#)
- PseudoCatScope
 - DynGenPar::PseudoCatScope, [72](#)
- qHash
 - DynGenPar, [21](#)
 - dyngenpar.cpp, [116](#)
 - dyngenpar.h, [119](#)
- QList, [74](#)
- QSharedData, [75](#)
- readToken
 - DynGenPar::ByteTokenSource, [29](#)
 - DynGenPar::FlexLexerTokenSource, [35](#)
 - DynGenPar::ListTokenSource, [41](#)
 - DynGenPar::TextByteTokenSource, [106](#)
 - DynGenPar::TokenSource, [110](#)
- refUsage
 - DynGenPar::UnifiedStackItemData, [113](#)
- reset
 - DynGenPar::ByteTokenSource, [29](#)
 - DynGenPar::ParseState, [62](#)
 - DynGenPar::TextByteTokenSource, [106](#)
 - DynGenPar::TextPosition, [108](#)
- rewindTo
 - DynGenPar::ByteTokenSource, [30](#)
 - DynGenPar::ListTokenSource, [41](#)
 - DynGenPar::TextByteTokenSource, [106](#)
 - DynGenPar::TokenSource, [110](#)
- Rule
 - DynGenPar::Rule, [76](#)
- rule
 - DynGenPar::FullRule, [37](#)
 - DynGenPar::StackItemType3, [93](#)
- ruleno
 - DynGenPar::FullRule, [37](#)
 - DynGenPar::Match, [43](#)
 - DynGenPar::StackItemType3, [93](#)
- rules
 - DynGenPar::Cfg, [32](#)
 - DynGenPar::Parser, [60](#)
 - DynGenPar::Pmcfg, [69](#)
- RuleSet
 - DynGenPar, [20](#)
- saveState

- DynGenPar::TextByteTokenSource, 106
- DynGenPar::TokenSource, 111
- scope
 - DynGenPar::Match, 43
 - DynGenPar::StackItemType0, 86
 - DynGenPar::StackItemType1, 88
 - DynGenPar::StackItemType3, 93
 - DynGenPar::StackItemType5, 98
 - DynGenPar::StackItemType6, 100
- Sequence
 - DynGenPar::Sequence, 79
- setInputBuffer
 - DynGenPar::ByteTokenSource, 30
 - DynGenPar::PgfParser, 67
- setInputBytes
 - DynGenPar::ByteTokenSource, 30
 - DynGenPar::PgfParser, 67
- setInputFile
 - DynGenPar::ByteTokenSource, 30
 - DynGenPar::PgfParser, 67
- setInputStdin
 - DynGenPar::ByteTokenSource, 30
 - DynGenPar::PgfParser, 67
- setInputString
 - DynGenPar::ByteTokenSource, 30
 - DynGenPar::PgfParser, 67
- setLabel
 - DynGenPar::Alternative, 27
 - DynGenPar::Rule, 77
- setParents
 - DynGenPar::StackItem, 82
 - DynGenPar::StackItemData, 84
 - DynGenPar::StackItemType0, 86
 - DynGenPar::StackItemType1, 88
 - DynGenPar::StackItemType2, 91
 - DynGenPar::StackItemType3, 93
 - DynGenPar::StackItemType4, 96
 - DynGenPar::StackItemType5, 98
 - DynGenPar::StackItemType6, 101
- simpleRewind
 - DynGenPar::TokenSource, 111
- StackItem
 - DynGenPar::StackItem, 81, 82
- StackItemType0
 - DynGenPar::StackItemType0, 85
- StackItemType1
 - DynGenPar::StackItemType1, 88
- StackItemType2
 - DynGenPar::StackItemType2, 90
- StackItemType3
 - DynGenPar::StackItemType3, 92
- StackItemType4
 - DynGenPar::StackItemType4, 95
- StackItemType5
 - DynGenPar::StackItemType5, 97
- StackItemType6
 - DynGenPar::StackItemType6, 100
- startCat
 - DynGenPar::Cfg, 32
 - DynGenPar::Parser, 60
 - DynGenPar::Pmcf, 69
- STATIC
 - pgf.h, 120
- stream
 - DynGenPar::ByteTokenSource, 31
- streamPos
 - DynGenPar::TextByteLexerStateData, 104
- suffixes
 - DynGenPar::Pgf, 65
- taboo
 - DynGenPar::NextTokenConstraints, 46
- target
 - DynGenPar::StackItemType4, 96
- Term
 - DynGenPar::Term, 102
- TextByteLexerStateData
 - DynGenPar::TextByteLexerStateData, 104
- TextByteTokenSource
 - DynGenPar::TextByteTokenSource, 106
- textPos
 - DynGenPar::TextByteLexerStateData, 104
- TextPosition
 - DynGenPar::TextPosition, 107
- textPosition
 - DynGenPar::TextByteTokenSource, 106
- token
 - DynGenPar::Term, 103
- tokenHash
 - DynGenPar::Pgf, 65
- tokens
 - DynGenPar::Cfg, 32
 - DynGenPar::Parser, 61
 - DynGenPar::Pmcf, 70
- TokenSet
 - DynGenPar, 20
- TokenSource
 - DynGenPar::TokenSource, 109
- toList
 - DynGenPar::Alternative, 28
 - DynGenPar::Function, 39

DynGenPar::Rule, [77](#)

DynGenPar::Sequence, [80](#)

tree

DynGenPar::ActionInfo, [25](#)

DynGenPar::Match, [43](#)

DynGenPar::StackItemType3, [94](#)

DynGenPar::StackItemType6, [101](#)

DynGenPar::TokenSource, [111](#)

type

DynGenPar::StackItem, [82](#)

DynGenPar::StackItemData, [84](#)

DynGenPar::StackItemType0, [86](#)

DynGenPar::StackItemType1, [89](#)

DynGenPar::StackItemType2, [91](#)

DynGenPar::StackItemType3, [94](#)

DynGenPar::StackItemType4, [96](#)

DynGenPar::StackItemType5, [98](#)

DynGenPar::StackItemType6, [101](#)

UnifiedStackItemData

DynGenPar::UnifiedStackItemData, [113](#)