

MSDscript

Generated by Doxygen 1.8.17

1 Hierarchical Index	1
1.1 Class Hierarchy	1
2 Class Index	3
2.1 Class List	3
3 Class Documentation	5
3.1 AddExpr Class Reference	5
3.1.1 Detailed Description	5
3.1.2 Constructor & Destructor Documentation	6
3.1.2.1 AddExpr()	6
3.1.3 Member Function Documentation	6
3.1.3.1 contains_var()	6
3.1.3.2 equals()	6
3.1.3.3 expr_print()	7
3.1.3.4 interp()	7
3.1.3.5 optimize()	7
3.2 BoolExpr Class Reference	8
3.2.1 Detailed Description	8
3.2.2 Constructor & Destructor Documentation	8
3.2.2.1 BoolExpr()	8
3.2.3 Member Function Documentation	9
3.2.3.1 contains_var()	9
3.2.3.2 equals()	9
3.2.3.3 expr_print()	9
3.2.3.4 interp()	10
3.2.3.5 optimize()	10
3.3 BoolVal Class Reference	11
3.3.1 Detailed Description	11
3.3.2 Constructor & Destructor Documentation	11
3.3.2.1 BoolVal()	11
3.4 CallExpr Class Reference	12
3.4.1 Detailed Description	12
3.4.2 Constructor & Destructor Documentation	12
3.4.2.1 CallExpr()	12
3.4.3 Member Function Documentation	13
3.4.3.1 contains_var()	13
3.4.3.2 equals()	13
3.4.3.3 expr_print()	13
3.4.3.4 interp()	14
3.4.3.5 optimize()	14
3.5 EqualExpr Class Reference	15
3.5.1 Detailed Description	15

3.5.2 Constructor & Destructor Documentation	15
3.5.2.1 EqualExpr()	15
3.5.3 Member Function Documentation	16
3.5.3.1 contains_var()	16
3.5.3.2 equals()	16
3.5.3.3 expr_print()	16
3.5.3.4 interp()	17
3.5.3.5 optimize()	17
3.6 Expr Class Reference	17
3.7 FunExpr Class Reference	18
3.7.1 Detailed Description	19
3.7.2 Constructor & Destructor Documentation	19
3.7.2.1 FunExpr()	19
3.7.3 Member Function Documentation	19
3.7.3.1 contains_var()	19
3.7.3.2 equals()	20
3.7.3.3 expr_print()	20
3.7.3.4 interp()	20
3.7.3.5 optimize()	21
3.8 FunVal Class Reference	21
3.8.1 Detailed Description	22
3.8.2 Constructor & Destructor Documentation	22
3.8.2.1 FunVal()	22
3.9 IfExpr Class Reference	22
3.9.1 Detailed Description	23
3.9.2 Constructor & Destructor Documentation	23
3.9.2.1 IfExpr()	23
3.9.3 Member Function Documentation	23
3.9.3.1 contains_var()	23
3.9.3.2 equals()	24
3.9.3.3 expr_print()	24
3.9.3.4 interp()	24
3.9.3.5 optimize()	25
3.10 LetExpr Class Reference	25
3.10.1 Detailed Description	26
3.10.2 Constructor & Destructor Documentation	26
3.10.2.1 LetExpr()	26
3.10.3 Member Function Documentation	26
3.10.3.1 contains_var()	26
3.10.3.2 equals()	27
3.10.3.3 expr_print()	27
3.10.3.4 interp()	27

3.10.3.5 optimize()	28
3.11 MultExpr Class Reference	28
3.11.1 Detailed Description	29
3.11.2 Constructor & Destructor Documentation	29
3.11.2.1 MultExpr()	29
3.11.3 Member Function Documentation	29
3.11.3.1 contains_var()	29
3.11.3.2 equals()	29
3.11.3.3 expr_print()	30
3.11.3.4 interp()	30
3.11.3.5 optimize()	31
3.12 NumExpr Class Reference	31
3.12.1 Detailed Description	31
3.12.2 Constructor & Destructor Documentation	32
3.12.2.1 NumExpr()	32
3.12.3 Member Function Documentation	32
3.12.3.1 contains_var()	32
3.12.3.2 equals()	32
3.12.3.3 expr_print()	33
3.12.3.4 interp()	33
3.12.3.5 optimize()	33
3.13 NumVal Class Reference	34
3.13.1 Detailed Description	34
3.13.2 Constructor & Destructor Documentation	34
3.13.2.1 NumVal()	34
3.14 Step Class Reference	35
3.14.1 Detailed Description	35
3.14.2 Member Function Documentation	35
3.14.2.1 interp_by_steps()	35
3.15 Val Class Reference	36
3.16 VarExpr Class Reference	36
3.16.1 Detailed Description	36
3.16.2 Constructor & Destructor Documentation	37
3.16.2.1 VarExpr()	37
3.16.3 Member Function Documentation	37
3.16.3.1 contains_var()	37
3.16.3.2 equals()	37
3.16.3.3 expr_print()	38
3.16.3.4 interp()	38
3.16.3.5 optimize()	38

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

enable_shared_from_this	
Expr	17
AddExpr	5
BoolExpr	8
CallExpr	12
EqualExpr	15
FunExpr	18
IfExpr	22
LetExpr	25
MultExpr	28
NumExpr	31
VarExpr	36
Val	36
BoolVal	11
FunVal	21
NumVal	34
Step	35

Chapter 2

Class Index

2.1 Class List

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

AddExpr	5
BoolExpr	8
BoolVal	11
CallExpr	12
EqualExpr	15
Expr	17
FunExpr	18
FunVal	21
IfExpr	22
LetExpr	25
MultExpr	28
NumExpr	31
NumVal	34
Step	35
Val	36
VarExpr	36

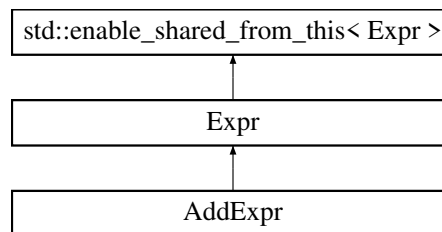
Chapter 3

Class Documentation

3.1 AddExpr Class Reference

```
#include <Expr.hpp>
```

Inheritance diagram for AddExpr:



Public Member Functions

- [AddExpr](#) (std::shared_ptr< [Expr](#) > lhs, std::shared_ptr< [Expr](#) > rhs)
- bool [equals](#) (std::shared_ptr< [Expr](#) > e)
- std::shared_ptr< [Val](#) > [interp](#) (std::shared_ptr< Env > env)
- bool [contains_var](#) ()
- std::shared_ptr< [Expr](#) > [optimize](#) ()
- std::string [expr_print](#) ()

Public Attributes

- std::shared_ptr< [Expr](#) > [lhs](#)
- std::shared_ptr< [Expr](#) > [rhs](#)

3.1.1 Detailed Description

The [AddExpr](#) class is used to represent the addition of two separate [Expr](#)'s.

