QMLExpression

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::QMLExpression Namespace Reference 7

4.3.1 Typedef Documentation 8

4.3.1.1 Expression 8

4.3.2 Enumeration Type Documentation 8

4.3.2.1 Operator 8

4.3.2.2 Quantifier 9

4.3.3 Function Documentation 9

4.3.3.1 format() 9

5 Class Documentation **11**

5.1 iif_sadaf::talk::QMLExpression::BinaryNode Struct Reference 11

5.1.1 Detailed Description 11

5.1.2 Constructor & Destructor Documentation 11

5.1.2.1 BinaryNode() 11

5.1.3 Member Data Documentation 12

5.1.3.1 lhs 12

5.1.3.2 op 12

5.1.3.3 rhs 12

5.2 iif_sadaf::talk::QMLExpression::Formatter Struct Reference 12

5.2.1 Detailed Description 12

5.2.2 Member Function Documentation 12

5.2.2.1 operator()() `[1/5]` 12

5.2.2.2 operator()() `[2/5]` 13

5.2.2.3 operator()() `[3/5]` 13

5.2.2.4 operator()() `[4/5]` 13

5.2.2.5 operator()() `[5/5]` 14

5.3 iif_sadaf::talk::QMLExpression::IdentityNode Struct Reference 14

5.3.1 Detailed Description 14

5.3.2 Constructor & Destructor Documentation 15

5.3.2.1 IdentityNode() 15

5.3.3 Member Data Documentation 15

5.3.3.1 lhs 15

5.3.3.2 rhs 15
5.4 iif_sadaf::talk::QMLExpression::PredicationNode Struct Reference 15
5.4.1 Detailed Description 15
5.4.2 Constructor & Destructor Documentation 16
5.4.2.1 PredicationNode() 16
5.4.3 Member Data Documentation 16
5.4.3.1 arguments 16
5.4.3.2 predicate 16
5.5 iif_sadaf::talk::QMLExpression::QuantificationNode Struct Reference 16
5.5.1 Detailed Description 16
5.5.2 Constructor & Destructor Documentation 17
5.5.2.1 QuantificationNode() 17
5.5.3 Member Data Documentation 17
5.5.3.1 quantifier 17
5.5.3.2 scope 17
5.5.3.3 variable 17
5.6 iif_sadaf::talk::QMLExpression::Term Struct Reference 17
5.6.1 Detailed Description 17
5.6.2 Member Enumeration Documentation 17
5.6.2.1 Type 17
5.6.3 Member Data Documentation 18
5.6.3.1 literal 18
5.6.3.2 type 18
5.7 iif_sadaf::talk::QMLExpression::UnaryNode Struct Reference 18
5.7.1 Detailed Description 18
5.7.2 Constructor & Destructor Documentation 19
5.7.2.1 UnaryNode() 19
5.7.3 Member Data Documentation 19
5.7.3.1 op 19
5.7.3.2 scope 19

6 File Documentation **21**

6.1 include/expression.hpp File Reference 21
6.2 expression.hpp 22
6.3 include/formatter.hpp File Reference 23
6.4 formatter.hpp 24
6.5 include/QMLExpression.hpp File Reference 24
6.6 QMLExpression.hpp 24
6.7 src/expression.cpp File Reference 24
6.8 src/formatter.cpp File Reference 25

Index **27**

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::QMLExpression 7

Chapter 2

Class Index

2.1 Class List

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

iif_sadaf::talk::QMLExpression::BinaryNode
Represents a binary QML formula . . . 11
iif_sadaf::talk::QMLExpression::Formatter
Provides string formatting for QML expressions . . . 12
iif_sadaf::talk::QMLExpression::IdentityNode
Represents an identity atomic formula . . . 14
iif_sadaf::talk::QMLExpression::PredicationNode
Represents a predicative atomic formula . . . 15
iif_sadaf::talk::QMLExpression::QuantificationNode
Represents a quantified QML formula . . . 16
iif_sadaf::talk::QMLExpression::Term
Represents a term in a QML expression . . . 17
iif_sadaf::talk::QMLExpression::UnaryNode
Represents a unary QML formula . . . 18

