QMLParser

1.0

Generated by Doxygen 1.13.2

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Chapter 4

Namespace Documentation

4.1 iif_sadaf Namespace Reference

Namespaces

- namespace talk

4.2 iif_sadaf::talk Namespace Reference

Namespaces

- namespace QMLParser

4.3 iif_sadaf::talk::QMLParser Namespace Reference

Classes

- class Parser

  Parses a sequence of tokens into a Quantified Modal Logic (QML) expression tree.
- struct Token

  Represents a single lexical token.

Enumerations

- enum class TokenType : uint8_t {
  NIL , EOI , ILLEGAL , NOT ,
  AND , OR , IF , EQ ,
  NEC , POS , FORALL , EXISTS ,
  NOT_EXISTS , ID , NEQ , VARIABLE ,
  IDENTIFIER , LPAREN , RPAREN , LBRACKET ,
  RBRACKET , COMMA }

  Represents different types of tokens used in the QMLParser.

Functions

- std::vector< Token > lex (const std::string &string)

  Tokenizes a given QML formula.
- std::optional< QMLExpression::Operator > mapToAlethicOperator (TokenType type)

  Maps token types to alethic modal operators.
- std::optional< QMLExpression::Operator > mapToDeonticOperator (TokenType type)

  Maps token types to deontic modal operators.
- std::optional< QMLExpression::Operator > mapToEpistemicOperator (TokenType type)

  Maps token types to epistemic modal operators.
- std::expected< QMLExpression::Expression, std::string > parse (const std::string &formula, Parser::ParseFunction entryPoint=&Parser::equivalence, Parser::MappingFunction mappingFunction=&mapToAlethicOperator)
- std::expected< QMLExpression::Expression, std::string > parse (const std::string &formula, Parser::MappingFunction mappingFunction)

4.3.1 Enumeration Type Documentation

4.3.1.1 TokenType

```
enum class iif_sadaf::talk::QMLParser::TokenType : uint8_t [strong]
```

Represents different types of tokens used in the QMLParser.

Enumerator

NIL	
EOI	
ILLEGAL	
NOT	
AND	
OR	
IF	
EQ	
NEC	
POS	
FORALL	
EXISTS	
NOT_EXISTS	
ID	
NEQ	
VARIABLE	
IDENTIFIER	
LPAREN	
RPAREN	
LBRACKET	
RBRACKET	
COMMA	

4.3.2 Function Documentation

4.3.2.1 lex()

```
std::vector< Token > iif_sadaf::talk::QMLParser::lex (
            const std::string & string)
```

Tokenizes a given QML formula.

Parameters

formula	The input string representing a QML formula.

Returns

A vector of `Token` objects.

4.3.2.2 mapToAlethicOperator()

```
std::optional< QMLExpression::Operator > iif_sadaf::talk::QMLParser::mapToAlethicOperator (
            TokenType type)
```

Maps token types to alethic modal operators.

Parameters

type	The token type.

Returns

The corresponding QMLExpression::Operator, or std::nullopt if the token is not a modal operator.

4.3.2.3 mapToDeonticOperator()

```
std::optional< QMLExpression::Operator > iif_sadaf::talk::QMLParser::mapToDeonticOperator (
            TokenType type)
```

Maps token types to deontic modal operators.

Parameters

type	The token type.

Returns

The corresponding QMLExpression::Operator, or std::nullopt if the token is not a modal operator.

4.3.2.4 mapToEpistemicOperator()

```
std::optional< QMLExpression::Operator > iif_sadaf::talk::QMLParser::mapToEpistemicOperator (
            TokenType type)
```

Maps token types to epistemic modal operators.

Parameters

type	The token type.

Returns

The corresponding QMLExpression::Operator, or std::nullopt if the token is not a modal operator.

4.3.2.5 parse() [1/2]

```
std::expected< QMLExpression::Expression, std::string > iif_sadaf::talk::QMLParser::parse (
            const std::string & formula,
            Parser::MappingFunction mappingFunction)
```

4.3.2.6 parse() [2/2]

```
std::expected< QMLExpression::Expression, std::string > iif_sadaf::talk::QMLParser::parse (
            const std::string & formula,
            Parser::ParseFunction entryPoint = &Parser::equivalence,
            Parser::MappingFunction mappingFunction = &mapToAlethicOperator)
```

Chapter 5

Class Documentation

5.1 iif_sadaf::talk::QMLParser::Parser Class Reference

Parses a sequence of tokens into a Quantified Modal Logic (QML) expression tree.