3.1.2 Constructor & Destructor Documentation

3.1.2.1 AddExpr()

```
AddExpr::AddExpr (
    std::shared_ptr< Expr > lhs,
    std::shared_ptr< Expr > rhs )
```

Construct an [AddExpr](#) from two [Expr](#)'s.

Parameters

<i>lhs</i>	left hand side Expr .
<i>rhs</i>	right hand side Expr .

3.1.3 Member Function Documentation

3.1.3.1 contains_var()

```
bool AddExpr::contains_var ( ) [virtual]
```

Determines if this [Expr](#) contains a variable and returns a boolean.

Returns

true if [Expr](#) contains variable, false otherwise.

Implements [Expr](#).

3.1.3.2 equals()

```
bool AddExpr::equals (
    std::shared_ptr< Expr > e ) [virtual]
```

Return boolean specifying if this [Expr](#) is equal to the [Expr](#) passed in as parameter e.

Parameters

<i>e</i>	Expr to compare.
----------	----------------------------------

Returns

true if expressions are equal, false otherwise.

Implements [Expr](#).

3.1.3.3 `expr_print()`

```
std::string AddExpr::expr_print ( ) [virtual]
```

Returns the [Expr](#) in a human readable string.

Returns

String of the [Expr](#).

Implements [Expr](#).

3.1.3.4 `interp()`

```
std::shared_ptr< Val > AddExpr::interp (
    std::shared_ptr< Env > env ) [virtual]
```

Evaluates the [Expr](#) and returns a [Val](#). Takes an Env as parameter e. /exception If the evaluation reaches a free variable, an error will be thrown.

Parameters

<i>env</i>	
------------	--

Returns

[Val](#) representing the [Expr](#) solution or a semantically equivalent value.

Implements [Expr](#).

3.1.3.5 `optimize()`

```
std::shared_ptr< Expr > AddExpr::optimize ( ) [virtual]
```

Evaluates the [Expr](#) and returns a semantically equivalent [Expr](#) that is no larger than the input [Expr](#). Unlike the `interp` method, `optimize` will not throw an error if the evaluations reaches a free variable.

Returns

[Expr](#) representing the most optimized solution or a semantically equivalent [Expr](#).

Implements [Expr](#).

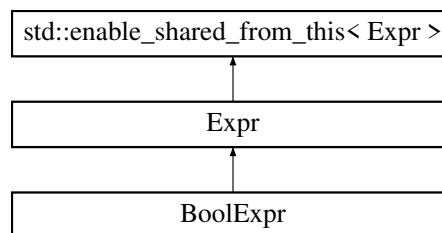
The documentation for this class was generated from the following file:

- [Expr.hpp](#)

3.2 BoolExpr Class Reference

```
#include <Expr.hpp>
```

Inheritance diagram for BoolExpr:



Public Member Functions

- [BoolExpr](#) (bool rep)
- bool [equals](#) (std::shared_ptr< [Expr](#) > e)
- std::shared_ptr< [Val](#) > [interp](#) (std::shared_ptr< Env > env)
- bool [contains_var](#) ()
- std::shared_ptr< [Expr](#) > [optimize](#) ()
- std::string [expr_print](#) ()

Public Attributes

- bool [rep](#)

3.2.1 Detailed Description

The [BoolExpr](#) class is used to represent a boolean as an [Expr](#).

3.2.2 Constructor & Destructor Documentation

3.2.2.1 BoolExpr()

```
BoolExpr::BoolExpr (
    bool rep )
```

Construct a [BoolExpr](#) from a bool.

Parameters

<i>rep</i>	bool to be represented by this BoolExpr .
------------	---

3.2.3 Member Function Documentation

3.2.3.1 contains_var()

```
bool BoolExpr::contains_var ( ) [virtual]
```

Determines if this [Expr](#) contains a variable and returns a boolean.

Returns

true if [Expr](#) contains variable, false otherwise.

Implements [Expr](#).

3.2.3.2 equals()

```
bool BoolExpr::equals (
    std::shared_ptr< Expr > e ) [virtual]
```

Return boolean specifying if this [Expr](#) is equal to the [Expr](#) passed in as parameter e.

Parameters

<i>e</i>	Expr to compare.
----------	----------------------------------

Returns

true if expressions are equal, false otherwise.

Implements [Expr](#).

3.2.3.3 expr_print()

```
std::string BoolExpr::expr_print ( ) [virtual]
```

Returns the [Expr](#) in a human readable string.

Returns

String of the [Expr](#).

Implements [Expr](#).

3.2.3.4 interp()

```
std::shared_ptr< Val > BoolExpr::interp (
    std::shared_ptr< Env > env ) [virtual]
```

Evaluates the [Expr](#) and returns a [Val](#). Takes an Env as parameter e. /exception If the evaluation reaches a free variable, an error will be thrown.

Parameters

<i>env</i>	
------------	--

Returns

[Val](#) representing the [Expr](#) solution or a semantically equivalent value.

Implements [Expr](#).

3.2.3.5 optimize()

```
std::shared_ptr< Expr > BoolExpr::optimize ( ) [virtual]
```

Evaluates the [Expr](#) and returns a semantically equivalent [Expr](#) that is no larger than the input [Expr](#). Unlike the `interp` method, `optimize` will not throw an error if the evaluations reaches a free variable.

Returns

[Expr](#) representing the most optimized solution or a semantically equivalent [Expr](#).

Implements [Expr](#).

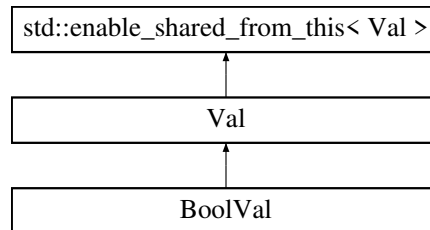
The documentation for this class was generated from the following file:

- [Expr.hpp](#)

3.3 BoolVal Class Reference

```
#include <value.hpp>
```

Inheritance diagram for BoolVal:



Public Member Functions

- [BoolVal](#) (bool rep)

Public Attributes

- bool rep

3.3.1 Detailed Description

Stores a bool. Can be returned from the `interp()` method of an [Expr](#). The actual bool can be accessed via the `rep` member variable.

