# Contents

## 1 Main Page

1.1 Introduction ................................................................. 1  
1.2 Basic Usage ............................................................... 1  
1.3 Java Bindings .............................................................. 3  
1.4 Features ................................................................. 4  
  1.4.1 Dynamic Grammar Additions ........................................... 4  
  1.4.2 Incremental Parsing ................................................... 4  
  1.4.3 Prediction .............................................................. 5  
  1.4.4 Rule Labels ............................................................. 5  
  1.4.5 Custom Parse Actions .................................................. 5  
  1.4.6 Arbitrary Token Sources .............................................. 5  
    1.4.6.1 Flex Lexer Token Source ......................................... 5  
  1.4.7 Hierarchical Parsing ................................................... 5  
  1.4.8 Parallel Multiple Context-Free Grammars (PMCFGs) .......... 5  
    1.4.8.1 Grammatical Framework (GF) PGF Grammars .................. 6  
  1.4.9 Next Token Constraints ............................................... 6  
1.5 TODO ................................................................. 6  
1.6 Copyright .............................................................. 6  
  1.6.1 Acknowledgments ...................................................... 6  
  1.6.2 Licensing and Warranty Disclaimer ................................. 6  
  1.6.3 Citations .............................................................. 7  

## 2 Namespace Index

2.1 Namespace List ........................................................... 9  

## 3 Hierarchical Index

3.1 Class Hierarchy .......................................................... 11  

## 4 Class Index

4.1 Class List .............................................................. 13  

## 5 File Index

5.1 File List .............................................................. 15
6 Namespace Documentation

6.1 DynGenPar Namespace Reference

   6.1.1 Typedef Documentation
        6.1.1.1 Cat
        6.1.1.2 CatArg
        6.1.1.3 ConstrainedMultiPredictions
        6.1.1.4 ConstrainedPredictions
        6.1.1.5 MultiPredictions
        6.1.1.6 Predictions
        6.1.1.7 RuleSet
        6.1.1.8 TokenSet

6.1.2 Enumeration Type Documentation
        6.1.2.1 PgfReservedTokens

6.1.3 Function Documentation
        6.1.3.1 parseTreeToPmcfgSyntaxTree
        6.1.3.2 qHash

6.1.4 Variable Documentation
        6.1.4.1 PreludeBind

7 Class Documentation

7.1 DynGenPar::AbstractLexerStateData Class Reference

        7.1.1 Detailed Description
        7.1.2 Constructor & Destructor Documentation
        7.1.2.1 ~AbstractLexerStateData
        7.1.3 Member Function Documentation
        7.1.3.1 clone

7.2 DynGenPar::Action Class Reference

        7.2.1 Detailed Description
        7.2.2 Constructor & Destructor Documentation
        7.2.2.1 ~Action
        7.2.3 Member Function Documentation
        7.2.3.1 execute

7.3 DynGenPar::ActionInfo Struct Reference

        7.3.1 Detailed Description
        7.3.2 Constructor & Destructor Documentation
        7.3.2.1 ActionInfo
        7.3.2.2 ActionInfo
        7.3.3 Member Data Documentation
        7.3.3.1 tree
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.10.2.2 Function</td>
<td>36</td>
</tr>
<tr>
<td>7.10.3 Member Function Documentation</td>
<td>36</td>
</tr>
<tr>
<td>7.10.3.1 add</td>
<td>36</td>
</tr>
<tr>
<td>7.10.3.2 operator+=</td>
<td>37</td>
</tr>
<tr>
<td>7.10.3.3 operator+=</td>
<td>37</td>
</tr>
<tr>
<td>7.10.3.4 operator&lt;&lt;</td>
<td>37</td>
</tr>
<tr>
<td>7.10.3.5 operator&lt;&lt;</td>
<td>37</td>
</tr>
<tr>
<td>7.10.3.6 toList</td>
<td>37</td>
</tr>
<tr>
<td>7.11 DynGenPar::LexerState Class Reference</td>
<td>37</td>
</tr>
<tr>
<td>7.11.1 Detailed Description</td>
<td>37</td>
</tr>
<tr>
<td>7.11.2 Constructor &amp; Destructor Documentation</td>
<td>37</td>
</tr>
<tr>
<td>7.11.2.1 LexerState</td>
<td>37</td>
</tr>
<tr>
<td>7.11.2.2 LexerState</td>
<td>37</td>
</tr>
<tr>
<td>7.11.3 Member Function Documentation</td>
<td>38</td>
</tr>
<tr>
<td>7.11.3.1 clear</td>
<td>38</td>
</tr>
<tr>
<td>7.11.3.2 data</td>
<td>38</td>
</tr>
<tr>
<td>7.11.3.3 isNull</td>
<td>38</td>
</tr>
<tr>
<td>7.11.3.4 operator==</td>
<td>38</td>
</tr>
<tr>
<td>7.12 DynGenPar::ListTokenSource Class Reference</td>
<td>38</td>
</tr>
<tr>
<td>7.12.1 Detailed Description</td>
<td>39</td>
</tr>
<tr>
<td>7.12.2 Constructor &amp; Destructor Documentation</td>
<td>39</td>
</tr>
<tr>
<td>7.12.2.1 ListTokenSource</td>
<td>39</td>
</tr>
<tr>
<td>7.12.2.2 ~ListTokenSource</td>
<td>39</td>
</tr>
<tr>
<td>7.12.3 Member Function Documentation</td>
<td>39</td>
</tr>
<tr>
<td>7.12.3.1 readToken</td>
<td>39</td>
</tr>
<tr>
<td>7.12.3.2 rewindTo</td>
<td>39</td>
</tr>
<tr>
<td>7.12.4 Member Data Documentation</td>
<td>39</td>
</tr>
<tr>
<td>7.12.4.1 inputTokens</td>
<td>39</td>
</tr>
<tr>
<td>7.13 DynGenPar::Match Struct Reference</td>
<td>39</td>
</tr>
<tr>
<td>7.13.1 Detailed Description</td>
<td>40</td>
</tr>
<tr>
<td>7.13.2 Constructor &amp; Destructor Documentation</td>
<td>40</td>
</tr>
<tr>
<td>7.13.2.1 Match</td>
<td>40</td>
</tr>
<tr>
<td>7.13.2.2 Match</td>
<td>40</td>
</tr>
<tr>
<td>7.13.2.3 Match</td>
<td>40</td>
</tr>
<tr>
<td>7.13.3 Member Data Documentation</td>
<td>40</td>
</tr>
<tr>
<td>7.13.3.1 len</td>
<td>40</td>
</tr>
<tr>
<td>7.13.3.2 nextTokenConstraints</td>
<td>40</td>
</tr>
<tr>
<td>7.13.3.3 ruleno</td>
<td>41</td>
</tr>
<tr>
<td>7.13.3.4 scope</td>
<td>41</td>
</tr>
<tr>
<td>7.13.3.5 tree</td>
<td>41</td>
</tr>
</tbody>
</table>
7.17.3.6 computeConstrainedPredictions ................................................. 48
7.17.3.7 computeConstrainedPredictions ................................................. 48
7.17.3.8 computeMultiPredictions ......................................................... 49
7.17.3.9 computeMultiPredictions ......................................................... 49
7.17.3.10 computePredictions .................................................................. 49
7.17.3.11 computePredictions .................................................................. 49
7.17.3.12 expandNonterminalPrediction .................................................... 50
7.17.3.13 expandNonterminalPredictionC ................................................... 50
7.17.3.14 expandNonterminalPredictionC ................................................... 50
7.17.3.15 expandNonterminalPredictionC ................................................... 50
7.17.3.16 expandNonterminalPredictionMulti ............................................ 51
7.17.3.17 expandNonterminalPredictionMultiC .......................................... 51
7.17.3.18 expandNonterminalPredictionMultiC .......................................... 51
7.17.3.19 expandNonterminalPredictionMultiC .......................................... 51
7.17.3.20 initCaches .............................................................................. 51
7.17.3.21 isLiteral ................................................................................. 52
7.17.3.22 isToken .................................................................................. 52
7.17.3.23 loadCfg ................................................................................ 52
7.17.3.24 loadPmcfg ........................................................................... 52
7.17.3.25 parse ...................................................................................... 52
7.17.3.26 parse ...................................................................................... 53
7.17.4 Member Data Documentation .......................................................... 53
7.17.4.1 catComponents ........................................................................ 53
7.17.4.2 componentCats ....................................................................... 54
7.17.4.3 inputSource .............................................................................. 54
7.17.4.4 pseudoCats ............................................................................ 54
7.17.4.5 rules ......................................................................................... 54
7.17.4.6 startCat .................................................................................... 54
7.17.4.7 tokens ...................................................................................... 55
7.18 DynGenPar::ParseState Struct Reference .......................................... 55
7.18.1 Detailed Description ...................................................................... 55
7.18.2 Constructor & Destructor Documentation ....................................... 55
7.18.2.1 ParseState ............................................................................... 55
7.18.2.2 ParseState ............................................................................... 55
7.18.3 Member Function Documentation .................................................. 55
7.18.3.1 reset ......................................................................................... 55
7.18.4 Member Data Documentation .......................................................... 56
7.18.4.1 errorPos ................................................................................. 56
7.18.4.2 errorToken .............................................................................. 56
7.18.4.3 incrementalMatches ................................................................. 56

Generated on Tue Nov 11 2014 19:15:00 for DynGenPar by Doxygen
7.18.4.4 incrementalPos ................................................. 56
7.18.4.5 incrementalStacks ............................................. 56
7.18.4.6 lexerState ..................................................... 56
7.19 DynGenPar::Pgf Struct Reference ................................. 56
  7.19.1 Detailed Description ........................................... 57
  7.19.2 Constructor & Destructor Documentation ...................... 57
    7.19.2.1 Pgf ..................................................... 57
    7.19.2.2 Pgf ..................................................... 57
  7.19.3 Member Data Documentation .................................. 57
    7.19.3.1 catNames ................................................. 57
    7.19.3.2 componentNames .......................................... 57
    7.19.3.3 firstFunction ............................................ 57
    7.19.3.4 functionNames ........................................... 58
    7.19.3.5 pmcfg ................................................... 58
    7.19.3.6 suffixes ............................................... 58
    7.19.3.7 tokenHash ............................................... 58
7.20 DynGenPar::PgfParser Class Reference ............................ 58
  7.20.1 Detailed Description ......................................... 59
  7.20.2 Constructor & Destructor Documentation ...................... 59
    7.20.2.1 PgfParser ............................................... 59
    7.20.2.2 PgfParser ............................................... 59
    7.20.2.3 ~PgfParser .............................................. 59
  7.20.3 Member Function Documentation ............................... 59
    7.20.3.1 catName ................................................ 59
    7.20.3.2 filterCoercionsFromSyntaxTree .......................... 59
    7.20.3.3 functionName ........................................... 59
    7.20.3.4 setInputBuffer ......................................... 59
    7.20.3.5 setInputBytes .......................................... 60
    7.20.3.6 setInputFile ............................................ 60
    7.20.3.7 setInputStdin .......................................... 60
    7.20.3.8 setInputString ......................................... 60
  7.20.4 Member Data Documentation .................................. 60
    7.20.4.1 pgf ..................................................... 60
7.21 DynGenPar::Pmcfg Struct Reference ............................... 60
  7.21.1 Detailed Description ......................................... 61
  7.21.2 Member Function Documentation ................................ 61
    7.21.2.1 addFunction ............................................ 61
    7.21.2.2 lookupFunction ........................................ 61
  7.21.3 Member Data Documentation .................................. 61
    7.21.3.1 cfRules ................................................ 61
7.35.3.1 addParent ........................................ 84
7.35.3.2 clone .............................................. 84
7.35.3.3 len ............................................... 84
7.35.3.4 parent ............................................ 84
7.35.3.5 pos ............................................... 84
7.35.3.6 setParents ........................................ 85
7.35.3.7 target ............................................. 85
7.35.3.8 type .............................................. 85