`#include <parser.hpp>`

Public Types

- using ParseFunction = std::function<std::expected<QMLExpression::Expression, std::string>(Parser&)>
  Defines the function signature for parsing rules.
- using MappingFunction = std::function<std::optional<QMLExpression::Operator>(TokenType)>
  Maps token types to QML logical operators.

Public Member Functions

- Parser (const std::vector< Token > &tokens, MappingFunction mapFunc=mapToAlethicOperator)
  Constructs a Parser instance.
- std::expected< QMLExpression::Expression, std::string > parse (ParseFunction entryPoint=&equivalence)
  Parses the token stream into a QML expression.
- std::expected< QMLExpression::Expression, std::string > equivalence ()
- std::expected< QMLExpression::Expression, std::string > implication ()
- std::expected< QMLExpression::Expression, std::string > conjunction_disjunction ()
- std::expected< QMLExpression::Expression, std::string > unary ()
- std::expected< QMLExpression::Expression, std::string > atomic ()
- std::expected< QMLExpression::Expression, std::string > predication ()
- std::expected< QMLExpression::Expression, std::string > identity ()
- std::expected< QMLExpression::Expression, std::string > inequality ()

5.1.1 Detailed Description

Parses a sequence of tokens into a Quantified Modal Logic (QML) expression tree.

This class implements a recursive descent parser for QML expressions. It processes a vector of tokens and constructs an abstract syntax tree (AST) representing a well-formed formula.

5.1.2 Member Typedef Documentation

5.1.2.1 MappingFunction

```
using iif_sadaf::talk::QMLParser::Parser::MappingFunction = std::function<std::optional<QMLExpression↩
::Operator>(TokenType)>
```

Maps token types to QML logical operators.

5.1.2.2 ParseFunction

```
using iif_sadaf::talk::QMLParser::Parser::ParseFunction = std::function<std::expected<QMLExpression↩
::Expression, std::string>(Parser&)>
```

Defines the function signature for parsing rules.

5.1.3 Constructor & Destructor Documentation

5.1.3.1 Parser()

```
iif_sadaf::talk::QMLParser::Parser::Parser (
            const std::vector< Token > & tokens,
            MappingFunction mapFunc = mapToAlethicOperator)
```

Constructs a Parser instance.

Parameters

tokens	The list of tokens to parse.
mapFunc	Function that maps tokens to modal operators (default: alethic logic).

5.1.4 Member Function Documentation

5.1.4.1 atomic()

```
std::expected< QMLExpression::Expression, std::string > iif_sadaf::talk::QMLParser::Parser↩
::atomic ()
```

5.1.4.2 conjunction_disjunction()

```
std::expected< QMLExpression::Expression, std::string > iif_sadaf::talk::QMLParser::Parser↩
::conjunction_disjunction ()
```

5.1.4.3 equivalence()

```
std::expected< QMLExpression::Expression, std::string > iif_sadaf::talk::QMLParser::Parser↩
::equivalence ()
```

5.1.4.4 identity()

```
std::expected< QMLExpression::Expression, std::string > iif_sadaf::talk::QMLParser::Parser↩
::identity ()
```

5.1.4.5 implication()

```
std::expected< QMLExpression::Expression, std::string > iif_sadaf::talk::QMLParser::Parser↩
::implication ()
```

5.1.4.6 inequality()

```
std::expected< QMLExpression::Expression, std::string > iif_sadaf::talk::QMLParser::Parser↩
::inequality ()
```

5.1.4.7 parse()

```
std::expected< QMLExpression::Expression, std::string > iif_sadaf::talk::QMLParser::Parser↩
::parse (
            ParseFunction entryPoint = &equivalence)
```

Parses the token stream into a QML expression.

Parameters

entryPoint	The starting parse rule (default: equivalence).

Returns

Parsed QML expression or an error message.