3.3.2 Constructor & Destructor Documentation

3.3.2.1 BoolVal()

```
BoolVal::BoolVal (
    bool rep )
```

Constructs a [BoolVal](#) with the provided bool.

Parameters

<i>rep</i>	bool to be stored in BoolVal .
------------	--

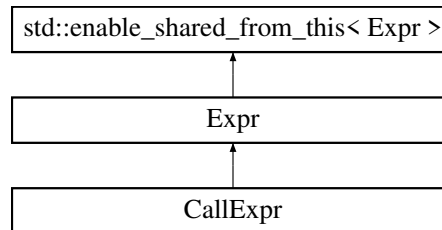
The documentation for this class was generated from the following file:

- value.hpp

3.4 CallExpr Class Reference

```
#include <Expr.hpp>
```

Inheritance diagram for CallExpr:



Public Member Functions

- `CallExpr` (`std::shared_ptr< Expr > to_be_called`, `std::shared_ptr< Expr > actual_argument`)
- `bool equals` (`std::shared_ptr< Expr > e`)
- `std::shared_ptr< Val > interp` (`std::shared_ptr< Env > env`)
- `bool contains_var` ()
- `std::shared_ptr< Expr > optimize` ()
- `std::string expr_print` ()

Public Attributes

- `std::shared_ptr< Expr > to_be_called`
- `std::shared_ptr< Expr > actual_argument`

3.4.1 Detailed Description

The `CallExpr` class is used to represent the calling of one `Expr` and passing in another `Expr`. This class is useful when paired with the `FunExpr` class.

3.4.2 Constructor & Destructor Documentation

3.4.2.1 CallExpr()

```
CallExpr::CallExpr (
    std::shared_ptr< Expr > to_be_called,
    std::shared_ptr< Expr > actual_argument )
```

Construct a `CallExpr` with two `Expr`'s.

Parameters

<i>to_be_called</i>	Expr
<i>actual_argument</i>	Expr

3.4.3 Member Function Documentation

3.4.3.1 contains_var()

```
bool CallExpr::contains_var ( ) [virtual]
```

Determines if this [Expr](#) contains a variable and returns a boolean.

Returns

true if [Expr](#) contains variable, false otherwise.

Implements [Expr](#).

3.4.3.2 equals()

```
bool CallExpr::equals (
    std::shared_ptr< Expr > e ) [virtual]
```

Return boolean specifying if this [Expr](#) is equal to the [Expr](#) passed in as parameter e.

Parameters

e	Expr to compare.
---	----------------------------------

Returns

true if expressions are equal, false otherwise.

Implements [Expr](#).

3.4.3.3 expr_print()

```
std::string CallExpr::expr_print ( ) [virtual]
```

Returns the [Expr](#) in a human readable string.

Returns

String of the [Expr](#).

Implements [Expr](#).

3.4.3.4 interp()

```
std::shared_ptr< Val > CallExpr::interp (
    std::shared_ptr< Env > env ) [virtual]
```

Evaluates the [Expr](#) and returns a [Val](#). Takes an Env as parameter e. /exception If the evaluation reaches a free variable, an error will be thrown.

Parameters

<i>env</i>	
------------	--

Returns

[Val](#) representing the [Expr](#) solution or a semantically equivalent value.

Implements [Expr](#).

3.4.3.5 optimize()

```
std::shared_ptr< Expr > CallExpr::optimize ( ) [virtual]
```

Evaluates the [Expr](#) and returns a semantically equivalent [Expr](#) that is no larger than the input [Expr](#). Unlike the `interp` method, `optimize` will not throw an error if the evaluations reaches a free variable.

Returns

[Expr](#) representing the most optimized solution or a semantically equivalent [Expr](#).

Implements [Expr](#).

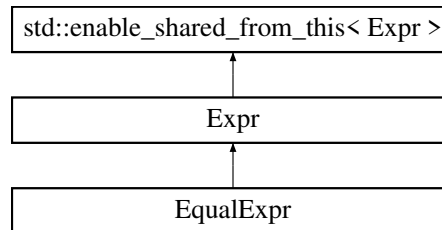
The documentation for this class was generated from the following file:

- [Expr.hpp](#)

3.5 EqualExpr Class Reference

```
#include <Expr.hpp>
```

Inheritance diagram for EqualExpr:



Public Member Functions

- `EqualExpr` (`std::shared_ptr< Expr > lhs`, `std::shared_ptr< Expr > rhs`)
- `bool equals` (`std::shared_ptr< Expr > e`)
- `std::shared_ptr< Val > interp` (`std::shared_ptr< Env > env`)
- `bool contains_var` ()
- `std::shared_ptr< Expr > optimize` ()
- `std::string expr_print` ()

Public Attributes

- `std::shared_ptr< Expr > rhs`
- `std::shared_ptr< Expr > lhs`

3.5.1 Detailed Description

The `EqualExpr` class is used to represent a comparison of two `Expr`'s to determine equality.

3.5.2 Constructor & Destructor Documentation

3.5.2.1 EqualExpr()

```
EqualExpr::EqualExpr (
    std::shared_ptr< Expr > lhs,
    std::shared_ptr< Expr > rhs )
```

Construct an `EqualExpr` with two `Expr`'s.

Parameters

<i>lhs</i>	left hand side <code>Expr</code> .
<i>rhs</i>	right hand side <code>Expr</code> .

3.5.3 Member Function Documentation

3.5.3.1 contains_var()

```
bool EqualExpr::contains_var ( ) [virtual]
```

Determines if this [Expr](#) contains a variable and returns a boolean.

Returns

true if [Expr](#) contains variable, false otherwise.

Implements [Expr](#).

3.5.3.2 equals()

```
bool EqualExpr::equals (
    std::shared_ptr< Expr > e ) [virtual]
```

Return boolean specifying if this [Expr](#) is equal to the [Expr](#) passed in as parameter e.

Parameters

e	Expr to compare.
---	----------------------------------

Returns

true if expressions are equal, false otherwise.

Implements [Expr](#).

3.5.3.3 expr_print()

```
std::string EqualExpr::expr_print ( ) [virtual]
```

Returns the [Expr](#) in a human readable string.

Returns

String of the [Expr](#).

Implements [Expr](#).

3.5.3.4 interp()

```
std::shared_ptr< Val > EqualExpr::interp (
    std::shared_ptr< Env > env ) [virtual]
```

Evaluates the [Expr](#) and returns a [Val](#). Takes an [Env](#) as parameter *e*. /exception If the evaluation reaches a free variable, an error will be thrown.