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

include/expression.hpp 21
include/formatter.hpp 23
include/QMLExpression.hpp 24
src/expression.cpp 24
src/formatter.cpp 25

Chapter 4

Namespace Documentation

4.1 iif_sadaf Namespace Reference

Namespaces

- namespace talk

4.2 iif_sadaf::talk Namespace Reference

Namespaces

- namespace QMLExpression

4.3 iif_sadaf::talk::QMLExpression Namespace Reference

Classes

- struct BinaryNode

  Represents a binary QML formula.
- struct Formatter

  Provides string formatting for QML expressions.
- struct IdentityNode

  Represents an identity atomic formula.
- struct PredicationNode

  Represents a predicative atomic formula.
- struct QuantificationNode

  Represents a quantified QML formula.
- struct Term

  Represents a term in a QML expression.
- struct UnaryNode

  Represents a unary QML formula.

Typedefs

- using Expression

    Represents a generic QML expression using a variant type.

Enumerations

- enum class Operator : uint8_t {
  NEGATION , CONJUNCTION , DISJUNCTION , CONDITIONAL ,
  CONDITIONAL_MATERIAL , CONDITIONAL_STRICT , BICONDITIONAL , ACTUALITY ,
  NECESSITY , POSSIBILITY , EPISTEMIC_NECESSITY , EPISTEMIC_POSSIBILITY ,
  DEONTIC_NECESSITY , DEONTIC_POSSIBILITY , NORMAL_NECESSITY , NORMAL_POSSIBILITY }

    Enum representing logical and modal operators.
- enum class Quantifier : uint8_t { EXISTENTIAL , UNIVERSAL }

    Enum representing quantifiers in QML.

Functions

- std::string format (const Expression &expr)

    Formats a generic QML expression as a string.

4.3.1 Typedef Documentation

4.3.1.1 Expression

```
using iif_sadaf::talk::QMLExpression::Expression
```

Initial value:

```
 std::variant<
    std::shared_ptr<UnaryNode>,
    std::shared_ptr<BinaryNode>,
    std::shared_ptr<QuantificationNode>,
    std::shared_ptr<IdentityNode>,
    std::shared_ptr<PredicationNode>
>
```

Represents a generic QML expression using a variant type.

4.3.2 Enumeration Type Documentation

4.3.2.1 Operator

```
enum class iif_sadaf::talk::QMLExpression::Operator :  uint8_t  [strong]
```

Enum representing logical and modal operators.

Enumerator

NEGATION	
CONJUNCTION	
DISJUNCTION	
CONDITIONAL	

Enumerator

CONDITIONAL_MATERIAL	
CONDITIONAL_STRICT	
BICONDITIONAL	
ACTUALITY	
NECESSITY	
POSSIBILITY	
EPISTEMIC_NECESSITY	
EPISTEMIC_POSSIBILITY	
DEONTIC_NECESSITY	
DEONTIC_POSSIBILITY	
NORMAL_NECESSITY	
NORMAL_POSSIBILITY	

4.3.2.2 Quantifier

```
enum class iif_sadaf::talk::QMLExpression::Quantifier :  uint8_t  [strong]
```

Enum representing quantifiers in QML.

Enumerator

EXISTENTIAL	
UNIVERSAL	

4.3.3 Function Documentation

4.3.3.1 format()

```
std::string iif_sadaf::talk::QMLExpression::format (
             const Expression & expr)
```

Formats a generic QML expression as a string.

Parameters

expr	The expression to format.

Returns

The formatted string representation.

Chapter 5

Class Documentation

5.1 iif_sadaf::talk::QMLExpression::BinaryNode Struct Reference

Represents a binary QML formula.

```
#include <expression.hpp>
```

Public Member Functions

- BinaryNode (Operator op, Expression lhs, Expression rhs)

Public Attributes

- Operator op
- Expression lhs
- Expression rhs

5.1.1 Detailed Description

Represents a binary QML formula.