7.36 DynGenPar::StackItemType5 Class Reference ........................................ 85
  7.36.1 Detailed Description .................................. 86
  7.36.2 Constructor & Destructor Documentation .............. 86
    7.36.2.1 StackItemType5 ................................ 86
    7.36.2.2 ~StackItemType5 ............................... 86
  7.36.3 Member Function Documentation ........................ 86
    7.36.3.1 addParent .................................... 86
    7.36.3.2 cat .......................................... 86
    7.36.3.3 clone ........................................ 86
    7.36.3.4 parent ....................................... 86
    7.36.3.5 scope ........................................ 86
    7.36.3.6 setParents .................................... 86
    7.36.3.7 type ........................................ 87

7.37 DynGenPar::StackItemType6 Class Reference ........................................ 87
  7.37.1 Detailed Description .................................. 87
  7.37.2 Constructor & Destructor Documentation .............. 88
    7.37.2.1 StackItemType6 ................................ 88
    7.37.2.2 ~StackItemType6 ............................... 88
  7.37.3 Member Function Documentation ........................ 88
    7.37.3.1 addParent .................................... 88
    7.37.3.2 clone ........................................ 88
    7.37.3.3 i ............................................. 88
    7.37.3.4 leaves ....................................... 88
    7.37.3.5 nextTokenConstraints .......................... 88
    7.37.3.6 parent ....................................... 88
    7.37.3.7 scope ........................................ 88
    7.37.3.8 setParents .................................... 88
    7.37.3.9 tree .......................................... 89
    7.37.3.10 type ....................................... 89

7.38 DynGenPar::Term Struct Reference ............................................... 89
  7.38.1 Detailed Description .................................. 89
  7.38.2 Constructor & Destructor Documentation .............. 89
7.38.2.1 Term .................................................. 89
7.38.2.2 Term .................................................. 89
7.38.2.3 Term .................................................. 90
7.38.3 Member Function Documentation ................................................. 90
7.38.3.1 isComponent ........................................... 90
7.38.3.2 isToken ............................................... 90
7.38.3.3 operator== ........................................... 90
7.38.4 Member Data Documentation ..................................................... 90
7.38.4.1 arg ..................................................... 90
7.38.4.2 component ............................................ 90
7.38.4.3 token ............................................... 90
7.39 DynGenPar::TextByteLexerStateData Class Reference ................................................. 90
7.39.1 Detailed Description .................................................. 91
7.39.2 Constructor & Destructor Documentation ................................................. 91
7.39.2.1 TextByteLexerStateData .................................... 91
7.39.3 Member Function Documentation ................................................. 91
7.39.3.1 clone ................................................. 91
7.39.4 Member Data Documentation .................................................. 92
7.39.4.1 streamPos ........................................... 92
7.39.4.2 textPos .............................................. 92
7.40 DynGenPar::TextByteTokenSource Class Reference ................................................. 92
7.40.1 Detailed Description .................................................. 93
7.40.2 Constructor & Destructor Documentation ................................................. 93
7.40.2.1 TextByteTokenSource ...................................... 93
7.40.2.2 TextByteTokenSource ...................................... 93
7.40.2.3 ~TextByteTokenSource ...................................... 93
7.40.3 Member Function Documentation ................................................. 93
7.40.3.1 readToken ............................................. 93
7.40.3.2 reset .................................................. 93
7.40.3.3 rewindTo .............................................. 93
7.40.3.4 saveState ............................................. 94
7.40.3.5 textPosition ........................................... 94
7.41 DynGenPar::TextPosition Struct Reference .................................................. 94
7.41.1 Detailed Description .................................................. 94
7.41.2 Constructor & Destructor Documentation ................................................. 94
7.41.2.1 TextPosition ............................................ 94
7.41.2.2 TextPosition ............................................ 95
7.41.3 Member Function Documentation ................................................. 95
7.41.3.1 countCharacter ........................................ 95
7.41.3.2 reset .................................................. 95
## CONTENTS

7.41.4 Member Data Documentation . 95
7.41.4.1 col . 95
7.41.4.2 line . 95

7.42 DynGenPar::TokenSource Class Reference . 95
7.42.1 Detailed Description . 96
7.42.2 Constructor & Destructor Documentation . 96
7.42.2.1 TokenSource . 96
7.42.2.2 ~TokenSource . 96

7.42.3 Member Function Documentation . 96
7.42.3.1 currentPosition . 96
7.42.3.2 matchParseTree . 97
7.42.3.3 nextToken . 97
7.42.3.4 parseTree . 97
7.42.3.5 readToken . 97
7.42.3.6 rewindTo . 97
7.42.3.7 saveState . 97
7.42.3.8 simpleRewind . 98

7.42.4 Member Data Documentation . 98
7.42.4.1 currPos . 98
7.42.4.2 tree . 98

7.43 DynGenPar::UnifiedStackItemData Class Reference . 98
7.43.1 Detailed Description . 99
7.43.2 Constructor & Destructor Documentation . 99
7.43.2.1 UnifiedStackItemData . 99
7.43.2.2 ~UnifiedStackItemData . 99

7.43.3 Member Function Documentation . 99
7.43.3.1 derefUsage . 99
7.43.3.2 refUsage . 99

8 File Documentation  101
8.1 bytetokensource.h File Reference . 101
8.1.1 Macro Definition Documentation . 101
8.1.1.1 DYNGENPAR_INTEGER_CATEGORIES . 101
8.1.2 Enumeration Type Documentation . 101
8.1.2.1 ByteTokens . 101

8.2 dyngenpar.cpp File Reference . 102
8.2.1 Function Documentation . 102
8.2.1.1 qHash . 102

8.3 dyngenpar.h File Reference . 102
8.3.1 Macro Definition Documentation . 104
8.3.1.1 IS_EPSILON .......................................................... 104

8.3.2 Function Documentation ............................................. 104
  8.3.2.1 qHash .............................................................. 104

8.4 flexiexertokensource.h File Reference .............................. 104

8.5 pgf.cpp File Reference .................................................. 105
  8.5.1 Macro Definition Documentation ................................ 105
    8.5.1.1 CHECK_STATUS .............................................. 105

8.6 pgf.h File Reference ................................................... 105
  8.6.1 Macro Definition Documentation ................................ 106
    8.6.1.1 DYNGENPAR_INTEGER_CATEGORIES ......................... 106
    8.6.1.2 STATIC ...................................................... 106

Index .............................................................................. 107
Chapter 1

Main Page

1.1 Introduction

DynGenPar\[1\] 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

```cpp
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::PgfParser. 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:

```cpp
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.

```cpp
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.
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

```java
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.
- Likewise, the `DynGenPar::PseudoCatScopeData` class and the `DynGenPar::PseudoCatScope::data` method that returns a pointer to it are not available.
- 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.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');
parser.initCaches();

// Whenever we set the grammar directly, we have to call this function.

// 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();
matches = parser.parse(parseState);

The DynGenPar Java bindings are used in the FMathL Concise environment.[2]

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 Features

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) 2010-2013 Kevin Kofler <kevin.kofler@chello.at>
Copyright (C) 2014 DAGOPT Optimization Technologies GmbH, written by Kevin Kofler <kofler@dagopt.com>

1.6.1 Acknowledgments

Support by the Austrian Science Fund FWF under contract numbers P20631, P23554 and P22239 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/>.
1.6.3 Citations

The publication[1] constitutes a suitable citation for academic works. Though not required by the software license, it is kindly requested to include that citation in any research papers using this program.
Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

  DynGenPar .......................................................... 17
# Chapter 3