Parameters

<i>env</i>	
------------	--

Returns

[Val](#) representing the [Expr](#) solution or a semantically equivalent value.

Implements [Expr](#).

3.5.3.5 optimize()

```
std::shared_ptr< Expr > EqualExpr::optimize ( ) [virtual]
```

Evaluates the [Expr](#) and returns a semantically equivalent [Expr](#) that is no larger than the input [Expr](#). Unlike the `interp` method, `optimize` will not throw an error if the evaluations reaches a free variable.

Returns

[Expr](#) representing the most optimized solution or a semantically equivalent [Expr](#).

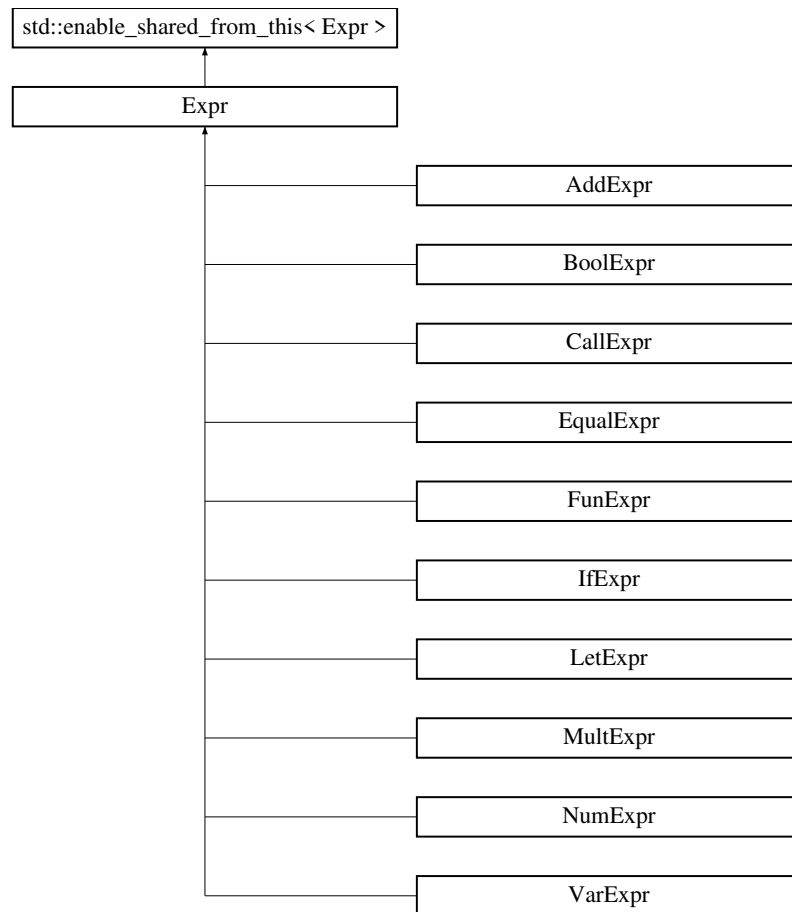
Implements [Expr](#).

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

- [Expr.hpp](#)

3.6 Expr Class Reference

Inheritance diagram for [Expr](#):



Public Member Functions

- virtual bool **equals** (std::shared_ptr< [Expr](#) > e)=0
- virtual std::shared_ptr< [Val](#) > **interp** (std::shared_ptr< Env > env)=0
- virtual bool **contains_var** ()=0
- virtual std::shared_ptr< [Expr](#) > **optimize** ()=0
- virtual std::string **expr_print** ()=0

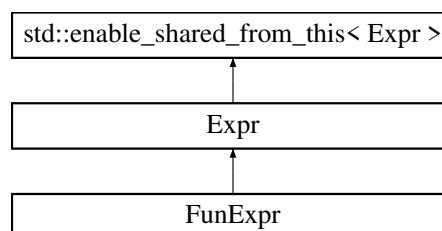
The documentation for this class was generated from the following file:

- Expr.hpp

3.7 FunExpr Class Reference

```
#include <Expr.hpp>
```

Inheritance diagram for FunExpr:



Public Member Functions

- [FunExpr](#) (std::string formal_arg, std::shared_ptr< [Expr](#) > actual_arg)
- bool [equals](#) (std::shared_ptr< [Expr](#) > e)
- std::shared_ptr< [Val](#) > [interp](#) (std::shared_ptr< Env > env)
- bool [contains_var](#) ()
- std::shared_ptr< [Expr](#) > [optimize](#) ()
- std::string [expr_print](#) ()

Public Attributes

- std::string **formal_arg**
- std::shared_ptr< [Expr](#) > **actual_arg**

3.7.1 Detailed Description

The [FunExpr](#) class is used to represent user defined functions. This class can be used in conjunction with the [CallExpr](#) class to implement function calls.

3.7.2 Constructor & Destructor Documentation

3.7.2.1 FunExpr()

```
FunExpr::FunExpr (
    std::string formal_arg,
    std::shared_ptr< Expr > actual_arg )
```

Construct a [FunExpr](#) with the formal_arg represented by a string and the actual_arg represented by an [Expr](#).

Parameters

<i>formal_arg</i>	string
<i>actual_arg</i>	Expr representing the body of the FunExpr .

3.7.3 Member Function Documentation

3.7.3.1 contains_var()

```
bool FunExpr::contains_var ( ) [virtual]
```

Determines if this [Expr](#) contains a variable and returns a boolean.

Returns

true if [Expr](#) contains variable, false otherwise.

Implements [Expr](#).

3.7.3.2 equals()

```
bool FunExpr::equals (
    std::shared_ptr< Expr > e ) [virtual]
```

Return boolean specifying if this [Expr](#) is equal to the [Expr](#) passed in as parameter e.

Parameters

<i>e</i>	Expr to compare.
----------	----------------------------------

Returns

true if expressions are equal, false otherwise.

Implements [Expr](#).

3.7.3.3 expr_print()

```
std::string FunExpr::expr_print ( ) [virtual]
```

Returns the [Expr](#) in a human readable string.

Returns

String of the [Expr](#).

Implements [Expr](#).

3.7.3.4 interp()

```
std::shared_ptr< Val > FunExpr::interp (
    std::shared_ptr< Env > env ) [virtual]
```

Evaluates the [Expr](#) and returns a [Val](#). Takes an Env as parameter e. /exception If the evaluation reaches a free variable, an error will be thrown.

Parameters

<i>env</i>	
------------	--

Returns

[Val](#) representing the [Expr](#) solution or a semantically equivalent value.

Implements [Expr](#).