5.1.2 Constructor & Destructor Documentation

5.1.2.1 BinaryNode()

```
iif_sadaf::talk::QMLExpression::BinaryNode::BinaryNode (
            Operator op,
            Expression lhs,
            Expression rhs)
```

5.1.3 Member Data Documentation

5.1.3.1 lhs

```
Expression iif_sadaf::talk::QMLExpression::BinaryNode::lhs
```

5.1.3.2 op

```
Operator iif_sadaf::talk::QMLExpression::BinaryNode::op
```

5.1.3.3 rhs

```
Expression iif_sadaf::talk::QMLExpression::BinaryNode::rhs
```

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

- include/expression.hpp
- src/expression.cpp

5.2 iif_sadaf::talk::QMLExpression::Formatter Struct Reference

Provides string formatting for QML expressions.

```
#include <formatter.hpp>
```

Public Member Functions

- std::string operator() (std::shared_ptr< UnaryNode > expr) const

  Formats a unary node as a string.
- std::string operator() (std::shared_ptr< BinaryNode > expr) const

  Formats a binary node as a string.
- std::string operator() (std::shared_ptr< QuantificationNode > expr) const

  Formats a quantification node as a string.
- std::string operator() (std::shared_ptr< PredicationNode > expr) const

  Formats a predication node as a string.
- std::string operator() (std::shared_ptr< IdentityNode > expr) const

  Formats an identity node as a string.

5.2.1 Detailed Description

Provides string formatting for QML expressions.

5.2.2 Member Function Documentation

5.2.2.1 operator()() [1/5]

```
std::string iif_sadaf::talk::QMLExpression::Formatter::operator() (
            std::shared_ptr< BinaryNode > expr) const
```

Formats a binary node as a string.

Parameters

expr	Shared pointer to a BinaryNode.

Returns

The formatted string representation.

5.2.2.2 operator()() [2/5]

```
std::string iif_sadaf::talk::QMLExpression::Formatter::operator() (
             std::shared_ptr< IdentityNode > expr) const
```

Formats an identity node as a string.

Parameters

expr	Shared pointer to an IdentityNode.

Returns

The formatted string representation.

5.2.2.3 operator()() [3/5]

```
std::string iif_sadaf::talk::QMLExpression::Formatter::operator() (
             std::shared_ptr< PredicationNode > expr) const
```

Formats a predication node as a string.

Parameters

expr	Shared pointer to a PredicationNode.

Returns

The formatted string representation.

5.2.2.4 operator()() [4/5]

```
std::string iif_sadaf::talk::QMLExpression::Formatter::operator() (
             std::shared_ptr< QuantificationNode > expr) const
```

Formats a quantification node as a string.

Parameters

expr	Shared pointer to a QuantificationNode.

Returns

The formatted string representation.

5.2.2.5 operator()() [5/5]

```
std::string iif_sadaf::talk::QMLExpression::Formatter::operator() (
            std::shared_ptr< UnaryNode > expr) const
```

Formats a unary node as a string.

Parameters

expr	Shared pointer to a UnaryNode.

Returns

The formatted string representation.

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

- include/formatter.hpp
- src/formatter.cpp

5.3 iif_sadaf::talk::QMLExpression::IdentityNode Struct Reference

Represents an identity atomic formula.

```
#include <expression.hpp>
```

Public Member Functions

- IdentityNode (Term lhs, Term rhs)

Public Attributes

- Term lhs
- Term rhs

5.3.1 Detailed Description

Represents an identity atomic formula.

5.3.2 Constructor & Destructor Documentation

5.3.2.1 IdentityNode()

```
iif_sadaf::talk::QMLExpression::IdentityNode::IdentityNode (
            Term lhs,
            Term rhs)
```

5.3.3 Member Data Documentation

5.3.3.1 lhs

```
Term iif_sadaf::talk::QMLExpression::IdentityNode::lhs
```

5.3.3.2 rhs

```
Term iif_sadaf::talk::QMLExpression::IdentityNode::rhs
```

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

- include/expression.hpp
- src/expression.cpp

5.4 iif_sadaf::talk::QMLExpression::PredicationNode Struct Reference

Represents a predicative atomic formula.

```
#include <expression.hpp>
```

Public Member Functions

- PredicationNode (std::string predicate, std::vector< Term > arguments)

Public Attributes

- std::string predicate
- std::vector< Term > arguments

5.4.1 Detailed Description

Represents a predicative atomic formula.