## Hierarchical Index

### 3.1 Class Hierarchy

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

<table>
<thead>
<tr>
<th>Class</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>DynGenPar::Action</td>
<td>24</td>
</tr>
<tr>
<td>DynGenPar::ActionInfo</td>
<td>24</td>
</tr>
<tr>
<td>DynGenPar::Cfg</td>
<td>30</td>
</tr>
<tr>
<td>DynGenPar::ConstrainedMultiPrediction</td>
<td>31</td>
</tr>
<tr>
<td>DynGenPar::FullRule</td>
<td>34</td>
</tr>
<tr>
<td>DynGenPar::LexerState</td>
<td>37</td>
</tr>
<tr>
<td>DynGenPar::Match</td>
<td>39</td>
</tr>
<tr>
<td>DynGenPar::MultiPrediction</td>
<td>41</td>
</tr>
<tr>
<td>DynGenPar::NextTokenConstraints</td>
<td>42</td>
</tr>
<tr>
<td>DynGenPar::Node</td>
<td>43</td>
</tr>
<tr>
<td>DynGenPar::Parser</td>
<td>44</td>
</tr>
<tr>
<td>DynGenPar::PgIParser</td>
<td>58</td>
</tr>
<tr>
<td>DynGenPar::ParseState</td>
<td>55</td>
</tr>
<tr>
<td>DynGenPar::Pgf</td>
<td>56</td>
</tr>
<tr>
<td>DynGenPar::Pmcfg</td>
<td>60</td>
</tr>
<tr>
<td>DynGenPar::PmcfgComponentInfo</td>
<td>62</td>
</tr>
<tr>
<td>DynGenPar::PseudoCatScope</td>
<td>63</td>
</tr>
<tr>
<td>QList</td>
<td>66</td>
</tr>
<tr>
<td>DynGenPar::AbstractLexerStateData</td>
<td>23</td>
</tr>
<tr>
<td>DynGenPar::TextByteLexerStateData</td>
<td>90</td>
</tr>
<tr>
<td>DynGenPar::PseudoCatScopeData</td>
<td>65</td>
</tr>
<tr>
<td>DynGenPar::StackItemData</td>
<td>73</td>
</tr>
<tr>
<td>DynGenPar::StackItemType0</td>
<td>74</td>
</tr>
<tr>
<td>DynGenPar::StackItemType1</td>
<td>77</td>
</tr>
<tr>
<td>DynGenPar::StackItemType3</td>
<td>80</td>
</tr>
<tr>
<td>DynGenPar::StackItemType5</td>
<td>85</td>
</tr>
<tr>
<td>DynGenPar::StackItemType6</td>
<td>87</td>
</tr>
<tr>
<td>DynGenPar::UnifiedStackItemData</td>
<td>98</td>
</tr>
<tr>
<td>DynGenPar::StackItemType2</td>
<td>78</td>
</tr>
<tr>
<td>DynGenPar::StackItemType4</td>
<td>83</td>
</tr>
<tr>
<td>DynGenPar::StackItem</td>
<td>71</td>
</tr>
<tr>
<td>DynGenPar::Term</td>
<td>89</td>
</tr>
<tr>
<td>DynGenPar::TextPosition</td>
<td>94</td>
</tr>
</tbody>
</table>
DynGenPar::TokenSource .......................................................... 95
DynGenPar::ByteTokenSource ..................................................... 28
DynGenPar::TextByteTokenSource .............................................. 92
DynGenPar::FlexLexerTokenSource ............................................ 33
DynGenPar::ListTokenSource ................................................... 38
QList< Cat >
DynGenPar::Rule ................................................................. 67
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 ........................................................................................ 24
DynGenPar::Alternative
   Multi-token predictions ................................................................................................. 26
DynGenPar::ByteTokenSource
   Token source that returns bytes ...................................................................................... 28
DynGenPar::Cfg
   An object representing a CFG (or a PMCFG in our internal representation) .......... 30
DynGenPar::ConstrainedMultiPrediction
   Multi-token predictions with next token constraints .................................................... 31
DynGenPar::FlexLexerTokenSource
   Token source for flex style lexers .................................................................................... 33
DynGenPar::FullRule
   Full rule as found in the initial graph .............................................................................. 34
DynGenPar::Function
   PMCFG function ............................................................................................................... 35
DynGenPar::LexerState
   Parse state struct, for bindings ...................................................................................... 37
DynGenPar::ListTokenSource
   Full rule as found in the initial graph .............................................................................. 38
DynGenPar::Match
   (possibly partial) match ................................................................................................. 39
DynGenPar::MultiPrediction
   Multi-token predictions .................................................................................................... 41
DynGenPar::NextTokenConstraints
   Rule constraints affecting the next token, for scannerless parsing .................................. 42
DynGenPar::Node
   Node in the parse tree ..................................................................................................... 43
DynGenPar::Parser
   Main class ........................................................................................................................ 44
DynGenPar::ParseState
   Parse state struct, for bindings ...................................................................................... 55
DynGenPar::Pgf
   Representation of the information in .pgf files in a format we can process ................. 56
DynGenPar::PgfParser
   Parser for PGF grammars ............................................................................................... 58
DynGenPar::Pmcfg
   PMCFG ............................................................................................................................. 60
DynGenPar::PmcfgComponentInfo
Attached to the parse trees as rule labels to allow obtaining syntax trees .......................... 62
DynGenPar::PseudoCatScope ................................................................. 63
DynGenPar::PseudoCatScopeData ......................................................... 65
QList ........................................................................................................ 66
QSharedData ............................................................................................. 66
DynGenPar::Rule ....................................................................................... 67
DynGenPar::Sequence
Component of a PMCFG function, a sequence of terms ......................................................... 69
DynGenPar::StackItem ................................................................................ 71
DynGenPar::StackItemData ........................................................................... 73
DynGenPar::StackItemType0
Type 0 item: during match, we're waiting for a token to shift .............................................. 74
DynGenPar::StackItemType1
Type 1 item: during type 0 item processing, we're executing a reduce .................................. 77
DynGenPar::StackItemType2
Type 2 item: during reduce, we're recursively executing another reduce ............................. 78
DynGenPar::StackItemType3
Type 3 item: during matchRemaining, we're executing a match .......................................... 80
DynGenPar::StackItemType4
Type 4 item: during reduce, we're executing a matchRemaining ......................................... 83
DynGenPar::StackItemType5
Type 5 item: during match (of an MCFG constraint), we're executing a matchRemaining ....... 85
DynGenPar::StackItemType6
Type 6 item: during match, we're matching a P constraint .................................................... 87
DynGenPar::Term
Term in the expression of a component of a PMCFG function ............................................. 89
DynGenPar::TextByteLexerStateData
You should not have to use this class directly, ever ........................................................... 90
DynGenPar::TextByteTokenSource ................................................................................. 92
DynGenPar::TextPosition
Text position ................................................................................................................. 94
DynGenPar::TokenSource ......................................................................................... 95
DynGenPar::UnifiedStackItemData ................................................................................. 98
Chapter 5

File Index

5.1 File List

Here is a list of all files with brief descriptions:

- bytetokensource.h .................................................. 101
- dyngenpar.cpp .................................................... 102
- dyngenpar.h ....................................................... 102
- flexlexertokensource.h ........................................... 104
- pgf.cpp ............................................................ 105
- pgf.h ............................................................... 105
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**

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 QMap< QList< Cat > > **MultiPrediction**
  MultiPredictions

• typedef QMap< QList< Cat > > **ConstrainedMultiPrediction**
  ConstrainedMultiPredictions

• typedef QMap< NextTokenConstraints > **ConstrainedPredictions**

• typedef QMap< QList< Cat > > **ConstrainedMultiPrediction**
  ConstrainedMultiPredictions
6.1 DynGenPar Namespace Reference

Enumerations

- enum PgReservedTokens {
  PgTokenEpsilon, PgTokenLexError, PgTokenVar, PgTokenFloat,
  PgTokenInt, PgTokenString
}

Functions

- uint qHash (const NextTokenConstraints &nextTokenConstraints)
  simple hash function for next token constraints
- Node parseTreeToPmcfgSyntaxTree (const Node &parseTree)
  converts a parse tree obtained from a PMCFG to a PMCFG syntax tree
- 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 198 of file dyngenpar.h.

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

Definition at line 175 of file dyngenpar.h.
6.1.1.5 typedef QMultiHash<QList<Cat>, MultiPrediction> DynGenPar::MultiPredictions
Definition at line 174 of file dyngenpar.h.

6.1.1.6 typedef QSet<Cat> DynGenPar::Predictions
Definition at line 158 of file dyngenpar.h.

6.1.1.7 typedef QHash<Cat, QList<Rule>> DynGenPar::RuleSet
Definition at line 139 of file dyngenpar.h.

6.1.1.8 typedef QSet<Cat> DynGenPar::TokenSet
Definition at line 140 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 41 of file pgf.h.

6.1.3 Function Documentation
6.1.3.1 Node DynGenPar::parseTreeToPmcfgSyntaxTree ( const Node & parseTree )
converts a parse tree obtained from a PMCFG to a PMCFG syntax tree
Definition at line 533 of file dyngenpar.cpp.

6.1.3.2 uint DynGenPar::qHash ( const PseudoCatScope & scope ) [inline]
Definition at line 325 of file dyngenpar.h.

6.1.3.3 uint DynGenPar::qHash ( const NextTokenConstraints & nextTokenConstraints )
simple hash function for next token constraints
Definition at line 446 of file dyngenpar.cpp.
6.1.4 Variable Documentation

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

Definition at line 47 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:

```
QSharedData

DynGenPar::AbstractLexerStateData

DynGenPar::PgfLexerStateData
DynGenPar::TextByteLexerStateData
```

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 661 of file dyngenpar.h.

7.1.2  Constructor & Destructor Documentation
7.1.2.1 virtual DynGenPar::AbstractLexerStateData::~AbstractLexerStateData ( ) [inline],[virtual]

Definition at line 663 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 354 of file dyngenpar.h.

7.2.2 Constructor & Destructor Documentation

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

Definition at line 356 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
7.3 DynGenPar::ActionInfo Struct Reference

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 346 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 348 of file dyngenpar.h.

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

Definition at line 349 of file dyngenpar.h.

7.3.3 Member Data Documentation

7.3.3.1 Node DynGenPar::ActionInfo::tree

Definition at line 350 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:

![Inheritance Diagram]

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 217 of file dyngenpar.h.

7.4.2  Constructor & Destructor Documentation

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

Definition at line 219 of file dyngenpar.h.

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

Definition at line 220 of file dyngenpar.h.

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

Definition at line 222 of file dyngenpar.h.
7.4.2.4 DynGenPar::Alternative::Alternative ( const QList<DynGenPar::Node> & list, const QVariant & label )
[inline]

Definition at line 224 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 245 of file dyngenpar.h.

7.4.3.2 QVariant DynGenPar::Alternative::label ( ) const [inline]
Definition at line 226 of file dyngenpar.h.

7.4.3.3 Alternative & DynGenPar::Alternative::operator+= ( const QList<DynGenPar::Node> & other ) [inline]
Definition at line 228 of file dyngenpar.h.

7.4.3.4 Alternative & DynGenPar::Alternative::operator+= ( const DynGenPar::Node & value ) [inline]
Definition at line 232 of file dyngenpar.h.

7.4.3.5 Alternative & DynGenPar::Alternative::operator<< ( const QList<DynGenPar::Node> & other )
[inline]
Definition at line 236 of file dyngenpar.h.

7.4.3.6 Alternative & DynGenPar::Alternative::operator<< ( const DynGenPar::Node & value ) [inline]
Definition at line 240 of file dyngenpar.h.

7.4.3.7 void DynGenPar::Alternative::setLabel ( const QVariant & label ) [inline]
Definition at line 227 of file dyngenpar.h.

7.4.3.8 QList<DynGenPar::Node> & DynGenPar::Alternative::toList ( ) [inline]
for bindings
Definition at line 249 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:

```
DynGenPar::TokenSource
    ↓
DynGenPar::ByteTokenSource
    ↓
DynGenPar::TextByteTokenSource
```

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 QByteArrayList &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 DynGenPar::ByteTokenSource Class Reference

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 146 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 148 of file dyngenpar.h.

7.6.2.2 DynGenPar::Cfg::Cfg ( const RuleSet & r, const TokenSet & t, CatArg sc ) [inline]
Definition at line 149 of file dyngenpar.h.

7.6.3 Member Function Documentation

7.6.3.1 void DynGenPar::Cfg::addToken ( CatArg cat ) [inline]
Definition at line 155 of file dyngenpar.h.