3.7.3.5 optimize()

```
std::shared_ptr< Expr > FunExpr::optimize ( ) [virtual]
```

Evaluates the [Expr](#) and returns a semantically equivalent [Expr](#) that is no larger than the input [Expr](#). Unlike the `interp` method, `optimize` will not throw an error if the evaluations reaches a free variable.

Returns

[Expr](#) representing the most optimized solution or a semantically equivalent [Expr](#).

Implements [Expr](#).

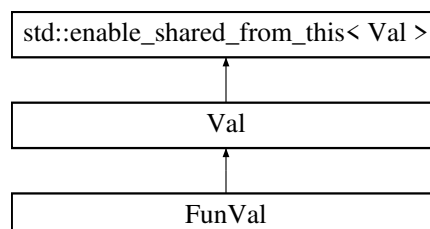
The documentation for this class was generated from the following file:

- `Expr.hpp`

3.8 FunVal Class Reference

```
#include <value.hpp>
```

Inheritance diagram for FunVal:



Public Member Functions

- [FunVal](#) (std::string formal_arg, std::shared_ptr< [Expr](#) > body, std::shared_ptr< Env > env)

Public Attributes

- `std::string formal_arg`
- `std::shared_ptr< Expr > body`
- `std::shared_ptr< Env > env`

3.8.1 Detailed Description

Stores the components of a function. Can be returned from the `interp()` method of an [Expr](#).

3.8.2 Constructor & Destructor Documentation

3.8.2.1 FunVal()

```
FunVal::FunVal (
    std::string formal_arg,
    std::shared_ptr< Expr > body,
    std::shared_ptr< Env > env )
```

Constructs a [FunVal](#) from a string `formal_arg`, [Expr](#) `body`, and `Env` `env`.

Parameters

<i>formal_arg</i>	string representing the <code>formal_arg</code> .
<i>body</i>	Expr representing the actual function.
<i>env</i>	<code>Env</code> to pass along into the FunVal .

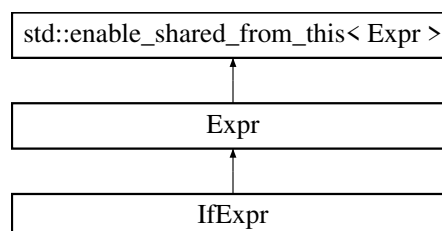
The documentation for this class was generated from the following file:

- `value.hpp`

3.9 IfExpr Class Reference

```
#include <Expr.hpp>
```

Inheritance diagram for `IfExpr`:



Public Member Functions

- `IfExpr` (`std::shared_ptr< Expr > test_part`, `std::shared_ptr< Expr > then_part`, `std::shared_ptr< Expr > else_part`)
- `bool equals` (`std::shared_ptr< Expr > e`)
- `std::shared_ptr< Val > interp` (`std::shared_ptr< Env > env`)
- `bool contains_var` ()
- `std::shared_ptr< Expr > optimize` ()
- `std::string expr_print` ()

Public Attributes

- `std::shared_ptr< Expr > test_part`
- `std::shared_ptr< Expr > then_part`
- `std::shared_ptr< Expr > else_part`

3.9.1 Detailed Description

The `IfExpr` class is used to represent an if else then statement.

3.9.2 Constructor & Destructor Documentation

3.9.2.1 IfExpr()

```
IfExpr::IfExpr (
    std::shared_ptr< Expr > test_part,
    std::shared_ptr< Expr > then_part,
    std::shared_ptr< Expr > else_part )
```

Construct an `IfExpr` consisting of three `Expr`'s: `test_part`, `then_part`, and `else_part`.

Parameters

<i>test_part</i>	<code>Expr</code> that when evaluated determines if <code>then_part</code> or <code>else_part</code> is evaluated.
<i>then_part</i>	<code>Expr</code> that can be evaluated if <code>test_part</code> is true.
<i>else_part</i>	<code>Expr</code> that can be evaluated if <code>else_part</code> is true.

3.9.3 Member Function Documentation

3.9.3.1 contains_var()

```
bool IfExpr::contains_var ( ) [virtual]
```

Determines if this [Expr](#) contains a variable and returns a boolean.

Returns

true if [Expr](#) contains variable, false otherwise.

Implements [Expr](#).

3.9.3.2 equals()

```
bool IfExpr::equals (
    std::shared_ptr< Expr > e ) [virtual]
```

Return boolean specifying if this [Expr](#) is equal to the [Expr](#) passed in as parameter e.

Parameters

<i>e</i>	Expr to compare.
----------	----------------------------------

Returns

true if expressions are equal, false otherwise.

Implements [Expr](#).

3.9.3.3 expr_print()

```
std::string IfExpr::expr_print ( ) [virtual]
```

Returns the [Expr](#) in a human readable string.

Returns

String of the [Expr](#).

Implements [Expr](#).

3.9.3.4 interp()

```
std::shared_ptr< Val > IfExpr::interp (
    std::shared_ptr< Env > env ) [virtual]
```

Evaluates the [Expr](#) and returns a [Val](#). Takes an Env as parameter e. /exception If the evaluation reaches a free variable, an error will be thrown.

Parameters

<i>env</i>	
------------	--

Returns

[Val](#) representing the [Expr](#) solution or a semantically equivalent value.

Implements [Expr](#).

3.9.3.5 optimize()

```
std::shared_ptr< Expr > IfExpr::optimize ( ) [virtual]
```

Evaluates the [Expr](#) and returns a semantically equivalent [Expr](#) that is no larger than the input [Expr](#). Unlike the `interp` method, `optimize` will not throw an error if the evaluations reaches a free variable.

Returns

[Expr](#) representing the most optimized solution or a semantically equivalent [Expr](#).

Implements [Expr](#).

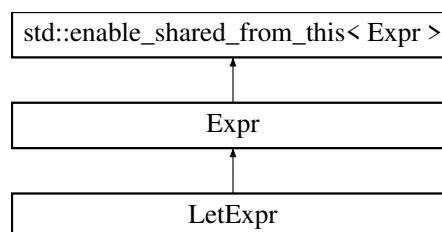
The documentation for this class was generated from the following file:

- `Expr.hpp`

3.10 LetExpr Class Reference

```
#include <Expr.hpp>
```

Inheritance diagram for LetExpr:



Public Member Functions

- [LetExpr](#) (std::string name, std::shared_ptr< [Expr](#) > var_val, std::shared_ptr< [Expr](#) > in_expr)
- bool [equals](#) (std::shared_ptr< [Expr](#) > e)
- std::shared_ptr< [Val](#) > [interp](#) (std::shared_ptr< Env > env)
- bool [contains_var](#) ()
- std::shared_ptr< [Expr](#) > [optimize](#) ()
- std::string [expr_print](#) ()

Public Attributes

- `std::string name`
- `std::shared_ptr< Expr > var_val`
- `std::shared_ptr< Expr > in_expr`

3.10.1 Detailed Description

The `LetExpr` class is used to assign an `Expr` to a variable within another `Expr`. For example, `_let x = 5 _in x + 1`

3.10.2 Constructor & Destructor Documentation

3.10.2.1 LetExpr()

```
LetExpr::LetExpr (
    std::string name,
    std::shared_ptr< Expr > var_val,
    std::shared_ptr< Expr > in_expr )
```

Construct an `LetExpr` using a string to represent the variable, an `Expr` representing the value of that variable, and another `Expr` representing the body of the `LetExpr`.