5.1.4.8 predication()

```
std::expected< QMLExpression::Expression, std::string > iif_sadaf::talk::QMLParser::Parser↩
::predication ()
```

5.1.4.9 unary()

```
std::expected< QMLExpression::Expression, std::string > iif_sadaf::talk::QMLParser::Parser↩
::unary ()
```

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

- qml-parser/include/parser.hpp
- qml-parser/src/parser.cpp

5.2 iif_sadaf::talk::QMLParser::Token Struct Reference

Represents a single lexical token.

`#include <token.hpp>`

Public Member Functions

- Token (std::string literal, TokenType type)

  Constructs a Token.

Public Attributes

- std::string literal
- TokenType type

5.2.1 Detailed Description

Represents a single lexical token.

5.2.2 Constructor & Destructor Documentation

5.2.2.1 Token()

```
iif_sadaf::talk::QMLParser::Token::Token (
            std::string literal,
            TokenType type)
```

Constructs a Token.

Parameters

literal	The textual representation of the token.
type	The type of the token.

5.2.3 Member Data Documentation

5.2.3.1 literal

```
std::string iif_sadaf::talk::QMLParser::Token::literal
```

5.2.3.2 type

```
TokenType iif_sadaf::talk::QMLParser::Token::type
```

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

- qml-lexer/include/token.hpp
- qml-lexer/src/token.cpp

Chapter 6

File Documentation

6.1 qml-lexer/include/lexer.hpp File Reference

```
#include <string>
#include <vector>
#include "token.hpp"
```

Namespaces

- namespace iif_sadaf
- namespace iif_sadaf::talk
- namespace iif_sadaf::talk::QMLParser

Functions

- std::vector< Token > iif_sadaf::talk::QMLParser::lex (const std::string &string)

  Tokenizes a given QML formula.

6.2 lexer.hpp

Go to the documentation of this file.

```
00001 /*
00002  * SPDX-FileCopyrightText: 2024-2025 Ramiro Caso <caso.ramiro@conicet.gov.ar>
00003  *
00004  * SPDX-License-Identifier: BSD-3-Clause
00005  */
00006
00007 #pragma once
00008
00009 #include <string>
00010 #include <vector>
00011
00012 #include "token.hpp"
00013
00014 namespace iif_sadaf::talk::QMLParser {
00015
00022 std::vector<Token> lex(const std::string& string);
00023
00024 }
```

6.3 qml-lexer/include/token.hpp File Reference

```
#include <string>
```

Classes

- struct iif_sadaf::talk::QMLParser::Token

  Represents a single lexical token.

Namespaces

- namespace iif_sadaf
- namespace iif_sadaf::talk
- namespace iif_sadaf::talk::QMLParser

Enumerations

- enum class iif_sadaf::talk::QMLParser::TokenType : uint8_t {
  iif_sadaf::talk::QMLParser::NIL , iif_sadaf::talk::QMLParser::EOI , iif_sadaf::talk::QMLParser::ILLEGAL ,
  iif_sadaf::talk::QMLParser::NOT ,
  iif_sadaf::talk::QMLParser::AND , iif_sadaf::talk::QMLParser::OR , iif_sadaf::talk::QMLParser::IF ,
  iif_sadaf::talk::QMLParser::EQ ,
  iif_sadaf::talk::QMLParser::NEC , iif_sadaf::talk::QMLParser::POS , iif_sadaf::talk::QMLParser::FORALL ,
  iif_sadaf::talk::QMLParser::EXISTS ,
  iif_sadaf::talk::QMLParser::NOT_EXISTS , iif_sadaf::talk::QMLParser::ID , iif_sadaf::talk::QMLParser::NEQ ,
  iif_sadaf::talk::QMLParser::VARIABLE ,
  iif_sadaf::talk::QMLParser::IDENTIFIER , iif_sadaf::talk::QMLParser::LPAREN , iif_sadaf::talk::QMLParser::RPAREN
  , iif_sadaf::talk::QMLParser::LBRACKET ,
  iif_sadaf::talk::QMLParser::RBRACKET , iif_sadaf::talk::QMLParser::COMMA }

  Represents different types of tokens used in the QMLParser.

