Gauss Symbolic Library

Generated by Doxygen 1.8.17

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all documented namespaces with brief descriptions:

Chapter 2

File Index

2.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 3

Namespace Documentation

3.1 gauss Namespace Reference

Typedefs

- typedef alg::expr **expr**
- typedef alg::kind **kind**

Functions

- std::string toString (expr a)

  Return a string corresponding to a given expression.
- std::string toLatex (expr a, bool print_as_fractions, unsigned long max_den)

  Construct a latex representation of a given expression.

3.1.1 Detailed Description

TODO: add list support. TODO: add set support.

3.1.2 Function Documentation

3.1.2.1 toLatex()

```
std::string gauss::toLatex (
            expr a,
            bool print_as_fractions,
            unsigned long max_den )
```

Construct a latex representation of a given expression.

Parameters

in	*a*	A expression.
in	*useFractions*	If true, print rational numbers as fractions.
in	*maxDenominators*	This is the maximum denominator for a fraction representing a number between [0, 1], bigger the number, bigger the precision on representing double precision floating points. Because of the nature of floating arithmetic, you may not always want this number as big as it can be.

Returns

A string representing a expression on latex format.

Examples

/home/runner/work/gauss/gauss/gauss/Gauss.hpp.

3.1.2.2 toString()

```
std::string gauss::toString (
            expr a )
```

Return a string corresponding to a given expression.

Parameters

a	A algebraic expression.

Returns

A human friendly string representation of a given expression.

Examples

/home/runner/work/gauss/gauss/gauss/Gauss.hpp.

Chapter 4

File Documentation

4.1 gauss/Gauss.hpp File Reference

Header of the main API of this libary.

```
#include "Algebra/Expression.hpp"
#include "gauss/Algebra/Matrix.hpp"
#include <array>
```

Include dependency graph for Gauss.hpp:

Chapter 5

Example Documentation

5.1 /home/runner/work/gauss/gauss/gauss/Gauss.hpp

Return the degree expression of a power expression.Compute the degree of a power expression. if the expression is not a power, an error is raised.

Parameters

in	*a*	A power expression.

expr x = symbol("x"); expr t = pow(x, 2); assert(isEqual(powDegree(t), 2));

Returns

The degree of the power expression.

```
#include "Algebra/Expression.hpp"
#include "gauss/Algebra/Matrix.hpp"
#include <array>
#include <cstddef>
#include <string>
namespace gauss {
typedef alg::expr expr;
typedef alg::kind kind;
namespace algebra {
expr numberFromDouble(double v);
expr numberFromString(std::string v);
expr intFromString(const char *v);
expr intFromLong(long v);
expr symbol(std::string v);
expr pow(expr a, expr e);
expr &getOperand(expr a, size_t i);
void setOperand(expr &a, size_t i, expr b);
kind kindOf(expr a);
bool is(expr a, int k);
expr root(expr a, expr b);
expr sqrt(expr a);
bool isEqual(expr a, expr b);
expr powDegree(expr a);
expr powBase(expr a);
expr rootIndex(expr a);
expr rootRadicand(expr a);
expr numerator(expr a);
expr denominator(expr b);
expr add(expr a, expr b);
expr sub(expr a, expr b);
expr mul(expr a, expr b);
expr div(expr a, expr b);
expr expand(expr a);
expr reduce(expr a);
```

```
expr log(expr x, expr base);
expr exp(expr x);
expr abs(expr x);
expr ln(expr x);
expr replace(expr u, expr x, expr v);
expr eval(expr u, expr x, expr v);
expr freeVariables(expr u);
expr prime(size_t i);
expr primeFactors(expr a);
namespace trigonometry {
expr sinh(expr x);
expr cosh(expr x);
expr tanh(expr x);
expr cos(expr x);
expr sin(expr x);
expr tan(expr x);
expr csc(expr x);
expr cot(expr x);
expr sec(expr x);
expr coth(expr x);
expr sech(expr x);
expr csch(expr x);
expr arccos(expr x);
expr arcsin(expr x);
expr arctan(expr x);
expr arccot(expr x);
expr arcsec(expr x);
expr arccsc(expr x);
expr arccosh(expr x);
expr arctanh(expr x);
}; // namespace trigonometry
namespace linear {
expr matrix(unsigned l, unsigned c);
expr identity(unsigned l, unsigned c);
expr matrixGet(expr A, unsigned i, unsigned j);
void matrixSet(expr A, unsigned i, unsigned j, double a);
expr svd(expr A);
expr inverse(expr A);
expr det(expr A);
expr transpose(expr A);
expr solveLinear(expr A, expr b);
} // namespace linear
} // namespace algebra
namespace polynomial {
expr degreePoly(expr f, expr x);
expr coefficientPoly(expr f, expr x,
                                expr d);
expr leadingCoefficientPoly(expr f, expr x);
expr rootsOfPoly(expr a);
expr factorPoly(expr poly);
expr resultantOfPoly(expr a, expr b);
expr addPoly(expr a, expr b);
expr subPoly(expr a, expr b);
expr mulPoly(expr a, expr b);
expr divPoly(expr a, expr b);
expr quoPoly(expr a, expr b);
expr remPoly(expr a, expr b);
expr gcdPoly(expr a, expr b);
expr lcmPoly(expr a, expr b);
namespace finiteField {
expr projectPolyFiniteField(expr a, long long p);
expr addPolyFiniteField(expr a, expr b, long long p);
expr subPolyFiniteField(expr a, expr b, long long p);
expr mulPolyFiniteField(expr a, expr b, long long p);
expr divPolyFiniteField(expr a, expr b, long long p);
expr quoPolyFiniteField(expr a, expr b, long long p);
expr remPolyFiniteField(expr a, expr b, long long p);
} // namespace finiteField
} // namespace polynomial
namespace calculus {
expr derivative(expr a, expr x);
} // namespace calculus
std::string toString(expr a);
std::string toLatex(expr a, bool print_as_fractions,
                    unsigned long max_den);
} // namespace gauss
```

5.2 expand

Expand a expression.Expand and reduce an expression.

(x(3x + 4)) = 3x^2 + 4x.

Returns

A algebraic expression corresponding to the expansion of the expression 'a'.

5.3 reduce

Reduce an expression.Reduce a expression to the smallest possible form not regarding algebraix equalities or expansions. That means that it perform the operations of a given expression.

Parameters

in	*a*	An algebraic expression.

(3x + 4y^2 + 5x + (3x + 3y^2)) = 11x + 7y^2'

Returns

The reduced form of 'a'

Index