7.6.3.2 bool DynGenPar::Cfg::isToken ( CatArg cat ) const [inline]
Definition at line 154 of file dyngenpar.h.

7.6.4 Member Data Documentation

7.6.4.1 RuleSet DynGenPar::Cfg::rules
Definition at line 151 of file dyngenpar.h.

7.6.4.2 Cat DynGenPar::Cfg::startCat
Definition at line 153 of file dyngenpar.h.

7.6.4.3 TokenSet DynGenPar::Cfg::tokens
Definition at line 152 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 178 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 180 of file dyngenpar.h.

7.7.2.2 DynGenPar::ConstrainedMultiPrediction::ConstrainedMultiPrediction ( const QliSt< Cat > & fullLit, CatArg c ) [inline]
Definition at line 182 of file dyngenpar.h.

7.7.2.3 DynGenPar::ConstrainedMultiPrediction::ConstrainedMultiPrediction ( const QliSt< Cat > & fullLit, CatArg c, NextTokenConstraints ntc ) [inline]
Definition at line 184 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 192 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 188 of file dyngenpar.h.

7.7.4.2 QliSt< Cat > DynGenPar::ConstrainedMultiPrediction::fullLiteral
the entire literal completed by the prediction
Definition at line 187 of file dyngenpar.h.
7.7.4.3 NextTokenConstraints DynGenPar::ConstrainedMultiPrediction::nextTokenConstraints

only for nonterminals

Definition at line 189 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:

```
DynGenPar::TokenSource

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 201 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 203 of file dyngenpar.h.

7.9.2.2 DynGenPar::FullRule::FullRule ( CatArg c, const Rule & r, int epsSkipped, int n ) [inline]
Definition at line 204 of file dyngenpar.h.

7.9.3 Member Data Documentation

7.9.3.1 Cat DynGenPar::FullRule::cat
Definition at line 206 of file dyngenpar.h.

7.9.3.2 int DynGenPar::FullRule::epsilonsSkipped
Definition at line 208 of file dyngenpar.h.

7.9.3.3 Rule DynGenPar::FullRule::rule
Definition at line 207 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 212 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:

```
QList< Sequence >

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 858 of file dyngenpar.h.

7.10.2 Constructor & Destructor Documentation

7.10.2.1 DynGenPar::Function::Function ( ) [inline]
Definition at line 860 of file dyngenpar.h.

7.10.2.2 DynGenPar::Function::Function ( const QList< Sequence >& list ) [inline], [explicit]
Definition at line 861 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 879 of file dyngenpar.h.
7.10.3.2 Function & DynGenPar::Function::operator+= ( const QList<Sequence> & other ) [inline]
Definition at line 862 of file dyngenpar.h.

7.10.3.3 Function & DynGenPar::Function::operator+= ( const Sequence & value ) [inline]
Definition at line 866 of file dyngenpar.h.

7.10.3.4 Function & DynGenPar::Function::operator<< ( const QList<Sequence> & other ) [inline]
Definition at line 870 of file dyngenpar.h.

7.10.3.5 Function & DynGenPar::Function::operator<< ( const Sequence & value ) [inline]
Definition at line 874 of file dyngenpar.h.

7.10.3.6 QList<Sequence> & DynGenPar::Function::toList ( ) [inline]
for bindings
Definition at line 883 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 667 of file dyngenpar.h.

7.11.2 Constructor & Destructor Documentation

7.11.2.1 DynGenPar::LexerState::LexerState ( ) [inline]
Definition at line 669 of file dyngenpar.h.

7.11.2.2 DynGenPar::LexerState::LexerState ( AbstractLexerStateData * data ) [inline]
Definition at line 670 of file dyngenpar.h.
7.11.3 Member Function Documentation

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

Definition at line 671 of file dyngenpar.h.

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

Definition at line 673 of file dyngenpar.h.

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

Definition at line 672 of file dyngenpar.h.

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

Definition at line 674 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:

```
DynGenPar::TokenSource

DynGenPar::ListTokenSource
```

Public Member Functions

- ListTokenSource ()
- virtual ~ListTokenSource ()
- virtual bool rewindTo (int pos, const LexerState & other ) const
  
  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

Additional Inherited Members

7.12.1 Detailed Description

Definition at line 753 of file dyngenpar.h.

7.12.2 Constructor & Destructor Documentation

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

Definition at line 755 of file dyngenpar.h.

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

Definition at line 756 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 764 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 759 of file dyngenpar.h.

7.12.4 Member Data Documentation

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

Definition at line 757 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 330 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 332 of file dyngenpar.h.

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

Definition at line 333 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 335 of file dyngenpar.h.

7.13.3 Member Data Documentation

7.13.3.1 `int DynGenPar::Match::len`

Definition at line 337 of file dyngenpar.h.

7.13.3.2 `NextTokenConstraints DynGenPar::Match::nextTokenConstraints`

Definition at line 342 of file dyngenpar.h.
7.14 DynGenPar::MultiPrediction Struct Reference

7.13.3.3 int DynGenPar::Match::ruleno

used for PMCFGs
set to 0 where not needed to allow unification
Definition at line 339 of file dyngenpar.h.

7.13.3.4 PseudoCatScope DynGenPar::Match::scope

Definition at line 341 of file dyngenpar.h.

7.13.3.5 Node DynGenPar::Match::tree

Definition at line 338 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

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 161 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 163 of file dyngenpar.h.
7.14.2.2 DynGenPar::MultiPrediction::MultiPrediction ( const QList< Cat > & fullLit, CatArg c ) [inline]

Definition at line 164 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 170 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 167 of file dyngenpar.h.

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

the entire literal completed by the prediction
Definition at line 166 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 73 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 91 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 81 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 89 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

Generated on Tue Nov 11 2014 19:15:00 for DynGenPar by Doxygen
7.16.1 Detailed Description

node in the parse tree
Definition at line 258 of file dyngenpar.h.

7.16.2 Constructor & Destructor Documentation

7.16.2.1 DynGenPar::Node::Node ( ) [inline]
error node
Definition at line 259 of file dyngenpar.h.

7.16.2.2 DynGenPar::Node::Node ( CatArg c ) [inline]
Definition at line 260 of file dyngenpar.h.

7.16.2.3 DynGenPar::Node::Node ( CatArg c, const QVariant & d ) [inline]
Definition at line 261 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 268 of file dyngenpar.h.

7.16.4 Member Data Documentation

7.16.4.1 Cat DynGenPar::Node::cat
Definition at line 264 of file dyngenpar.h.

7.16.4.2 QList<Alternative> DynGenPar::Node::children
Definition at line 266 of file dyngenpar.h.

7.16.4.3 QVariant DynGenPar::Node::data
Definition at line 265 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:

```
DynGenPar::Parser
  DynGenPar::PgfParser
```

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 loadPmcfg (const Pmcfg &pmcfg)`
  
  *loads a PMCFG in standard form, converting it to the internal representation*
- `bool 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*
- `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<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, 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<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 997 of file dyngenpar.h.

7.17.2 Constructor & Destructor Documentation

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

Definition at line 1000 of file dyngenpar.h.

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

Definition at line 1002 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 950 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 680 of file dyngenpar.cpp.

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

Definition at line 1004 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

Generated on Tue Nov 11 2014 19:15:00 for DynGenPar by Doxygen
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 3096 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 1059 of file dyngenpar.h.

7.17.3.6  

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 2944 of file dyngenpar.cpp.

7.17.3.7  

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

overloaded version using ParseState, for bindings

Definition at line 1048 of file dyngenpar.h.
7.17.3.8 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 2603 of file dyngenpar.cpp.

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

overloaded version using ParseState, for bindings

Definition at line 1037 of file dyngenpar.h.

7.17.3.10 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 2312 of file dyngenpar.cpp.

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

overloaded version using ParseState, for bindings

Definition at line 1029 of file dyngenpar.h.
7.17.3.12  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 2372 of file dyngenpar.cpp.

7.17.3.13  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
This overload also enforces the next token constraints passed as a second argument.
Definition at line 2571 of file dyngenpar.cpp.

7.17.3.14  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 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 3012 of file dyngenpar.cpp.

7.17.3.15  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 3039 of file dyngenpar.cpp.
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 2715 of file dyngenpar.cpp.

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 3194 of file dyngenpar.cpp.

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 3222 of file dyngenpar.cpp.

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

Generated on Tue Nov 11 2014 19:15:00 for DynGenPar by Doxygen
should be called after each direct grammar modification (addRule takes care of updating the caches, which is more efficient than clearing.)

Definition at line 569 of file dyngenpar.cpp.

7.17.3.21 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 544 of file dyngenpar.cpp.

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

Definition at line 1003 of file dyngenpar.h.

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

Definition at line 1008 of file dyngenpar.h.

7.17.3.24 bool DynGenPar::Parser::loadPmcfg (const Pmcfg &pmcfg)
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 908 of file dyngenpar.cpp.

7.17.3.25 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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>errorPos</code></td>
<td>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.)</td>
</tr>
<tr>
<td><code>errorToken</code></td>
<td>if non-NULL, is filled with 0 (epsilon) on success and with the token triggering the error on error.</td>
</tr>
<tr>
<td><code>incrementalPos</code></td>
<td>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.</td>
</tr>
<tr>
<td><code>incrementalStacks</code></td>
<td>should be NULL for a non-incremental, non-predictive parse or a pointer to <code>QList&lt;StackItem&gt;</code> (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.)</td>
</tr>
<tr>
<td><code>incrementalMatches</code></td>
<td>should be NULL for a non-incremental parse. For an incremental parse, it can be NULL if you do not need to get your matches back a second time when there is no new input. If set to non-NULL, it will be returned as is if there is no new input in an incremental parse, or updated to the current return value otherwise.</td>
</tr>
<tr>
<td><code>lexerState</code></td>
<td>if non-NULL, 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 NULL 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 lexerState pointer is NULL, all rewind operations for the lexer will be passed a null (default-constructed) LexerState; stateful lexers will then fail any rewind operations.</td>
</tr>
</tbody>
</table>

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 2234 of file dyngenpar.cpp.

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

overloaded version using `ParseState`, for bindings

Definition at line 1022 of file dyngenpar.h.

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 1119 of file dyngenpar.h.

7.17.4.2  
\( \text{QHash} \langle \text{Cat}, \text{QPair} \langle \text{Cat}, \text{int} \rangle \rangle \) > 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 1113 of file dyngenpar.h.

7.17.4.3  
\( \text{TokenSource} \ast \) DynGenPar::Parser::inputSource \[\text{[protected]}\]

input source
Definition at line 1124 of file dyngenpar.h.