6.4 token.hpp

Go to the documentation of this file.

```
00001 /*
00002  * SPDX-FileCopyrightText: 2024-2025 Ramiro Caso <caso.ramiro@conicet.gov.ar>
00003  *
00004  * SPDX-License-Identifier: BSD-3-Clause
00005  */
00006
00007 #pragma once
00008
00009 #include <string>
00010
00011 namespace iif_sadaf::talk::QMLParser {
00012
00017 enum class TokenType : uint8_t {
00018     NIL, EOI, ILLEGAL,                          // parsing-related
00019     NOT, AND, OR, IF, EQ,                       // Logical operators
00020     NEC, POS,                                   // modal operators
00021     FORALL, EXISTS, NOT_EXISTS,                 // quantifiers
00022     ID, NEQ,                                    // identity and inequality
00023     VARIABLE,                                   // variables
00024     IDENTIFIER,                                 // everything else
00025     LPAREN, RPAREN, LBRACKET, RBRACKET, COMMA,  // punctuation
00026 };
00027
00032 struct Token {
00033     Token(std::string literal, TokenType type);
00034
00035     std::string literal;
00036     TokenType type;
00037 };
00038
00039 }
```

6.5 qml-lexer/src/lexer.cpp File Reference

```
#include "lexer.hpp"
#include <ranges>
#include <unordered_map>
#include <QMLExpression/expression.hpp>
```

Namespaces

- namespace iif_sadaf
- namespace iif_sadaf::talk
- namespace iif_sadaf::talk::QMLParser

Functions

- std::vector< Token > iif_sadaf::talk::QMLParser::lex (const std::string &string)

  Tokenizes a given QML formula.

6.6 qml-lexer/src/token.cpp File Reference

```
#include "token.hpp"
```

Namespaces

- namespace iif_sadaf
- namespace iif_sadaf::talk
- namespace iif_sadaf::talk::QMLParser

6.7 qml-parser/include/maps.hpp File Reference

```
#include <optional>
#include <QMLExpression/expression.hpp>
#include "lexer.hpp"
```

Namespaces

- namespace iif_sadaf
- namespace iif_sadaf::talk
- namespace iif_sadaf::talk::QMLParser

Functions

- std::optional< QMLExpression::Operator > iif_sadaf::talk::QMLParser::mapToAlethicOperator (TokenType type)

  Maps token types to alethic modal operators.
- std::optional< QMLExpression::Operator > iif_sadaf::talk::QMLParser::mapToDeonticOperator (TokenType type)

  Maps token types to deontic modal operators.
- std::optional< QMLExpression::Operator > iif_sadaf::talk::QMLParser::mapToEpistemicOperator (TokenType type)

  Maps token types to epistemic modal operators.

6.8 maps.hpp

Go to the documentation of this file.

```
00001 #pragma once
00002
00003 #include <optional>
00004
00005 #include <QMLExpression/expression.hpp>
00006
00007 #include "lexer.hpp"
00008
00009 namespace iif_sadaf::talk::QMLParser {
00010
00011 std::optional<QMLExpression::Operator> mapToAlethicOperator(TokenType type);
00012 std::optional<QMLExpression::Operator> mapToDeonticOperator(TokenType type);
00013 std::optional<QMLExpression::Operator> mapToEpistemicOperator(TokenType type);
00014
00015 }
```

6.9 qml-parser/include/parser.hpp File Reference

```
#include <expected>
#include <functional>
#include <optional>
#include <vector>
#include <string>
#include <QMLExpression/expression.hpp>
#include "lexer.hpp"
#include "maps.hpp"
#include "token.hpp"
```

Classes

- class iif_sadaf::talk::QMLParser::Parser

  Parses a sequence of tokens into a Quantified Modal Logic (QML) expression tree.

Namespaces

- namespace iif_sadaf
- namespace iif_sadaf::talk
- namespace iif_sadaf::talk::QMLParser

Functions

- std::expected< QMLExpression::Expression, std::string > iif_sadaf::talk::QMLParser::parse (const std::string &formula, Parser::ParseFunction entryPoint=&Parser::equivalence, Parser::MappingFunction mappingFunction=&mapToAlethicOperator)
- std::expected< QMLExpression::Expression, std::string > iif_sadaf::talk::QMLParser::parse (const std::string &formula, Parser::MappingFunction mappingFunction)