5.4.2 Constructor & Destructor Documentation

5.4.2.1 PredicationNode()

```
iif_sadaf::talk::QMLExpression::PredicationNode::PredicationNode (
            std::string predicate,
            std::vector< Term > arguments)
```

5.4.3 Member Data Documentation

5.4.3.1 arguments

```
std::vector<Term> iif_sadaf::talk::QMLExpression::PredicationNode::arguments
```

5.4.3.2 predicate

```
std::string iif_sadaf::talk::QMLExpression::PredicationNode::predicate
```

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

- include/expression.hpp
- src/expression.cpp

5.5 iif_sadaf::talk::QMLExpression::QuantificationNode Struct Reference

Represents a quantified QML formula.

```
#include <expression.hpp>
```

Public Member Functions

- QuantificationNode (Quantifier quantifier, Term variable, Expression scope)

Public Attributes

- Quantifier quantifier
- Term variable
- Expression scope

5.5.1 Detailed Description

Represents a quantified QML formula.

5.5.2 Constructor & Destructor Documentation

5.5.2.1 QuantificationNode()

```
iif_sadaf::talk::QMLExpression::QuantificationNode::QuantificationNode (
            Quantifier quantifier,
            Term variable,
            Expression scope)
```

5.5.3 Member Data Documentation

5.5.3.1 quantifier

```
Quantifier iif_sadaf::talk::QMLExpression::QuantificationNode::quantifier
```

5.5.3.2 scope

```
Expression iif_sadaf::talk::QMLExpression::QuantificationNode::scope
```

5.5.3.3 variable

```
Term iif_sadaf::talk::QMLExpression::QuantificationNode::variable
```

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

- include/expression.hpp
- src/expression.cpp

5.6 iif_sadaf::talk::QMLExpression::Term Struct Reference

Represents a term in a QML expression.

```
#include <expression.hpp>
```

Public Types

- enum class Type : uint8_t { CONSTANT , VARIABLE }

  Enum representing the type of term.

Public Attributes

- std::string literal
- Type type

5.6.1 Detailed Description

Represents a term in a QML expression.

5.6.2 Member Enumeration Documentation

5.6.2.1 Type

```
enum class iif_sadaf::talk::QMLExpression::Term::Type : uint8_t [strong]
```

Enum representing the type of term.

Enumerator

CONSTANT	
VARIABLE	

5.6.3 Member Data Documentation

5.6.3.1 literal

```
std::string iif_sadaf::talk::QMLExpression::Term::literal
```

The string representation of the term.

5.6.3.2 type

```
Type iif_sadaf::talk::QMLExpression::Term::type
```

The type of the term (constant or variable).

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

- include/expression.hpp

5.7 iif_sadaf::talk::QMLExpression::UnaryNode Struct Reference

Represents a unary QML formula.

```
#include <expression.hpp>
```

Public Member Functions

- UnaryNode (Operator op, Expression scope)

Public Attributes

- Operator op
- Expression scope

5.7.1 Detailed Description

Represents a unary QML formula.

5.7.2 Constructor & Destructor Documentation

5.7.2.1 UnaryNode()

```
iif_sadaf::talk::QMLExpression::UnaryNode::UnaryNode (
            Operator op,
            Expression scope)
```

5.7.3 Member Data Documentation

5.7.3.1 op

```
Operator iif_sadaf::talk::QMLExpression::UnaryNode::op
```

5.7.3.2 scope

```
Expression iif_sadaf::talk::QMLExpression::UnaryNode::scope
```

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

- include/expression.hpp
- src/expression.cpp

Chapter 6

File Documentation

6.1 include/expression.hpp File Reference

```
#include <memory>
#include <string>
#include <variant>
#include <vector>
```

Classes

- struct iif_sadaf::talk::QMLExpression::Term

  Represents a term in a QML expression.
- struct iif_sadaf::talk::QMLExpression::UnaryNode

  Represents a unary QML formula.
- struct iif_sadaf::talk::QMLExpression::BinaryNode

  Represents a binary QML formula.
- struct iif_sadaf::talk::QMLExpression::QuantificationNode

  Represents a quantified QML formula.
- struct iif_sadaf::talk::QMLExpression::IdentityNode

  Represents an identity atomic formula.
- struct iif_sadaf::talk::QMLExpression::PredicationNode

  Represents a predicative atomic formula.