7.17.4.4  
\( \text{QHash} \langle \text{Cat}, \text{QPair} \langle \text{Cat}, \text{QList} \langle \text{Cat} \rangle \rangle \rangle \) > 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 1107 of file dyngenpar.h.

7.17.4.5  
\( \text{RuleSet} \) DynGenPar::Parser::rules

grammar rules
Warning

Modifying the grammar directly requires \text{initCaches}, use \text{addRule} if possible.

Definition at line 1077 of file dyngenpar.h.

7.17.4.6  
\( \text{Cat} \) DynGenPar::Parser::startCat

start category
Definition at line 1081 of file dyngenpar.h.
tokens
Definition at line 1079 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 971 of file dyngenpar.h.

### 7.18.2 Constructor & Destructor Documentation

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

Definition at line 972 of file dyngenpar.h.

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

Definition at line 976 of file dyngenpar.h.

### 7.18.3 Member Function Documentation

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

Definition at line 990 of file dyngenpar.h.
7.18.4 Member Data Documentation

7.18.4.1 `int DynGenPar::ParseState::errorPos`

Definition at line 983 of file dyngenpar.h.

7.18.4.2 `Cat DynGenPar::ParseState::errorToken`

Definition at line 984 of file dyngenpar.h.

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

Definition at line 987 of file dyngenpar.h.

7.18.4.4 `int DynGenPar::ParseState::incrementalPos`

Definition at line 985 of file dyngenpar.h.

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

Definition at line 986 of file dyngenpar.h.

7.18.4.6 `LexerState DynGenPar::ParseState::lexerState`

Definition at line 988 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

- `Pmcfg pmcfg`
  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 51 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 53 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 565 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 58 of file pgf.h.

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

names of category components

Definition at line 63 of file pgf.h.

#### 7.19.3.3 int DynGenPar::Pgf::firstFunction

the function ID of the first non-coercion function

Definition at line 64 of file pgf.h.
7.19.3.4 QStringList DynGenPar::Pgf::functionNames

names of functions, in general not unique
Definition at line 59 of file pgf.h.

7.19.3.5 Pmcfg DynGenPar::Pgf::pmcfg

the PMCFG (in (almost) standard form)
Definition at line 57 of file pgf.h.

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

list of &+ suffixes with their IDs
Definition at line 61 of file pgf.h.

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

hash table for quick token lexing
Definition at line 60 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:

```
DynGenPar::Parser
  DynGenPar::PgfParser
```

Public Member Functions

- PgfParser (const Pgf &p)
- PgfParser (const QString &fileName, const QString &concreteName=QString())
- virtual ~PgfParser ()
- void setInputStdin ()
7.20 DynGenPar::PgfParser Class Reference

- 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

Public Attributes

- Pgf pgf

Additional Inherited Members

7.20.1 Detailed Description

parser for PGF grammars

Definition at line 68 of file pgf.h.

7.20.2 Constructor & Destructor Documentation

7.20.2.1 DynGenPar::PgfParser::PgfParser (const Pgf &p)

Definition at line 1248 of file pgf.cpp.

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

Definition at line 1253 of file pgf.cpp.

7.20.2.3 virtual DynGenPar::PgfParser::~PgfParser ()

[inline],[virtual]

Definition at line 72 of file pgf.h.

7.20.3 Member Function Documentation

7.20.3.1 QString DynGenPar::PgfParser::catName (int cat) const

Definition at line 1284 of file pgf.cpp.

7.20.3.2 void DynGenPar::PgfParser::filterCoercionsFromSyntaxTree (Node &tree)

Definition at line 1304 of file pgf.cpp.

7.20.3.3 QString DynGenPar::PgfParser::functionName (int id) const

[inline]

Definition at line 79 of file pgf.h.

7.20.3.4 void DynGenPar::PgfParser::setInputBuffer (QByteArray *buffer)

Definition at line 1279 of file pgf.cpp.
Class Documentation

7.20.3.5   void DynGenPar::PgfParser::setInputBytes ( const QByteArray & bytes )

Definition at line 1269 of file pgf.cpp.

7.20.3.6   void DynGenPar::PgfParser::setInputFile ( const QString & fileName )

Definition at line 1264 of file pgf.cpp.

7.20.3.7   void DynGenPar::PgfParser::setInputStdin ( )

Definition at line 1259 of file pgf.cpp.

7.20.3.8   void DynGenPar::PgfParser::setInputString ( const QString & string )

Definition at line 1274 of file pgf.cpp.

7.20.4   Member Data Documentation

7.20.4.1   Pgf DynGenPar::PgfParser::pgf

Definition at line 81 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 DynGenPar::Pmcfg Struct Reference

7.21.1 Detailed Description

PMCFG.
Definition at line 889 of file dynenpar.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 937 of file dynenpar.h.

7.21.2.2 Function DynGenPar::Pmcfg::lookupFunction ( const QVariant & id ) const [inline]
Definition at line 943 of file dynenpar.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 930 of file dynenpar.h.

7.21.3.2 QHash<QString, int> DynGenPar::Pmcfg::functionIndices
Definition at line 936 of file dynenpar.h.

7.21.3.3 QHash<int, QString> DynGenPar::Pmcfg::functionNames
Definition at line 935 of file dynenpar.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 896 of file dynenpar.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 915 of file dyngenpar.h.

7.21.3.6  Cat DynGenPar::Pmcfg::startCat

start category
The start category must be 1-dimensional.
Definition at line 921 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 918 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 954 of file dyngenpar.h.

7.22.2  Constructor & Destructor Documentation

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

Definition at line 955 of file dyngenpar.h.
7.22.2.2 DynGenPar::PmcfgComponentInfo::PmcfgComponentInfo ( const Rule & rule ) [inline]

Definition at line 956 of file dyngenpar.h.

7.22.3 Member Data Documentation

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

must be the same size as pmcfgRule (even if the last arguments are not used)
Definition at line 959 of file dyngenpar.h.

7.22.3.2 Rule DynGenPar::PmcfgComponentInfo::pmcfgRule

Definition at line 958 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>> & mcfgConstraints ()
• bool hasPConstraint (CatArg cat) const
• bool hasMcfgConstraint (CatArg cat) const
• QPair<QPair<Node, NextTokenConstraints>, int> pConstraint (CatArg cat) const
• QPair<int, PseudoCatScope> mcfgConstraint (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 290 of file dyngenpar.h.

7.23.2 Constructor & Destructor Documentation

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

Definition at line 292 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 317 of file dyngenpar.h.

7.23.3.2 bool DynGenPar::PseudoCatScope::hasMcfgConstraint(CatArg cat) const [inline]
Definition at line 304 of file dyngenpar.h.

7.23.3.3 bool DynGenPar::PseudoCatScope::hasPConstraint(CatArg cat) const [inline]
Definition at line 301 of file dyngenpar.h.

7.23.3.4 bool DynGenPar::PseudoCatScope::isNull() const [inline]
Definition at line 315 of file dyngenpar.h.

7.23.3.5 QPair<int, PseudoCatScope> DynGenPar::PseudoCatScope::mcfgConstraint(CatArg cat) const [inline]
Definition at line 312 of file dyngenpar.h.

7.23.3.6 QHash<Cat, QPair<int, PseudoCatScope>> & DynGenPar::PseudoCatScope::mcfgConstraints()
[inline]
Definition at line 297 of file dyngenpar.h.

7.23.3.7 bool DynGenPar::PseudoCatScope::operator==(const PseudoCatScope & other) const [inline]
Definition at line 318 of file dyngenpar.h.

7.23.3.8 QPair<QPair<Node, NextTokenConstraints>, int> DynGenPar::PseudoCatScope::pConstraint(CatArg cat) const [inline]
Definition at line 307 of file dyngenpar.h.

7.23.3.9 QHash<Cat, QPair<QPair<Node, NextTokenConstraints>, int>> & DynGenPar::PseudoCatScope::p-Constraints()
[inline]
Definition at line 293 of file dyngenpar.h.

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

- dyngenpar.h
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 275 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 288 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 283 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
Inheritance diagram for DynGenPar::Rule:

```
DynGenPar::Rule
QList< Cat >
```

**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 99 of file dyngenpar.h.

### 7.27.2 Constructor & Destructor Documentation

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

Definition at line 101 of file dyngenpar.h.
7.27.2.2  DynGenPar::Rule::Rule ( const QVariant & label )  [inline],[explicit]
Definition at line 102 of file dyngenpar.h.

7.27.2.3  DynGenPar::Rule::Rule ( const QList< Cat > & list )  [inline],[explicit]
Definition at line 104 of file dyngenpar.h.

7.27.2.4  DynGenPar::Rule::Rule ( const QList< Cat > & list, const QVariant & label )  [inline]
Definition at line 106 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 129 of file dyngenpar.h.

7.27.3.2  QVariant DynGenPar::Rule::label ( ) const  [inline]
Definition at line 108 of file dyngenpar.h.

7.27.3.3  Rule& DynGenPar::Rule::operator+= ( const QList< Cat > & other )  [inline]
Definition at line 112 of file dyngenpar.h.

7.27.3.4  Rule& DynGenPar::Rule::operator+= ( const Cat & value )  [inline]
Definition at line 116 of file dyngenpar.h.

7.27.3.5  Rule& DynGenPar::Rule::operator<< ( const QList< Cat > & other )  [inline]
Definition at line 120 of file dyngenpar.h.

7.27.3.6  Rule& DynGenPar::Rule::operator<< ( const Cat & value )  [inline]
Definition at line 124 of file dyngenpar.h.

7.27.3.7  void DynGenPar::Rule::setLabel ( const QVariant & label )  [inline]
Definition at line 109 of file dyngenpar.h.

7.27.3.8  QList< Cat >& DynGenPar::Rule::toList ( )  [inline]
for bindings
Definition at line 133 of file dyngenpar.h.
7.27.4 Member Data Documentation

7.27.4.1 Action

\*DynGenPar::Rule::action

Definition at line 111 of file dyngenpar.h.