Parameters

<i>name</i>	string representing the variable.
<i>var_val</i>	<code>Expr</code> representing the value of variable.
<i>in_expr</i>	<code>Expr</code> representing the body of the <code>LetExpr</code> .

3.10.3 Member Function Documentation

3.10.3.1 contains_var()

```
bool LetExpr::contains_var ( ) [virtual]
```

Determines if this `Expr` contains a variable and returns a boolean.

Returns

true if `Expr` contains variable, false otherwise.

Implements `Expr`.

3.10.3.2 equals()

```
bool LetExpr::equals (
    std::shared_ptr< Expr > e ) [virtual]
```

Return boolean specifying if this [Expr](#) is equal to the [Expr](#) passed in as parameter e.

Parameters

<i>e</i>	Expr to compare.
----------	----------------------------------

Returns

true if expressions are equal, false otherwise.

Implements [Expr](#).

3.10.3.3 expr_print()

```
std::string LetExpr::expr_print ( ) [virtual]
```

Returns the [Expr](#) in a human readable string.

Returns

String of the [Expr](#).

Implements [Expr](#).

3.10.3.4 interp()

```
std::shared_ptr< Val > LetExpr::interp (
    std::shared_ptr< Env > env ) [virtual]
```

Evaluates the [Expr](#) and returns a [Val](#). Takes an Env as parameter e. /exception If the evaluation reaches a free variable, an error will be thrown.

Parameters

<i>env</i>	
------------	--

Returns

[Val](#) representing the [Expr](#) solution or a semantically equivalent value.

Implements [Expr](#).

3.10.3.5 optimize()

```
std::shared_ptr< Expr > LetExpr::optimize ( ) [virtual]
```

Evaluates the [Expr](#) and returns a semantically equivalent [Expr](#) that is no larger than the input [Expr](#). Unlike the `interp` method, `optimize` will not throw an error if the evaluations reaches a free variable.

Returns

[Expr](#) representing the most optimized solution or a semantically equivalent [Expr](#).

Implements [Expr](#).

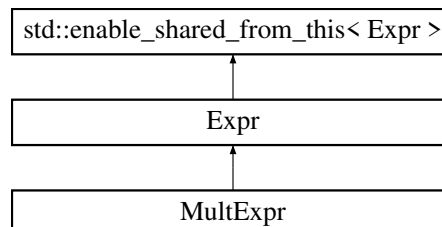
The documentation for this class was generated from the following file:

- `Expr.hpp`

3.11 MultExpr Class Reference

```
#include <Expr.hpp>
```

Inheritance diagram for MultExpr:



Public Member Functions

- [MultExpr](#) (std::shared_ptr< [Expr](#) > lhs, std::shared_ptr< [Expr](#) > rhs)
- bool [equals](#) (std::shared_ptr< [Expr](#) > e)
- std::shared_ptr< [Val](#) > [interp](#) (std::shared_ptr< Env > env)
- bool [contains_var](#) ()
- std::shared_ptr< [Expr](#) > [optimize](#) ()
- std::string [expr_print](#) ()

Public Attributes

- std::shared_ptr< [Expr](#) > **lhs**
- std::shared_ptr< [Expr](#) > **rhs**

3.11.1 Detailed Description

The `MultExpr` class is used to represent the multiplication of two separate `Expr`'s.

3.11.2 Constructor & Destructor Documentation

3.11.2.1 MultExpr()

```
MultExpr::MultExpr (
    std::shared_ptr< Expr > lhs,
    std::shared_ptr< Expr > rhs )
```

Construct an `MultExpr` from two `Expr`'s.

Parameters

<i>lhs</i>	left hand side <code>Expr</code> .
<i>rhs</i>	right hand side <code>Expr</code> .

3.11.3 Member Function Documentation

3.11.3.1 contains_var()

```
bool MultExpr::contains_var ( ) [virtual]
```

Determines if this `Expr` contains a variable and returns a boolean.

Returns

true if `Expr` contains variable, false otherwise.

Implements `Expr`.

3.11.3.2 equals()

```
bool MultExpr::equals (
    std::shared_ptr< Expr > e ) [virtual]
```

Return boolean specifying if this `Expr` is equal to the `Expr` passed in as parameter `e`.

Parameters

<i>e</i>	Expr to compare.
----------	----------------------------------

Returns

true if expressions are equal, false otherwise.

Implements [Expr](#).

3.11.3.3 `expr_print()`

```
std::string MultExpr::expr_print ( ) [virtual]
```

Returns the [Expr](#) in a human readable string.

Returns

String of the [Expr](#).

Implements [Expr](#).

3.11.3.4 `interp()`

```
std::shared_ptr< Val > MultExpr::interp (
    std::shared_ptr< Env > env ) [virtual]
```

Evaluates the [Expr](#) and returns a [Val](#). Takes an Env as parameter *e*. /exception If the evaluation reaches a free variable, an error will be thrown.

Parameters

<i>env</i>	
------------	--

Returns

[Val](#) representing the [Expr](#) solution or a semantically equivalent value.

Implements [Expr](#).

3.11.3.5 optimize()

```
std::shared_ptr< Expr > MultExpr::optimize ( ) [virtual]
```

Evaluates the `Expr` and returns a semantically equivalent `Expr` that is no larger than the input `Expr`. Unlike the `interp` method, `optimize` will not throw an error if the evaluations reaches a free variable.

Returns

`Expr` representing the most optimized solution or a semantically equivalent `Expr`.

Implements `Expr`.