Namespaces

- namespace iif_sadaf
- namespace iif_sadaf::talk
- namespace iif_sadaf::talk::QMLExpression

Typedefs

- using iif_sadaf::talk::QMLExpression::Expression

  Represents a generic QML expression using a variant type.

Enumerations

- enum class iif_sadaf::talk::QMLExpression::Operator : uint8_t {
  iif_sadaf::talk::QMLExpression::NEGATION , iif_sadaf::talk::QMLExpression::CONJUNCTION , iif_sadaf::talk::QMLExpression:
  , iif_sadaf::talk::QMLExpression::CONDITIONAL ,
  iif_sadaf::talk::QMLExpression::CONDITIONAL_MATERIAL , iif_sadaf::talk::QMLExpression::CONDITIONAL_STRICT
  , iif_sadaf::talk::QMLExpression::BICONDITIONAL , iif_sadaf::talk::QMLExpression::ACTUALITY ,
  iif_sadaf::talk::QMLExpression::NECESSITY , iif_sadaf::talk::QMLExpression::POSSIBILITY , iif_sadaf::talk::QMLExpression::E
  , iif_sadaf::talk::QMLExpression::EPISTEMIC_POSSIBILITY ,
  iif_sadaf::talk::QMLExpression::DEONTIC_NECESSITY , iif_sadaf::talk::QMLExpression::DEONTIC_POSSIBILITY
  , iif_sadaf::talk::QMLExpression::NORMAL_NECESSITY , iif_sadaf::talk::QMLExpression::NORMAL_POSSIBILITY
  }

  Enum representing logical and modal operators.
- enum class iif_sadaf::talk::QMLExpression::Quantifier : uint8_t { iif_sadaf::talk::QMLExpression::EXISTENTIAL
  , iif_sadaf::talk::QMLExpression::UNIVERSAL }

  Enum representing quantifiers in QML.

6.2 expression.hpp

Go to the documentation of this file.

```
/*
 * SPDX-FileCopyrightText: 2024-2025 Ramiro Caso <caso.ramiro@conicet.gov.ar>
 *
 * SPDX-License-Identifier: BSD-3-Clause
 */

#pragma once

#include <memory>
#include <string>
#include <variant>
#include <vector>

namespace iif_sadaf::talk::QMLExpression {

struct Term {
    enum class Type : uint8_t {
        CONSTANT,
        VARIABLE
    };

    std::string literal;
    Type type;
};

struct UnaryNode;
struct BinaryNode;
struct QuantificationNode;
struct IdentityNode;
struct PredicationNode;

using Expression = std::variant<
    std::shared_ptr<UnaryNode>,
    std::shared_ptr<BinaryNode>,
    std::shared_ptr<QuantificationNode>,
    std::shared_ptr<IdentityNode>,
    std::shared_ptr<PredicationNode>
>;

enum class Operator : uint8_t {
    NEGATION,
    CONJUNCTION, DISJUNCTION,
    CONDITIONAL, CONDITIONAL_MATERIAL, CONDITIONAL_STRICT,
    BICONDITIONAL,
    ACTUALITY,
    NECESSITY, POSSIBILITY,
    EPISTEMIC_NECESSITY, EPISTEMIC_POSSIBILITY,
    DEONTIC_NECESSITY, DEONTIC_POSSIBILITY,
    NORMAL_NECESSITY, NORMAL_POSSIBILITY,
};

enum class Quantifier : uint8_t {
    EXISTENTIAL,
    UNIVERSAL
};


struct UnaryNode {
    UnaryNode(Operator op, Expression scope);

    Operator op;
    Expression scope;
};

struct BinaryNode {
    BinaryNode(Operator op, Expression lhs, Expression rhs);

    Operator op;
    Expression lhs;
    Expression rhs;
};

struct QuantificationNode {
    QuantificationNode(Quantifier quantifier, Term variable, Expression scope);

    Quantifier quantifier;
    Term variable;
    Expression scope;
};

struct IdentityNode {
    IdentityNode(Term lhs, Term rhs);

    Term lhs;
    Term rhs;
};

struct PredicationNode {
    PredicationNode(std::string predicate, std::vector<Term> arguments);

    std::string predicate;
    std::vector<Term> arguments;
};

}
```

6.3 include/formatter.hpp File Reference

```
#include "expression.hpp"
#include <string>
```

Classes

- struct iif_sadaf::talk::QMLExpression::Formatter

  Provides string formatting for QML expressions.