7.27.4.2 NextTokenConstraints DynGenPar::Rule::nextTokenConstraints

Definition at line 110 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:

```
QList<Term>

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 820 of file dyngenpar.h.

7.28.2 Constructor & Destructor Documentation

7.28.2.1 DynGenPar::Sequence::Sequence ( ) [inline]
Definition at line 824 of file dyngenpar.h.

7.28.2.2 DynGenPar::Sequence::Sequence ( const NextTokenConstraints & ntc ) [inline],[explicit]
Definition at line 825 of file dyngenpar.h.

7.28.2.3 DynGenPar::Sequence::Sequence ( const QList<Term> & list ) [inline],[explicit]
Definition at line 827 of file dyngenpar.h.

7.28.2.4 DynGenPar::Sequence::Sequence ( const QList<Term> & list, const NextTokenConstraints & ntc ) [inline]
Definition at line 829 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 848 of file dyngenpar.h.

7.28.3.2 Sequence & DynGenPar::Sequence::operator+= ( const QList<Term> & other ) [inline]
Definition at line 831 of file dyngenpar.h.

7.28.3.3 Sequence & DynGenPar::Sequence::operator+= ( const Term & value ) [inline]
Definition at line 835 of file dyngenpar.h.

7.28.3.4 Sequence & DynGenPar::Sequence::operator<< ( const QList<Term> & other ) [inline]
Definition at line 839 of file dyngenpar.h.

7.28.3.5 Sequence & DynGenPar::Sequence::operator<<( const Term & value ) [inline]
Definition at line 843 of file dyngenpar.h.
7.28.3.6  QList<Term>& DynGenPar::Sequence::toList () [inline]

for bindings
Definition at line 852 of file dyngenpar.h.

7.28.4  Member Data Documentation

7.28.4.1  NextTokenConstraints DynGenPar::Sequence::nextTokenConstraints

Definition at line 822 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 393 of file dyngenpar.h.
7.29.2 Constructor & Destructor Documentation

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

invalid type
Definition at line 395 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 610 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 617 of file dyngenpar.h.

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

type 2
Definition at line 623 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 628 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 637 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 643 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 649 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 412 of file dyngenpar.h.

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

Definition at line 414 of file dyngenpar.h.

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

Definition at line 413 of file dyngenpar.h.

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

Definition at line 411 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 365 of file dyngenpar.h.
7.30.2 Constructor & Destructor Documentation

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

Definition at line 367 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::StackItemType6, DynGenPar::StackItemType5, DynGenPar::StackItemType4, DynGenPar::StackItemType3, DynGenPar::StackItemType2, DynGenPar::StackItemType1, and DynGenPar::StackItemType0.

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

Implemented in DynGenPar::StackItemType6, DynGenPar::StackItemType5, DynGenPar::StackItemType4, DynGenPar::StackItemType3, DynGenPar::StackItemType2, DynGenPar::StackItemType1, and DynGenPar::StackItemType0.

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

Implemented in DynGenPar::StackItemType6, DynGenPar::StackItemType5, DynGenPar::StackItemType4, DynGenPar::StackItemType3, DynGenPar::StackItemType2, DynGenPar::StackItemType1, and DynGenPar::StackItemType0.

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

Implemented in DynGenPar::StackItemType6, DynGenPar::StackItemType5, DynGenPar::StackItemType4, DynGenPar::StackItemType3, DynGenPar::StackItemType2, DynGenPar::StackItemType1, and DynGenPar::StackItemType0.

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 () [inline]`
- `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 420 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 422 of file dyngenpar.h.

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

Definition at line 426 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 429 of file dyngenpar.h.

7.31.3.2 Cat DynGenPar::StackItemType0::cat ( ) const [inline]
Definition at line 434 of file dyngenpar.h.

7.31.3.3 virtual StackItemData ∗ DynGenPar::StackItemType0::clone ( ) [inline],[virtual]
Implements DynGenPar::StackItemData.
Definition at line 427 of file dyngenpar.h.

7.31.3.4 Cat DynGenPar::StackItemType0::effCat ( ) const [inline]
Definition at line 435 of file dyngenpar.h.

7.31.3.5 const QList< StackItem > & DynGenPar::StackItemType0::parents ( ) const [inline]
Definition at line 433 of file dyngenpar.h.

7.31.3.6 int DynGenPar::StackItemType0::pos ( ) const [inline]
Definition at line 436 of file dyngenpar.h.

7.31.3.7 PseudoCatScope DynGenPar::StackItemType0::scope ( ) const [inline]
Definition at line 437 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 430 of file dyngenpar.h.

7.31.3.9 virtual int DynGenPar::StackItemType0::type ( ) const [inline],[virtual]
Implements DynGenPar::StackItemData.
Definition at line 428 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:

```
QSharedData

DynGenPar::StackItemData

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 446 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 448 of file dyngenpar.h.

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

Definition at line 451 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 454 of file dyngenpar.h.

7.32.3.2 Cat DynGenPar::StackItemType1::cat ( ) const [inline]
Definition at line 459 of file dyngenpar.h.

7.32.3.3 virtual StackItemData ∗ DynGenPar::StackItemType1::clone ( ) [inline],[virtual]
Implements DynGenPar::StackItemData.
Definition at line 452 of file dyngenpar.h.

7.32.3.4 Cat DynGenPar::StackItemType1::effCat ( ) const [inline]
Definition at line 460 of file dyngenpar.h.

7.32.3.5 const QList < StackItem > & DynGenPar::StackItemType1::parents ( ) const [inline]
Definition at line 458 of file dyngenpar.h.

7.32.3.6 PseudoCatScope DynGenPar::StackItemType1::scope ( ) const [inline]
Definition at line 461 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 455 of file dyngenpar.h.

7.32.3.8 virtual int DynGenPar::StackItemType1::type ( ) const [inline],[virtual]
Implements DynGenPar::StackItemData.
Definition at line 453 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
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 470 of file dyngenpar.h.

7.33.2 Constructor & Destructor Documentation

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

Definition at line 472 of file dyngenpar.h.

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

Definition at line 473 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 476 of file dyngenpar.h.

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

Implements DynGenPar::StackItemData.
Definition at line 474 of file dyngenpar.h.

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

Definition at line 482 of file dyngenpar.h.

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

Implements DynGenPar::StackItemData.
Definition at line 479 of file dyngenpar.h.

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

Implements DynGenPar::StackItemData.
Definition at line 475 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:

![Inheritance Diagram](image)

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
- 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 488 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 490 of file `dyngenpar.h`. 
7.34.2 virtual DynGenPar::StackItemType3::~StackItemType3 ( ) [inline],[virtual]
Definition at line 496 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 499 of file dyngenpar.h.

7.34.3.2 virtual StackItemData* DynGenPar::StackItemType3::clone ( ) [inline],[virtual]
Implements DynGenPar::StackItemData.
Definition at line 497 of file dyngenpar.h.

7.34.3.3 int DynGenPar::StackItemType3::curr ( ) const [inline]
Definition at line 508 of file dyngenpar.h.

7.34.3.4 int DynGenPar::StackItemType3::i ( ) const [inline]
Definition at line 509 of file dyngenpar.h.

7.34.3.5 int DynGenPar::StackItemType3::len ( ) const [inline]
Definition at line 507 of file dyngenpar.h.

7.34.3.6 NextTokenConstraints DynGenPar::StackItemType3::nextTokenConstraints ( ) const [inline]
Definition at line 513 of file dyngenpar.h.

7.34.3.7 const StackItem& DynGenPar::StackItemType3::parent ( ) const [inline]
Definition at line 505 of file dyngenpar.h.

7.34.3.8 Rule DynGenPar::StackItemType3::rule ( ) const [inline]
Definition at line 506 of file dyngenpar.h.

7.34.3.9 int DynGenPar::StackItemType3::ruleno ( ) const [inline]
Definition at line 512 of file dyngenpar.h.

7.34.3.10 PseudoCatScope DynGenPar::StackItemType3::scope ( ) const [inline]
Definition at line 511 of file dyngenpar.h.
virtual void DynGenPar::StackItemType3::setParents ( const QList< StackItem > & ) [inline], [virtual]

Implements DynGenPar::StackItemData.
Definition at line 502 of file dyngenpar.h.

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

Definition at line 510 of file dyngenpar.h.

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

Implements DynGenPar::StackItemData.
Definition at line 498 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 ()

Generated on Tue Nov 11 2014 19:15:00 for DynGenPar by Doxygen
• 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 528 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 530 of file dyngenpar.h.

7.35.2.2 virtual DynGenPar::StackItemType4::~StackItemType4 ( ) [inline],[virtual]
Definition at line 532 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 535 of file dyngenpar.h.

7.35.3.2 virtual StackItemData* DynGenPar::StackItemType4::clone ( ) [inline],[virtual]
Implements DynGenPar::StackItemData.
Definition at line 533 of file dyngenpar.h.

7.35.3.3 int DynGenPar::StackItemType4::len ( ) const [inline]
Definition at line 544 of file dyngenpar.h.

7.35.3.4 const StackItem& DynGenPar::StackItemType4::parent ( ) const [inline]
Definition at line 541 of file dyngenpar.h.

7.35.3.5 int DynGenPar::StackItemType4::pos ( ) const [inline]
Definition at line 543 of file dyngenpar.h.
7.35.3.6 virtual void DynGenPar::StackItemType5::setParents (const QList<StackItem>&) [inline], [virtual]

Implements DynGenPar::StackItemData.

Definition at line 538 of file dyngenpar.h.

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

Definition at line 542 of file dyngenpar.h.

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

Implements DynGenPar::StackItemData.

