

QMLParser

1.0

Generated by Doxygen 1.13.2

1 Namespace Index	1
1.1 Namespace List	1
2 Class Index	3
2.1 Class List	3
3 File Index	5
3.1 File List	5
4 Namespace Documentation	7
4.1 iif_sadaf Namespace Reference	7
4.2 iif_sadaf::talk Namespace Reference	7
4.3 iif_sadaf::talk::QMLParser Namespace Reference	7
4.3.1 Enumeration Type Documentation	8
4.3.1.1 TokenType	8
4.3.2 Function Documentation	9
4.3.2.1 lex()	9
4.3.2.2 mapToAlethicOperator()	9
4.3.2.3 mapToDeonticOperator()	9
4.3.2.4 mapToEpistemicOperator()	9
4.3.2.5 parse() [1/2]	10
4.3.2.6 parse() [2/2]	10
5 Class Documentation	11
5.1 iif_sadaf::talk::QMLParser::Parser Class Reference	11
5.1.1 Detailed Description	11
5.1.2 Member Typedef Documentation	12
5.1.2.1 MappingFunction	12
5.1.2.2 ParseFunction	12
5.1.3 Constructor & Destructor Documentation	12
5.1.3.1 Parser()	12
5.1.4 Member Function Documentation	12
5.1.4.1 atomic()	12
5.1.4.2 conjunction_disjunction()	12
5.1.4.3 equivalence()	12
5.1.4.4 identity()	13
5.1.4.5 implication()	13
5.1.4.6 inequality()	13
5.1.4.7 parse()	13
5.1.4.8 predication()	13
5.1.4.9 unary()	13
5.2 iif_sadaf::talk::QMLParser::Token Struct Reference	14
5.2.1 Detailed Description	14
5.2.2 Constructor & Destructor Documentation	14

5.2.2.1 Token()	14
5.2.3 Member Data Documentation	14
5.2.3.1 literal	14
5.2.3.2 type	14
6 File Documentation	15
6.1 qml-lexer/include/lexer.hpp File Reference	15
6.2 lexer.hpp	15
6.3 qml-lexer/include/token.hpp File Reference	16
6.4 token.hpp	16
6.5 qml-lexer/src/lexer.cpp File Reference	17
6.6 qml-lexer/src/token.cpp File Reference	17
6.7 qml-parser/include/maps.hpp File Reference	17
6.8 maps.hpp	18
6.9 qml-parser/include/parser.hpp File Reference	18
6.10 parser.hpp	19
6.11 qml-parser/src/maps.cpp File Reference	20
6.12 qml-parser/src/parser.cpp File Reference	20
6.13 QMLParser/include/QMLParser.hpp File Reference	20
6.14 QMLParser.hpp	20
Index	21

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

iif_sadaf	7
iif_sadaf::talk	7
iif_sadaf::talk::QMLParser	7

Chapter 2

Class Index

2.1 Class List

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

iif_sadaf::talk::QMLParser::Parser	
Parses a sequence of tokens into a Quantified Modal Logic (QML) expression tree	11
iif_sadaf::talk::QMLParser::Token	
Represents a single lexical token	14

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

qml-lexer/include/ lexer.hpp	15
qml-lexer/include/ token.hpp	16
qml-lexer/src/ lexer.cpp	17
qml-lexer/src/ token.cpp	17
qml-parser/include/ maps.hpp	17
qml-parser/include/ parser.hpp	18
qml-parser/src/ maps.cpp	20
qml-parser/src/ parser.cpp	20
QMLParser/include/ QMLParser.hpp	20

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

<i>formula</i>	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

<i>type</i>	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

<i>type</i>	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

<i>type</i>	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

<i>tokens</i>	The list of tokens to parse.
<i>mapFunc</i>	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

<i>entryPoint</i>	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

<i>literal</i>	The textual representation of the token.
<i>type</i>	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
00013     enum class TokenType : uint8_t {
00014         NIL, EOI, ILLEGAL, // parsing-related
00015         NOT, AND, OR, IF, EQ, // Logical operators
00016         NEC, POS, // modal operators
00017         FORALL, EXISTS, NOT_EXISTS, // quantifiers
00018         ID, NEQ, // identity and inequality
00019         VARIABLE, // variables
00020         IDENTIFIER, // everything else
00021         LPAREN, RPAREN, LBRACKET, RBRACKET, COMMA, // punctuation
00022     };
00023
00024     struct Token {
00025         Token(std::string literal, TokenType type);
00026
00027         std::string literal;
00028         TokenType type;
00029     };
00030 }
```

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
00023 class Parser
00024 {
00025 public:
00026     using ParseFunction = std::function<std::expected<QMLExpression::Expression,
00027 std::string>(Parser&)>;
00028
00029     using MappingFunction = std::function<std::optional<QMLExpression::Operator>(TokenType)>;
00030
00031     Parser(const std::vector<Token>& tokens, MappingFunction mapFunc = mapToAlethicOperator);
00032     std::expected<QMLExpression::Expression, std::string> parse(ParseFunction entryPoint =
00033 &equivalence);
00034
00035     // rules
00036     std::expected<QMLExpression::Expression, std::string> equivalence();
00037     std::expected<QMLExpression::Expression, std::string> implication();
00038     std::expected<QMLExpression::Expression, std::string> conjunction_disjunction();
00039     std::expected<QMLExpression::Expression, std::string> unary();
00040     std::expected<QMLExpression::Expression, std::string> atomic();
00041     std::expected<QMLExpression::Expression, std::string> predication();
00042     std::expected<QMLExpression::Expression, std::string> identity();
00043     std::expected<QMLExpression::Expression, std::string> inequality();
00044 private:
00045     void advance();
00046     TokenType peek(int offset = 0) const;
00047
00048     // start rule
00049     std::expected<QMLExpression::Expression, std::string> sentence(ParseFunction entryPoint =
00050 &equivalence);
00051
00052     int m_Index;
00053     TokenType m_LookAhead;
00054     std::vector<Token> m_TokenList;
00055     std::function<std::optional<QMLExpression::Operator>(TokenType)> m_MapToOperator;
00056 };
00057
00058 std::expected<QMLExpression::Expression, std::string> parse(const std::string& formula,
00059 Parser::ParseFunction entryPoint = &Parser::equivalence, Parser::MappingFunction mappingFunction =
00060 &mapToAlethicOperator);
00061
00062 std::expected<QMLExpression::Expression, std::string> parse(const std::string& formula,
00063 Parser::MappingFunction mappingFunction);
00064
00065 }
00066