Namespaces

- namespace iif_sadaf
- namespace iif_sadaf::talk
- namespace iif_sadaf::talk::QMLExpression

Functions

- std::string iif_sadaf::talk::QMLExpression::format (const Expression &expr)

  Formats a generic QML expression as a string.

6.4 formatter.hpp

Go to the documentation of this file.

```
/*
 * SPDX-FileCopyrightText: 2024-2025 Ramiro Caso <caso.ramiro@conicet.gov.ar>
 *
 * SPDX-License-Identifier: BSD-3-Clause
 */

#include "expression.hpp"

#include <string>

namespace iif_sadaf::talk::QMLExpression {

struct Formatter {
    std::string operator()(std::shared_ptr<UnaryNode> expr) const;
    std::string operator()(std::shared_ptr<BinaryNode> expr) const;
    std::string operator()(std::shared_ptr<QuantificationNode> expr) const;
    std::string operator()(std::shared_ptr<PredicationNode> expr) const;
    std::string operator()(std::shared_ptr<IdentityNode> expr) const;
};

std::string format(const Expression& expr);

}
```

6.5 include/QMLExpression.hpp File Reference

```
#include "expression.hpp"
#include "formatter.hpp"
```

6.6 QMLExpression.hpp

Go to the documentation of this file.

```
#pragma once

#include "expression.hpp"
#include "formatter.hpp"
```

6.7 src/expression.cpp File Reference

```
#include "expression.hpp"
```

Namespaces

- namespace iif_sadaf
- namespace iif_sadaf::talk
- namespace iif_sadaf::talk::QMLExpression

6.8 src/formatter.cpp File Reference

```
#include "formatter.hpp"
#include <format>
#include <unordered_map>
```

Namespaces

- namespace iif_sadaf
- namespace iif_sadaf::talk
- namespace iif_sadaf::talk::QMLExpression

Functions

- std::string iif_sadaf::talk::QMLExpression::format (const Expression &expr)

  Formats a generic QML expression as a string.

Index

ACTUALITY
  iif_sadaf::talk::QMLExpression, 9
arguments
  iif_sadaf::talk::QMLExpression::PredicationNode, 16

BICONDITIONAL
  iif_sadaf::talk::QMLExpression, 9
BinaryNode
  iif_sadaf::talk::QMLExpression::BinaryNode, 11

CONDITIONAL
  iif_sadaf::talk::QMLExpression, 8
CONDITIONAL_MATERIAL
  iif_sadaf::talk::QMLExpression, 9
CONDITIONAL_STRICT
  iif_sadaf::talk::QMLExpression, 9
CONJUNCTION
  iif_sadaf::talk::QMLExpression, 8
CONSTANT
  iif_sadaf::talk::QMLExpression::Term, 18

DEONTIC_NECESSITY
  iif_sadaf::talk::QMLExpression, 9
DEONTIC_POSSIBILITY
  iif_sadaf::talk::QMLExpression, 9
DISJUNCTION
  iif_sadaf::talk::QMLExpression, 8