6.10 parser.hpp

Go to the documentation of this file.

```
00001 /*
00002  * SPDX-FileCopyrightText: 2024-2025 Ramiro Caso <caso.ramiro@conicet.gov.ar>
00003  *
00004  * SPDX-License-Identifier: BSD-3-Clause
00005  */
00006
00007 #pragma once
00008
00009 #include <expected>
00010 #include <functional>
00011 #include <optional>
00012 #include <vector>
00013 #include <string>
00014
00015 #include <QMLExpression/expression.hpp>
00016
00017 #include "lexer.hpp"
00018 #include "maps.hpp"
00019 #include "token.hpp"
00020
00021 namespace iif_sadaf::talk::QMLParser {
00022
00030 class Parser
00031 {
00032 public:
00037     using ParseFunction = std::function<std::expected<QMLExpression::Expression,
      std::string>(Parser&)>;
00038
00043     using MappingFunction = std::function<std::optional<QMLExpression::Operator>(TokenType)>;
00044
00045     Parser(const std::vector<Token>& tokens, MappingFunction mapFunc = mapToAlethicOperator);
00046     std::expected<QMLExpression::Expression, std::string> parse(ParseFunction entryPoint =
      &equivalence);
00047
00048     // rules
00049     std::expected<QMLExpression::Expression, std::string> equivalence();
00050     std::expected<QMLExpression::Expression, std::string> implication();
00051     std::expected<QMLExpression::Expression, std::string> conjunction_disjunction();
00052     std::expected<QMLExpression::Expression, std::string> unary();
00053     std::expected<QMLExpression::Expression, std::string> atomic();
00054     std::expected<QMLExpression::Expression, std::string> predication();
00055     std::expected<QMLExpression::Expression, std::string> identity();
00056     std::expected<QMLExpression::Expression, std::string> inequality();
00057
00058 private:
00059     void advance();
00060     TokenType peek(int offset = 0) const;
00061
00062     // start rule
00063     std::expected<QMLExpression::Expression, std::string> sentence(ParseFunction entryPoint =
      &equivalence);
00064
00065     int m_Index;
00066     TokenType m_LookAhead;
00067     std::vector<Token> m_TokenList;
00068     std::function<std::optional<QMLExpression::Operator>(TokenType)> m_MapToOperator;
00069 };
00070
00071 std::expected<QMLExpression::Expression, std::string> parse(const std::string& formula,
      Parser::ParseFunction entryPoint = &Parser::equivalence, Parser::MappingFunction mappingFunction =
      &mapToAlethicOperator);
00072 std::expected<QMLExpression::Expression, std::string> parse(const std::string& formula,
      Parser::MappingFunction mappingFunction);
00073
00074 }
```

6.11 qml-parser/src/maps.cpp File Reference

```
#include "maps.hpp"
```

Namespaces

- namespace iif_sadaf
- namespace iif_sadaf::talk
- namespace iif_sadaf::talk::QMLParser

Functions

- std::optional< QMLExpression::Operator > iif_sadaf::talk::QMLParser::mapToAlethicOperator (TokenType type)

  Maps token types to alethic modal operators.
- std::optional< QMLExpression::Operator > iif_sadaf::talk::QMLParser::mapToDeonticOperator (TokenType type)

  Maps token types to deontic modal operators.
- std::optional< QMLExpression::Operator > iif_sadaf::talk::QMLParser::mapToEpistemicOperator (TokenType type)

  Maps token types to epistemic modal operators.

6.12 qml-parser/src/parser.cpp File Reference

```
#include "parser.hpp"
```

Namespaces

- namespace iif_sadaf
- namespace iif_sadaf::talk
- namespace iif_sadaf::talk::QMLParser

Functions

- std::expected< QMLExpression::Expression, std::string > iif_sadaf::talk::QMLParser::parse (const std::string &formula, Parser::ParseFunction entryPoint=&Parser::equivalence, Parser::MappingFunction mapping↩Function=&mapToAlethicOperator)
- std::expected< QMLExpression::Expression, std::string > iif_sadaf::talk::QMLParser::parse (const std::string &formula, Parser::MappingFunction mappingFunction)

6.13 QMLParser/include/QMLParser.hpp File Reference

```
#include "QMLParser/lexer.hpp"
#include "QMLParser/parser.hpp"
```

6.14 QMLParser.hpp

Go to the documentation of this file.

```
00001 #pragma once
00002
00003 #include "QMLParser/lexer.hpp"
00004 #include "QMLParser/parser.hpp"
```

Index