```

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

AND

iif_sadaf::talk::QMLParser, 8

atomic

iif_sadaf::talk::QMLParser::Parser, 12

COMMA

iif_sadaf::talk::QMLParser, 8

conjunction_disjunction

iif_sadaf::talk::QMLParser::Parser, 12

EOI

iif_sadaf::talk::QMLParser, 8

EQ

iif_sadaf::talk::QMLParser, 8

equivalence

iif_sadaf::talk::QMLParser::Parser, 12

EXISTS

iif_sadaf::talk::QMLParser, 8

FORALL

iif_sadaf::talk::QMLParser, 8

ID

iif_sadaf::talk::QMLParser, 8

IDENTIFIER

iif_sadaf::talk::QMLParser, 8

identity

iif_sadaf::talk::QMLParser::Parser, 12

IF

iif_sadaf::talk::QMLParser, 8

iif_sadaf, 7

iif_sadaf::talk, 7

iif_sadaf::talk::QMLParser, 7

AND, 8

COMMA, 8

EOI, 8

EQ, 8

EXISTS, 8

FORALL, 8

ID, 8

IDENTIFIER, 8

IF, 8

ILLEGAL, 8

LBRACKET, 8

lex, 9

LPAREN, 8

mapToAlethicOperator, 9

mapToDeonticOperator, 9

mapToEpistemicOperator, 9

NEC, 8

NEQ, 8

NIL, 8

NOT, 8

NOT_EXISTS, 8

OR, 8

parse, 10

POS, 8

RBRACKET, 8

RPAREN, 8

TokenType, 8

VARIABLE, 8

iif_sadaf::talk::QMLParser::Parser, 11

atomic, 12

conjunction_disjunction, 12

equivalence, 12

identity, 12

implication, 13

inequality, 13

MappingFunction, 12

parse, 13

ParseFunction, 12

Parser, 12

predication, 13

unary, 13

iif_sadaf::talk::QMLParser::Token, 14

literal, 14

Token, 14

type, 14

ILLEGAL

iif_sadaf::talk::QMLParser, 8

implication

iif_sadaf::talk::QMLParser::Parser, 13

inequality

iif_sadaf::talk::QMLParser::Parser, 13

LBRACKET

iif_sadaf::talk::QMLParser, 8

lex

iif_sadaf::talk::QMLParser, 9

literal

iif_sadaf::talk::QMLParser::Token, 14

LPAREN

iif_sadaf::talk::QMLParser, 8

MappingFunction

iif_sadaf::talk::QMLParser::Parser, 12

mapToAlethicOperator

iif_sadaf::talk::QMLParser, 9

mapToDeonticOperator

iif_sadaf::talk::QMLParser, 9

- mapToEpistemicOperator
 - iif_sadaf::talk::QMLParser, [9](#)
- NEC
 - iif_sadaf::talk::QMLParser, [8](#)
- NEQ
 - iif_sadaf::talk::QMLParser, [8](#)
- NIL
 - iif_sadaf::talk::QMLParser, [8](#)
- NOT
 - iif_sadaf::talk::QMLParser, [8](#)
- NOT_EXISTS
 - iif_sadaf::talk::QMLParser, [8](#)
- OR
 - iif_sadaf::talk::QMLParser, [8](#)
- parse
 - iif_sadaf::talk::QMLParser, [10](#)
 - iif_sadaf::talk::QMLParser::Parser, [13](#)
- ParseFunction
 - iif_sadaf::talk::QMLParser::Parser, [12](#)
- Parser
 - iif_sadaf::talk::QMLParser::Parser, [12](#)
- POS
 - iif_sadaf::talk::QMLParser, [8](#)
- predication
 - iif_sadaf::talk::QMLParser::Parser, [13](#)
- qml-lexer/include/lexer.hpp, [15](#)
- qml-lexer/include/token.hpp, [16](#)
- qml-lexer/src/lexer.cpp, [17](#)
- qml-lexer/src/token.cpp, [17](#)
- qml-parser/include/maps.hpp, [17](#), [18](#)
- qml-parser/include/parser.hpp, [18](#), [19](#)
- qml-parser/src/maps.cpp, [20](#)
- qml-parser/src/parser.cpp, [20](#)
- QMLParser/include/QMLParser.hpp, [20](#)
- RBRACKET
 - iif_sadaf::talk::QMLParser, [8](#)
- RPAREN
 - iif_sadaf::talk::QMLParser, [8](#)
- Token
 - iif_sadaf::talk::QMLParser::Token, [14](#)
- TokenType
 - iif_sadaf::talk::QMLParser, [8](#)
- type
 - iif_sadaf::talk::QMLParser::Token, [14](#)
- unary
 - iif_sadaf::talk::QMLParser::Parser, [13](#)
- VARIABLE
 - iif_sadaf::talk::QMLParser, [8](#)