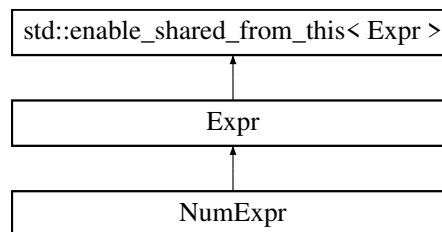
The documentation for this class was generated from the following file:

- Expr.hpp

3.12 NumExpr Class Reference

```
#include <Expr.hpp>
```

Inheritance diagram for NumExpr:



Public Member Functions

- `NumExpr` (int rep)
- bool `equals` (std::shared_ptr< Expr > e)
- std::shared_ptr< Val > `interp` (std::shared_ptr< Env > env)
- bool `contains_var` ()
- std::shared_ptr< Expr > `optimize` ()
- std::string `expr_print` ()

Public Attributes

- int `rep`
- std::shared_ptr< Val > `val`

3.12.1 Detailed Description

The `NumExpr` class represents an integer as an `Expr`.

3.12.2 Constructor & Destructor Documentation

3.12.2.1 NumExpr()

```
NumExpr::NumExpr (
    int rep )
```

Construct a [NumExpr](#) from an int.

Parameters

<i>rep</i>	int to be represented by this NumExpr .
------------	---

3.12.3 Member Function Documentation

3.12.3.1 contains_var()

```
bool NumExpr::contains_var ( ) [virtual]
```

Determines if this [Expr](#) contains a variable and returns a boolean.

Returns

true if [Expr](#) contains variable, false otherwise.

Implements [Expr](#).

3.12.3.2 equals()

```
bool NumExpr::equals (
    std::shared_ptr< Expr > e ) [virtual]
```

Return boolean specifying if this [Expr](#) is equal to the [Expr](#) passed in as parameter e.

Parameters

<i>e</i>	Expr to compare.
----------	----------------------------------

Returns

true if expressions are equal, false otherwise.

Implements [Expr](#).

3.12.3.3 `expr_print()`

```
std::string NumExpr::expr_print ( ) [virtual]
```

Returns the [Expr](#) in a human readable string.

Returns

String of the [Expr](#).

Implements [Expr](#).

3.12.3.4 `interp()`

```
std::shared_ptr< Val > NumExpr::interp (
    std::shared_ptr< Env > env ) [virtual]
```

Evaluates the [Expr](#) and returns a [Val](#). Takes an Env as parameter e. /exception If the evaluation reaches a free variable, an error will be thrown.

Parameters

<i>env</i>	
------------	--

Returns

[Val](#) representing the [Expr](#) solution or a semantically equivalent value.

Implements [Expr](#).

3.12.3.5 `optimize()`

```
std::shared_ptr< Expr > NumExpr::optimize ( ) [virtual]
```

Evaluates the [Expr](#) and returns a semantically equivalent [Expr](#) that is no larger than the input [Expr](#). Unlike the `interp` method, `optimize` will not throw an error if the evaluations reaches a free variable.

Returns

[Expr](#) representing the most optimized solution or a semantically equivalent [Expr](#).

Implements [Expr](#).

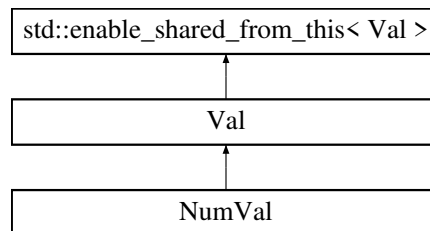
The documentation for this class was generated from the following file:

- Expr.hpp

3.13 NumVal Class Reference

```
#include <value.hpp>
```

Inheritance diagram for NumVal:

**Public Member Functions**

- [NumVal](#) (int rep)

Public Attributes

- int rep

3.13.1 Detailed Description

Stores an int. Can be returned from the `interp()` method of an [Expr](#). The actual int can be accessed via the `rep` member variable.

3.13.2 Constructor & Destructor Documentation

3.13.2.1 NumVal()

```
NumVal::NumVal (  
    int rep )
```

Constructs a [NumVal](#) with the provided int.

Parameters

<i>rep</i>	int to be stored in NumVal .
------------	--

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

- value.hpp

3.14 Step Class Reference

```
#include <Step.hpp>
```

Static Public Member Functions

- static std::shared_ptr< [Val](#) > [interp_by_steps](#) (std::shared_ptr< [Expr](#) > e)

3.14.1 Detailed Description

The [Step](#) introduces the static method `interp_by_steps`.

3.14.2 Member Function Documentation

3.14.2.1 `interp_by_steps()`

```
static std::shared_ptr< Val > Step::interp_by_steps (
    std::shared_ptr< Expr > e ) [static]
```

Evaluates the [Expr](#) and returns a [Val](#). Differs from the standard `interp` as this method allows the solving of deeply recursive functions without causing a stack overflow. /exception If the evaluation reaches a free variable, an error will be thrown.

Parameters

<i>e</i>	Expr to be evaluated.
----------	---------------------------------------

Returns

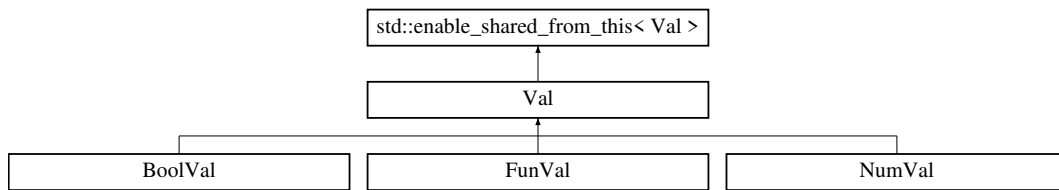
[Val](#) representing the [Expr](#) solution or a semantically equivalent value.

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

- Step.hpp

3.15 Val Class Reference

Inheritance diagram for Val:



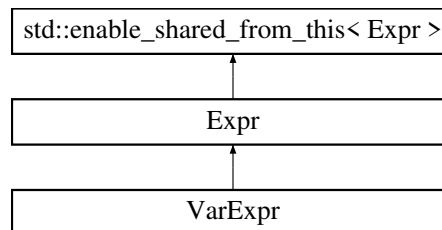
The documentation for this class was generated from the following file:

- value.hpp

3.16 VarExpr Class Reference

```
#include <Expr.hpp>
```

Inheritance diagram for VarExpr:



Public Member Functions

- [VarExpr](#) (std::string val)
- bool [equals](#) (std::shared_ptr< [Expr](#) > e)
- std::shared_ptr< [Val](#) > [interp](#) (std::shared_ptr< Env > env)
- bool [contains_var](#) ()
- std::shared_ptr< [Expr](#) > [optimize](#) ()
- std::string [expr_print](#) ()

Public Attributes

- std::string **name**

3.16.1 Detailed Description

The [VarExpr](#) class represents a variable as an [Expr](#).