EPISTEMIC_NECESSITY
  iif_sadaf::talk::QMLExpression, 9
EPISTEMIC_POSSIBILITY
  iif_sadaf::talk::QMLExpression, 9
EXISTENTIAL
  iif_sadaf::talk::QMLExpression, 9
Expression
  iif_sadaf::talk::QMLExpression, 8

format
  iif_sadaf::talk::QMLExpression, 9

IdentityNode
  iif_sadaf::talk::QMLExpression::IdentityNode, 15
iif_sadaf, 7
iif_sadaf::talk, 7
iif_sadaf::talk::QMLExpression, 7
  ACTUALITY, 9
  BICONDITIONAL, 9
  CONDITIONAL, 8
  CONDITIONAL_MATERIAL, 9
  CONDITIONAL_STRICT, 9
  CONJUNCTION, 8
  DEONTIC_NECESSITY, 9
  DEONTIC_POSSIBILITY, 9
  DISJUNCTION, 8
  EPISTEMIC_NECESSITY, 9
  EPISTEMIC_POSSIBILITY, 9
  EXISTENTIAL, 9
  Expression, 8
  format, 9
  NECESSITY, 9
  NEGATION, 8
  NORMAL_NECESSITY, 9
  NORMAL_POSSIBILITY, 9
  Operator, 8
  POSSIBILITY, 9
  Quantifier, 9
  UNIVERSAL, 9
iif_sadaf::talk::QMLExpression::BinaryNode, 11
  BinaryNode, 11
  lhs, 12
  op, 12
  rhs, 12
iif_sadaf::talk::QMLExpression::Formatter, 12
  operator(), 12–14
iif_sadaf::talk::QMLExpression::IdentityNode, 14
  IdentityNode, 15
  lhs, 15
  rhs, 15
iif_sadaf::talk::QMLExpression::PredicationNode, 15
  arguments, 16
  predicate, 16
  PredicationNode, 16
iif_sadaf::talk::QMLExpression::QuantificationNode, 16
  QuantificationNode, 17
  quantifier, 17
  scope, 17
  variable, 17
iif_sadaf::talk::QMLExpression::Term, 17
  CONSTANT, 18
  literal, 18
  Type, 17
  type, 18
  VARIABLE, 18
iif_sadaf::talk::QMLExpression::UnaryNode, 18
  op, 19
  scope, 19
  UnaryNode, 19
include/expression.hpp, 21, 22
include/formatter.hpp, 23, 24

include/QMLExpression.hpp, 24

lhs
    iif_sadaf::talk::QMLExpression::BinaryNode, 12
    iif_sadaf::talk::QMLExpression::IdentityNode, 15
literal
    iif_sadaf::talk::QMLExpression::Term, 18

NECESSITY
    iif_sadaf::talk::QMLExpression, 9
NEGATION
    iif_sadaf::talk::QMLExpression, 8
NORMAL_NECESSITY
    iif_sadaf::talk::QMLExpression, 9
NORMAL_POSSIBILITY
    iif_sadaf::talk::QMLExpression, 9

op
    iif_sadaf::talk::QMLExpression::BinaryNode, 12
    iif_sadaf::talk::QMLExpression::UnaryNode, 19
Operator
    iif_sadaf::talk::QMLExpression, 8
operator()
    iif_sadaf::talk::QMLExpression::Formatter, 12–14

POSSIBILITY
    iif_sadaf::talk::QMLExpression, 9
predicate
    iif_sadaf::talk::QMLExpression::PredicationNode, 16
PredicationNode
    iif_sadaf::talk::QMLExpression::PredicationNode, 16

QuantificationNode
    iif_sadaf::talk::QMLExpression::QuantificationNode, 17
Quantifier
    iif_sadaf::talk::QMLExpression, 9
quantifier
    iif_sadaf::talk::QMLExpression::QuantificationNode, 17

rhs
    iif_sadaf::talk::QMLExpression::BinaryNode, 12
    iif_sadaf::talk::QMLExpression::IdentityNode, 15

scope
    iif_sadaf::talk::QMLExpression::QuantificationNode, 17
    iif_sadaf::talk::QMLExpression::UnaryNode, 19
src/expression.cpp, 24
src/formatter.cpp, 25

Type
    iif_sadaf::talk::QMLExpression::Term, 17
type
    iif_sadaf::talk::QMLExpression::Term, 18

UnaryNode
    iif_sadaf::talk::QMLExpression::UnaryNode, 19
UNIVERSAL
    iif_sadaf::talk::QMLExpression, 9

VARIABLE
    iif_sadaf::talk::QMLExpression::Term, 18
variable
    iif_sadaf::talk::QMLExpression::QuantificationNode, 17