Definition at line 534 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:

```
DynGenPar::StackItemType5
|   |   |
|   QSharedData
|   |   |
DynGenPar::StackItemData
|   |   |
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

Generated on Tue Nov 11 2014 19:15:00 for DynGenPar by Doxygen
• **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 553 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 555 of file dyngenpar.h.

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

Definition at line 558 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 561 of file dyngenpar.h.

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

Definition at line 568 of file dyngenpar.h.

#### 7.36.3.3 virtual stackItemData * DynGenPar::StackItemType5::clone ( ) [inline], [virtual]

Implements DynGenPar::StackItemData.

Definition at line 559 of file dyngenpar.h.

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

Definition at line 567 of file dyngenpar.h.

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

Definition at line 569 of file dyngenpar.h.

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

Implements DynGenPar::StackItemData.

Definition at line 564 of file dyngenpar.h.
7.36.3.7 virtual int DynGenPar::StackItemType5::type ( ) const [inline],[virtual]

Implements DynGenPar::StackItemData.

Definition at line 560 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:

![Inheritance Diagram](image)

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 577 of file dyngenpar.h.
 Constructor & Destructor Documentation

7.37.2.1 DynGenPar::StackItemType6::StackItemType6 ( const StackItem & parent, const QList & leaves, int i, const Node & tree, const PseudoCatScope & scope, const NextTokenConstraints & nextTokenConstraints )
[inline]
Definition at line 579 of file dyngenpar.h.

7.37.2.2 virtual DynGenPar::StackItemType6::~StackItemType6 ( ) [inline],[virtual]
Definition at line 584 of file dyngenpar.h.

 Member Function Documentation

7.37.3.1 virtual void DynGenPar::StackItemType6::addParent ( const StackItem & ) [inline],[virtual]
Implements DynGenPar::StackItemData.
Definition at line 587 of file dyngenpar.h.

7.37.3.2 virtual StackItemData * DynGenPar::StackItemType6::clone ( ) [inline],[virtual]
Implements DynGenPar::StackItemData.
Definition at line 585 of file dyngenpar.h.

7.37.3.3 int DynGenPar::StackItemType6::i ( ) const [inline]
Definition at line 595 of file dyngenpar.h.

7.37.3.4 QList<Node> DynGenPar::StackItemType6::leaves ( ) const [inline]
Definition at line 594 of file dyngenpar.h.

7.37.3.5 NextTokenConstraints DynGenPar::StackItemType6::nextTokenConstraints ( ) const [inline]
Definition at line 598 of file dyngenpar.h.

7.37.3.6 const StackItem& DynGenPar::StackItemType6::parent ( ) const [inline]
Definition at line 593 of file dyngenpar.h.

7.37.3.7 PseudoCatScope DynGenPar::StackItemType6::scope ( ) const [inline]
Definition at line 597 of file dyngenpar.h.

7.37.3.8 virtual void DynGenPar::StackItemType6::setParents ( const QList & StackItem & ) [inline],
[virtual]
Implements DynGenPar::StackItemData.
Definition at line 590 of file dyngenpar.h.
7.37.9  Node DynGenPar::StackItemType6::tree ( ) const  [inline]
Definition at line 596 of file dyngenpar.h.

7.37.10 virtual int DynGenPar::StackItemType6::type ( ) const  [inline],[virtual]
Implements DynGenPar::StackItemData.
Definition at line 586 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 800 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 802 of file dyngenpar.h.

7.38.2.2  DynGenPar::Term::Term ( int a, int c )  [inline]
Definition at line 803 of file dyngenpar.h.
7.38.2.3 DynGenPar::Term::Term (CatArg t) [inline]

Definition at line 804 of file dyngenpar.h.

7.38.3 Member Function Documentation

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

Definition at line 810 of file dyngenpar.h.

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

Definition at line 811 of file dyngenpar.h.

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

needed for QList

Definition at line 813 of file dyngenpar.h.

7.38.4 Member Data Documentation

7.38.4.1 int DynGenPar::Term::arg

Definition at line 805 of file dyngenpar.h.

7.38.4.2 int DynGenPar::Term::component

Definition at line 805 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 809 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 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:

- DynGenPar::TokenSource
  - DynGenPar::ByteTokenSource
    - DynGenPar::TextByteTokenSource

Public Member Functions

• TextByteTokenSource() (constructs with name passed as argument)
• TextByteTokenSource(const QString &fileName)
• virtual ~TextByteTokenSource() (destructor)
• 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 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 ()
  Reimplemented from DynGenPar::ByteTokenSource.

Additional Inherited Members

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 775 of file dyngenpar.h.

7.41.2 Constructor & Destructor Documentation

7.41.2.1 DynGenPar::TextPosition::TextPosition ( ) [inline]

Definition at line 776 of file dyngenpar.h.
7.42  DynGenPar::TokenSource Class Reference

Public Member Functions

- **TokenSource ()**
• virtual TokenSource ()
  get the next token from the input, increment current position, save parse tree

• Cat nextToken ()
  get the current input position

• 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 679 of file dyngenpar.h.

7.42.2 Constructor & Destructor Documentation

7.42.2.1 DynGenPar::TokenSource::TokenSource ( ) [inline]

Definition at line 681 of file dyngenpar.h.

7.42.2.2 virtual DynGenPar::TokenSource::~TokenSource ( ) [inline],[virtual]

Definition at line 682 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 715 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 711 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 688 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 700 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::ListTokenSource, DynGenPar::TextByteTokenSource, DynGenPar::ByteTokenSource,
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::ListTokenSource, DynGenPar::TextByteTokenSource, and DynGenPar::ByteToken-
Source.
Definition at line 724 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 729 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 735 of file dyngenpar.h.

7.42.4  Member Data Documentation

7.42.4.1  int DynGenPar::TokenSource::currPos  [protected]

Definition at line 742 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 750 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 374 of file `dyngenpar.h`.

### 7.43.2 Constructor & Destructor Documentation

#### 7.43.2.1 `DynGenPar::UnifiedStackItemData::UnifiedStackItemData ( )` [inline]

Definition at line 376 of file `dyngenpar.h`.

#### 7.43.2.2 `virtual DynGenPar::UnifiedStackItemData::~UnifiedStackItemData ( )` [inline], [virtual]

Definition at line 377 of file `dyngenpar.h`.

### 7.43.3 Member Function Documentation

#### 7.43.3.1 `unsigned DynGenPar::UnifiedStackItemData::derefUsage ( ) const` [inline]

Definition at line 386 of file `dyngenpar.h`.

#### 7.43.3.2 `unsigned DynGenPar::UnifiedStackItemData::refUsage ( ) const` [inline]

Definition at line 385 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

Macros

• #define DYNGENPAR_INTEGER_CATEGORIES

Enumerations

• enum ByteTokens { ByteTokenEpsilon = 0, ByteTokenNul = 256, ByteTokenError = 257 }

8.1.1 Macro Definition Documentation

8.1.1.1 #define DYNGENPAR_INTEGER_CATEGORIES

Definition at line 23 of file bytetokensource.h.

8.1.2 Enumeration Type Documentation

8.1.2.1 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 436 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

Generated on Tue Nov 11 2014 19:15:00 for DynGenPar by Doxygen
• 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

Macros

• #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

- 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 Macro Definition 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 436 of file dyngenpar.cpp.

8.4 flexlexertokensource.h File Reference

Classes

- class DynGenPar::FlexLexerTokenSource

Namespaces

- namespace DynGenPar
8.5  

File Reference

Namespaces

• namespace DynGenPar

Macros

• #define CHECK_STATUS()

8.5.1  

Macro Definition Documentation

8.5.1.1  

#define CHECK_STATUS( )

Value:

```cpp
if (stream.status() != HaskellDataStream::Ok) \
  qFatal("invalid PGF file or wrong version of GF")
```

8.6  

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

Macros

• #define DYNGENPAR_INTEGER_CATEGORIES
• #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 = "&+

Generated on Tue Nov 11 2014 19:15:00 for DynGenPar by Doxygen
8.6.1 Macro Definition Documentation

8.6.1.1 #define DYNGENPAR_INTEGER_CATEGORIES

Definition at line 28 of file pgf.h.

8.6.1.2 #define STATIC static

Definition at line 46 of file pgf.h.
Bibliography


Index

~AbstractLexerStateData
   DynGenPar::AbstractLexerStateData, 23
~Action
   DynGenPar::Action, 24
~ByteTokenSource
   DynGenPar::ByteTokenSource, 29
~FlexLexerTokenSource
   DynGenPar::FlexLexerTokenSource, 33
~ListTokenSource
   DynGenPar::ListTokenSource, 39
~Parser
   DynGenPar::Parser, 47
~PgfParser
   DynGenPar::PgfParser, 59
~StackItemData
   DynGenPar::StackItemData, 74
~StackItemType0
   DynGenPar::StackItemType0, 76
~StackItemType1
   DynGenPar::StackItemType1, 78
~StackItemType2
   DynGenPar::StackItemType2, 80
~StackItemType3
   DynGenPar::StackItemType3, 82
~StackItemType4
   DynGenPar::StackItemType4, 84
~StackItemType5
   DynGenPar::StackItemType5, 86
~StackItemType6
   DynGenPar::StackItemType6, 88
addPmcfgRule
   DynGenPar::Parser, 47
addRule
   DynGenPar::Parser, 47
addToken
   DynGenPar::Cfg, 31
   DynGenPar::Parser, 47
Alternative
   DynGenPar::Alternative, 26
arg
   DynGenPar::Term, 90
argPositions
   DynGenPar::PmcfgComponentInfo, 63
ByteTokenEpsilon
   bytetokensource.h, 101
ByteTokenError
   bytetokensource.h, 101
ByteTokenNul
   bytetokensource.h, 101
ByteTokenSource
   DynGenPar::ByteTokenSource, 29
ByteTokens
   bytetokensource.h, 101
bytetokensource.h
   ByteTokenEpsilon, 101
   ByteTokenError, 101
   ByteTokenNul, 101
   ByteTokens, 101
CHECK_STATUS
   pgf.cpp, 105
Cat
   DynGenPar, 19
cat
   DynGenPar::ConstrainedMultiPrediction, 32
   DynGenPar::FullRule, 35
   DynGenPar::MultiPrediction, 42
   DynGenPar::Node, 44
   DynGenPar::StackItemType0, 76
   DynGenPar::StackItemType1, 78
action
   DynGenPar::Rule, 69
ActionInfo
   DynGenPar::ActionInfo, 25
add
   DynGenPar::Alternative, 27
   DynGenPar::Function, 36
   DynGenPar::Rule, 68
   DynGenPar::Sequence, 70
addFunction
   DynGenPar::Pmcfg, 61
addParent
   DynGenPar::StackItem, 73
<table>
<thead>
<tr>
<th>Function/Variable</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>DynGenPar::PgParser</td>
<td>59</td>
</tr>
<tr>
<td>DynGenPar::Pgf</td>
<td>57</td>
</tr>
<tr>
<td>DynGenPar::Pmcfg</td>
<td>61</td>
</tr>
<tr>
<td>DynGenPar::Pmcfg</td>
<td>61</td>
</tr>
<tr>
<td>DynGenPar::PseudoCatScope</td>
<td>64</td>
</tr>
<tr>
<td>hasMcfgConstraint</td>
<td>DynGenPar::PseudoCatScope, 64</td>
</tr>
<tr>
<td>hasPConstraint</td>
<td>DynGenPar::PseudoCatScope, 64</td>
</tr>
<tr>
<td>i</td>
<td>DynGenPar::StackItemType3, 82</td>
</tr>
<tr>
<td>DynGenPar::StackItemType6, 88</td>
<td></td>
</tr>
<tr>
<td>IS_EPSILON</td>
<td>dyngenpar.h, 104</td>
</tr>
<tr>
<td>incrementalMatches</td>
<td>DynGenPar::ParseState, 56</td>
</tr>
<tr>
<td>incrementalPos</td>
<td>DynGenPar::ParseState, 56</td>
</tr>
<tr>
<td>incrementalStacks</td>
<td>DynGenPar::ParseState, 56</td>
</tr>
<tr>
<td>initCaches</td>
<td>DynGenPar::Parser, 51</td>
</tr>
<tr>
<td>inputSource</td>
<td>DynGenPar::Parser, 54</td>
</tr>
<tr>
<td>inputTokens</td>
<td>DynGenPar::ListTokenSource, 39</td>
</tr>
<tr>
<td>isComponent</td>
<td>DynGenPar::Term, 90</td>
</tr>
<tr>
<td>isLiteral</td>
<td>DynGenPar::Parser, 52</td>
</tr>
<tr>
<td>isNull</td>
<td>DynGenPar::LexerState, 38</td>
</tr>
<tr>
<td>DynGenPar::PseudoCatScope, 64</td>
<td></td>
</tr>
<tr>
<td>isToken</td>
<td>DynGenPar::Ctg, 31</td>
</tr>
<tr>
<td>DynGenPar::Parser, 52</td>
<td></td>
</tr>
<tr>
<td>DynGenPar::Term, 90</td>
<td></td>
</tr>
<tr>
<td>label</td>
<td>DynGenPar::Alternative, 27</td>
</tr>
<tr>
<td>DynGenPar::Rule, 68</td>
<td></td>
</tr>
<tr>
<td>leaves</td>
<td>DynGenPar::StackItemType6, 88</td>
</tr>
<tr>
<td>len</td>
<td>DynGenPar::Match, 40</td>
</tr>
<tr>
<td>DynGenPar::StackItemType3, 82</td>
<td></td>
</tr>
<tr>
<td>DynGenPar::StackItemType4, 84</td>
<td></td>
</tr>
<tr>
<td>LexerState</td>
<td>DynGenPar::LexerState, 37</td>
</tr>
<tr>
<td>lexerState</td>
<td>DynGenPar::ParseState, 56</td>
</tr>
<tr>
<td>line</td>
<td>DynGenPar::TextPosition, 95</td>
</tr>
<tr>
<td>ListTokenSource</td>
<td>DynGenPar::ListTokenSource, 39</td>
</tr>
<tr>
<td>loadCfg</td>
<td>DynGenPar::Parser, 52</td>
</tr>
<tr>
<td>loadPmcfg</td>
<td>DynGenPar::Parser, 52</td>
</tr>
<tr>
<td>lookupFunction</td>
<td>DynGenPar::Pmcfg, 61</td>
</tr>
<tr>
<td>Match</td>
<td>DynGenPar::Match, 40</td>
</tr>
<tr>
<td>matchParseTree</td>
<td>DynGenPar::TokenSource, 96</td>
</tr>
<tr>
<td>mcfgConstraint</td>
<td>DynGenPar::PseudoCatScope, 64</td>
</tr>
<tr>
<td>mcfgConstraints</td>
<td>DynGenPar::PseudoCatScope, 64</td>
</tr>
<tr>
<td>DynGenPar::PseudoCatScopeData, 65</td>
<td></td>
</tr>
<tr>
<td>MultiPrediction</td>
<td>DynGenPar::MultiPrediction, 41</td>
</tr>
<tr>
<td>MultiPredictions</td>
<td>DynGenPar, 19</td>
</tr>
<tr>
<td>nextToken</td>
<td>DynGenPar::TokenSource, 97</td>
</tr>
<tr>
<td>nextTokenConstraints</td>
<td>DynGenPar::ConstrainedMultiPrediction, 32</td>
</tr>
<tr>
<td>DynGenPar::Match, 40</td>
<td></td>
</tr>
<tr>
<td>DynGenPar::Rule, 69</td>
<td></td>
</tr>
<tr>
<td>DynGenPar::Sequence, 71</td>
<td></td>
</tr>
<tr>
<td>DynGenPar::StackItemType3, 82</td>
<td></td>
</tr>
<tr>
<td>DynGenPar::StackItemType6, 88</td>
<td></td>
</tr>
<tr>
<td>Node</td>
<td>DynGenPar::Node, 44</td>
</tr>
<tr>
<td>operator&lt;&lt;</td>
<td>DynGenPar::Alternative, 27</td>
</tr>
<tr>
<td>DynGenPar::Function, 37</td>
<td></td>
</tr>
<tr>
<td>DynGenPar::Rule, 68</td>
<td></td>
</tr>
<tr>
<td>DynGenPar::Sequence, 70</td>
<td></td>
</tr>
<tr>
<td>operator+=</td>
<td>DynGenPar::Alternative, 27</td>
</tr>
<tr>
<td>DynGenPar::Function, 36, 37</td>
<td></td>
</tr>
<tr>
<td>DynGenPar::Rule, 68</td>
<td></td>
</tr>
<tr>
<td>DynGenPar::Sequence, 70</td>
<td></td>
</tr>
<tr>
<td>operator==</td>
<td>DynGenPar::ConstrainedMultiPrediction, 32</td>
</tr>
<tr>
<td>DynGenPar::LexerState, 38</td>
<td></td>
</tr>
<tr>
<td>DynGenPar::MultiPrediction, 42</td>
<td></td>
</tr>
<tr>
<td>DynGenPar::NextTokenConstraints, 43</td>
<td></td>
</tr>
<tr>
<td>DynGenPar::Node, 44</td>
<td></td>
</tr>
<tr>
<td>DynGenPar::PseudoCatScope, 64</td>
<td></td>
</tr>
<tr>
<td>DynGenPar::Term, 90</td>
<td></td>
</tr>
<tr>
<td>pConstraint</td>
<td>DynGenPar::PseudoCatScope, 64</td>
</tr>
<tr>
<td>pConstraints</td>
<td>DynGenPar::PseudoCatScope, 64</td>
</tr>
<tr>
<td>DynGenPar::PseudoCatScopeData, 65</td>
<td></td>
</tr>
<tr>
<td>parent</td>
<td>DynGenPar::StackItemType2, 80</td>
</tr>
<tr>
<td>DynGenPar::StackItemType3, 82</td>
<td></td>
</tr>
</tbody>
</table>

Generated on Tue Nov 11 2014 19:15:00 for DynGenPar by Doxygen
<table>
<thead>
<tr>
<th>Function</th>
<th>Line</th>
</tr>
</thead>
<tbody>
<tr>
<td>DynGenPar::PgfParser, 59</td>
<td></td>
</tr>
<tr>
<td>setInputFile</td>
<td></td>
</tr>
<tr>
<td>DynGenPar::ByteTokenSource, 30</td>
<td></td>
</tr>
<tr>
<td>DynGenPar::PgfParser, 60</td>
<td></td>
</tr>
<tr>
<td>setInputStdin</td>
<td></td>
</tr>
<tr>
<td>DynGenPar::ByteTokenSource, 30</td>
<td></td>
</tr>
<tr>
<td>DynGenPar::PgfParser, 60</td>
<td></td>
</tr>
<tr>
<td>setInputString</td>
<td></td>
</tr>
<tr>
<td>DynGenPar::ByteTokenSource, 30</td>
<td></td>
</tr>
<tr>
<td>DynGenPar::PgfParser, 60</td>
<td></td>
</tr>
<tr>
<td>setLabel</td>
<td></td>
</tr>
<tr>
<td>DynGenPar::Alternative, 27</td>
<td></td>
</tr>
<tr>
<td>DynGenPar::Rule, 68</td>
<td></td>
</tr>
<tr>
<td>setParents</td>
<td></td>
</tr>
<tr>
<td>DynGenPar::StackItem, 73</td>
<td></td>
</tr>
<tr>
<td>DynGenPar::StackItemData, 74</td>
<td></td>
</tr>
<tr>
<td>DynGenPar::StackItemType0, 76</td>
<td></td>
</tr>
<tr>
<td>DynGenPar::StackItemType1, 78</td>
<td></td>
</tr>
<tr>
<td>DynGenPar::StackItemType2, 80</td>
<td></td>
</tr>
<tr>
<td>DynGenPar::StackItemType3, 82</td>
<td></td>
</tr>
<tr>
<td>DynGenPar::StackItemType4, 84</td>
<td></td>
</tr>
<tr>
<td>DynGenPar::StackItemType5, 86</td>
<td></td>
</tr>
<tr>
<td>DynGenPar::StackItemType6, 88</td>
<td></td>
</tr>
<tr>
<td>simpleRewind</td>
<td></td>
</tr>
<tr>
<td>DynGenPar::TokenSource, 97</td>
<td></td>
</tr>
<tr>
<td>StackItem</td>
<td></td>
</tr>
<tr>
<td>DynGenPar::StackItem, 72</td>
<td></td>
</tr>
<tr>
<td>StackItemType0</td>
<td></td>
</tr>
<tr>
<td>DynGenPar::StackItem0, 75</td>
<td></td>
</tr>
<tr>
<td>StackItemType1</td>
<td></td>
</tr>
<tr>
<td>DynGenPar::StackItem1, 77</td>
<td></td>
</tr>
<tr>
<td>StackItemType2</td>
<td></td>
</tr>
<tr>
<td>DynGenPar::StackItem2, 79</td>
<td></td>
</tr>
<tr>
<td>StackItemType3</td>
<td></td>
</tr>
<tr>
<td>DynGenPar::StackItem3, 81</td>
<td></td>
</tr>
<tr>
<td>StackItemType4</td>
<td></td>
</tr>
<tr>
<td>DynGenPar::StackItem4, 84</td>
<td></td>
</tr>
<tr>
<td>StackItemType5</td>
<td></td>
</tr>
<tr>
<td>DynGenPar::StackItem5, 86</td>
<td></td>
</tr>
<tr>
<td>StackItemType6</td>
<td></td>
</tr>
<tr>
<td>DynGenPar::StackItem6, 88</td>
<td></td>
</tr>
<tr>
<td>startCat</td>
<td></td>
</tr>
<tr>
<td>DynGenPar::Cfg, 31</td>
<td></td>
</tr>
<tr>
<td>DynGenPar::Parser, 54</td>
<td></td>
</tr>
<tr>
<td>DynGenPar::Pmcfg, 62</td>
<td></td>
</tr>
<tr>
<td>stream</td>
<td></td>
</tr>
<tr>
<td>DynGenPar::ByteTokenSource, 30</td>
<td></td>
</tr>
<tr>
<td>streamPos</td>
<td></td>
</tr>
<tr>
<td>DynGenPar::TextByteLexerStateData, 92</td>
<td></td>
</tr>
<tr>
<td>suffixes</td>
<td></td>
</tr>
<tr>
<td>DynGenPar::Pgf, 58</td>
<td></td>
</tr>
<tr>
<td>taboo</td>
<td></td>
</tr>
<tr>
<td>DynGenPar::NextTokenConstraints, 43</td>
<td></td>
</tr>
<tr>
<td>target</td>
<td></td>
</tr>
<tr>
<td>DynGenPar::StackItemType4, 85</td>
<td></td>
</tr>
<tr>
<td>Term</td>
<td></td>
</tr>
<tr>
<td>DynGenPar::Term, 89</td>
<td></td>
</tr>
<tr>
<td>TextByteLexerStateData</td>
<td></td>
</tr>
</tbody>
</table>