3.16.2 Constructor & Destructor Documentation

3.16.2.1 VarExpr()

```
VarExpr::VarExpr (
    std::string val )
```

Construct a [VarExpr](#) from an string.

Parameters

<i>val</i>	string variable.
------------	------------------

3.16.3 Member Function Documentation

3.16.3.1 contains_var()

```
bool VarExpr::contains_var ( ) [virtual]
```

Determines if this [Expr](#) contains a variable and returns a boolean.

Returns

true if [Expr](#) contains variable, false otherwise.

Implements [Expr](#).

3.16.3.2 equals()

```
bool VarExpr::equals (
    std::shared_ptr< Expr > e ) [virtual]
```

Return boolean specifying if this [Expr](#) is equal to the [Expr](#) passed in as parameter e.

Parameters

<i>e</i>	Expr to compare.
----------	----------------------------------

Returns

true if expressions are equal, false otherwise.

Implements [Expr](#).

3.16.3.3 `expr_print()`

```
std::string VarExpr::expr_print ( ) [virtual]
```

Returns the [Expr](#) in a human readable string.

Returns

String of the [Expr](#).

Implements [Expr](#).

3.16.3.4 `interp()`

```
std::shared_ptr< Val > VarExpr::interp (
    std::shared_ptr< Env > env ) [virtual]
```

Evaluates the [Expr](#) and returns a [Val](#). Takes an Env as parameter e. /exception If the evaluation reaches a free variable, an error will be thrown.

Parameters

<i>env</i>	
------------	--

Returns

[Val](#) representing the [Expr](#) solution or a semantically equivalent value.

Implements [Expr](#).

3.16.3.5 `optimize()`

```
std::shared_ptr< Expr > VarExpr::optimize ( ) [virtual]
```

Evaluates the [Expr](#) and returns a semantically equivalent [Expr](#) that is no larger than the input [Expr](#). Unlike the `interp` method, `optimize` will not throw an error if the evaluations reaches a free variable.

Returns

[Expr](#) representing the most optimized solution or a semantically equivalent [Expr](#).

Implements [Expr](#).

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

- Expr.hpp

Index

AddExpr, [5](#)
 AddExpr, [6](#)
 contains_var, [6](#)
 equals, [6](#)
 expr_print, [7](#)
 interp, [7](#)
 optimize, [7](#)

BoolExpr, [8](#)
 BoolExpr, [8](#)
 contains_var, [9](#)
 equals, [9](#)
 expr_print, [9](#)
 interp, [10](#)
 optimize, [10](#)

BoolVal, [11](#)
 BoolVal, [11](#)

CallExpr, [12](#)
 CallExpr, [12](#)
 contains_var, [13](#)
 equals, [13](#)
 expr_print, [13](#)
 interp, [14](#)
 optimize, [14](#)

contains_var
 AddExpr, [6](#)
 BoolExpr, [9](#)
 CallExpr, [13](#)
 EqualExpr, [16](#)
 FunExpr, [19](#)
 IfExpr, [23](#)
 LetExpr, [26](#)
 MultExpr, [29](#)
 NumExpr, [32](#)
 VarExpr, [37](#)

EqualExpr, [15](#)
 contains_var, [16](#)
 EqualExpr, [15](#)
 equals, [16](#)
 expr_print, [16](#)
 interp, [16](#)
 optimize, [17](#)

equals
 AddExpr, [6](#)
 BoolExpr, [9](#)
 CallExpr, [13](#)
 EqualExpr, [16](#)
 FunExpr, [20](#)

IfExpr, [24](#)
 LetExpr, [26](#)
 MultExpr, [29](#)
 NumExpr, [32](#)
 VarExpr, [37](#)

Expr, [17](#)
 expr_print
 AddExpr, [7](#)
 BoolExpr, [9](#)
 CallExpr, [13](#)
 EqualExpr, [16](#)
 FunExpr, [20](#)
 IfExpr, [24](#)
 LetExpr, [27](#)
 MultExpr, [30](#)
 NumExpr, [33](#)
 VarExpr, [38](#)

FunExpr, [18](#)
 contains_var, [19](#)
 equals, [20](#)
 expr_print, [20](#)
 FunExpr, [19](#)
 interp, [20](#)
 optimize, [21](#)

FunVal, [21](#)
 FunVal, [22](#)

IfExpr, [22](#)
 contains_var, [23](#)
 equals, [24](#)
 expr_print, [24](#)
 IfExpr, [23](#)
 interp, [24](#)
 optimize, [25](#)

interp
 AddExpr, [7](#)
 BoolExpr, [10](#)
 CallExpr, [14](#)
 EqualExpr, [16](#)
 FunExpr, [20](#)
 IfExpr, [24](#)
 LetExpr, [27](#)
 MultExpr, [30](#)
 NumExpr, [33](#)
 VarExpr, [38](#)

interp_by_steps
 Step, [35](#)

LetExpr, [25](#)

- [contains_var](#), [26](#)
 - [equals](#), [26](#)
 - [expr_print](#), [27](#)
 - [interp](#), [27](#)
 - [LetExpr](#), [26](#)
 - [optimize](#), [28](#)
- [MultExpr](#), [28](#)
 - [contains_var](#), [29](#)
 - [equals](#), [29](#)
 - [expr_print](#), [30](#)
 - [interp](#), [30](#)
 - [MultExpr](#), [29](#)
 - [optimize](#), [30](#)
- [NumExpr](#), [31](#)
 - [contains_var](#), [32](#)
 - [equals](#), [32](#)
 - [expr_print](#), [33](#)
 - [interp](#), [33](#)
 - [NumExpr](#), [32](#)
 - [optimize](#), [33](#)
- [NumVal](#), [34](#)
 - [NumVal](#), [34](#)
- [optimize](#)
 - [AddExpr](#), [7](#)
 - [BoolExpr](#), [10](#)
 - [CallExpr](#), [14](#)
 - [EqualExpr](#), [17](#)
 - [FunExpr](#), [21](#)
 - [IfExpr](#), [25](#)
 - [LetExpr](#), [28](#)
 - [MultExpr](#), [30](#)
 - [NumExpr](#), [33](#)
 - [VarExpr](#), [38](#)
- [Step](#), [35](#)
 - [interp_by_steps](#), [35](#)
- [Val](#), [36](#)
- [VarExpr](#), [36](#)
 - [contains_var](#), [37](#)
 - [equals](#), [37](#)
 - [expr_print](#), [38](#)
 - [interp](#), [38](#)
 - [optimize](#), [38](#)
 - [VarExpr](#), [37](#)