GiNaC

Jens Vollinga

Institut für Physik

Universität Mainz

GiNaC Overview

- Developed since 1999
- Current version GiNaC 1.3.0
- C++ library
  $\longrightarrow$ Algebra becomes part of the C++ language
- Small command-line interface: *ginsh*

Outline

- libginac vs. *ginsh*
  (our first GiNaC program)
- Fundamentals of GiNaC
- Features step-by-step
- GiNaC for physicists

Programming GiNaC

- Write a C++ program, e.g. firstexample.cpp:

```
#include <iostream>
using namespace std;
#include <ginac/ginac.h>
using namespace GiNaC;

int main()
{
    symbol x("x");
    ex result = Li(2,x).diff(x);
    cout << result << endl;
    return 0;
}
```

Programming GiNaC

- Write a C++ program, e.g. firstexample.cpp:

```
#include <iostream>
using namespace std;
#include <ginac/ginac.h>
using namespace GiNaC;

int main()
{
    symbol x("x");
    ex result = Li(2,x).diff(x);
    cout << result << endl;
    return 0;
}
```

Library header

Programming GiNaC

- Write a C++ program, e.g. firstexample.cpp:

```
#include <iostream>
using namespace std;
#include <ginac/ginac.h>
using namespace GiNaC;

int main()
{
    symbol x("x");
    ex result = Li(2,x).diff(x);
    cout << result << endl;
    return 0;
}
```

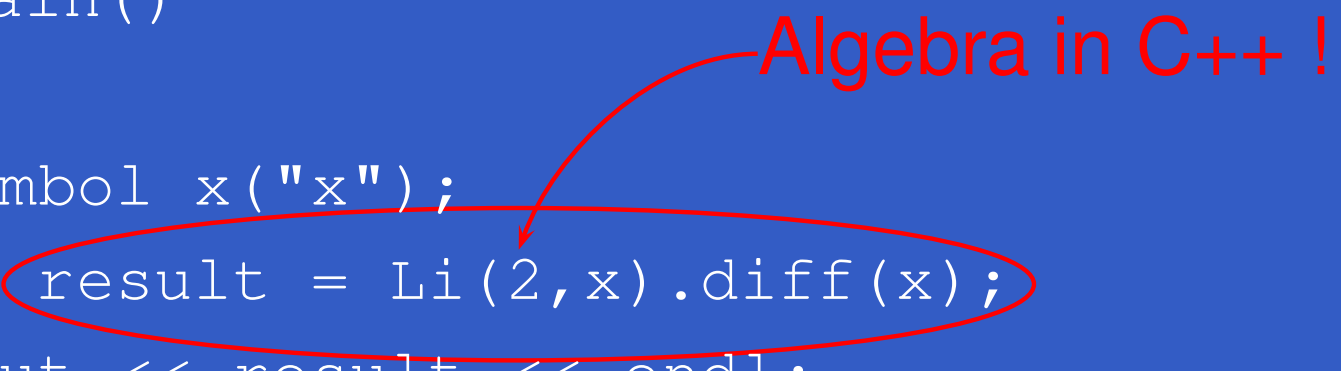
New data types from GiNaC

Programming GiNaC

- Write a C++ program, e.g. firstexample.cpp:

```
#include <iostream>
using namespace std;
#include <ginac/ginac.h>
using namespace GiNaC;

int main()
{
    symbol x("x");
    ex result = Li(2,x).diff(x);
    cout << result << endl;
    return 0;
}
```



Programming GiNaC

- Compile the program:

```
$ g++ firstexample.cpp -lginac
```

- Run it:

```
$ ./a.out
-log(1-x)*x^(-1)
```

Using *ginsh*

```
ginsh - GiNaC Interactive Shell (GiNaC V1.3.0)
  __,  _______  Copyright (C) 1999-2004 Johannes Gute
 (__) *       | Germany.  This is free software with
  ._) i N a C | You are welcome to redistribute it un
<-------------' For details type `warranty;'.

Type ?? for a list of help topics.
> diff(Li(2,x),x);
-x^(-1)*log(1-x)
> _
```

- Exit *ginsh*: type `exit;` or press CTRL-D
- End lines with a semicolon
- Pressing TAB shows all available commands

Fundamentals of GiNaC

Container for arbitrary algebraic expressions

$\longrightarrow$ *ex*

Fundamentals of GiNaC

Container for arbitrary algebraic expressions

$\longrightarrow$ *ex*

After declaration

```
ex myexpr;
```

or as a function parameter

```
ex nloopfct(ex momentum)
{...}
```

Fundamentals of GiNaC

Container for arbitrary algebraic expressions

$\longrightarrow$ *ex*

After declaration

```
ex myexpr;
```

or as a function parameter

```
ex nloopfct(ex momentum)
{...}
```

use it:

```
momentum = sqrt(p) * pow(mu, 2) - d;
myexpr = sin(x).series(x, 10);
cout << myexpr << endl;
if (myexpr.has(x)) y = myexpr;
```

Fundamentals of GiNaC

Container for arbitrary algebraic expressions

$\longrightarrow$ *ex*

After declaration

```
ex myexpr;
```

or as a function parameter

```
ex nloopfct(ex momentum)
{...}
```

use it:

```
momentum = sqrt(p) * pow(mu, 2) - d;
myexpr = sin(x).series(x, 10);
cout << myexpr << endl;
if (myexpr.has(x)) y = myexpr;
```

Fundamentals of GiNaC

Symbols $\longrightarrow$ `symbol`

Declaration:

```
symbol x("x");
symbol eps("epsilon")
```

Fundamentals of GiNaC

Symbols $\longrightarrow$ `symbol`

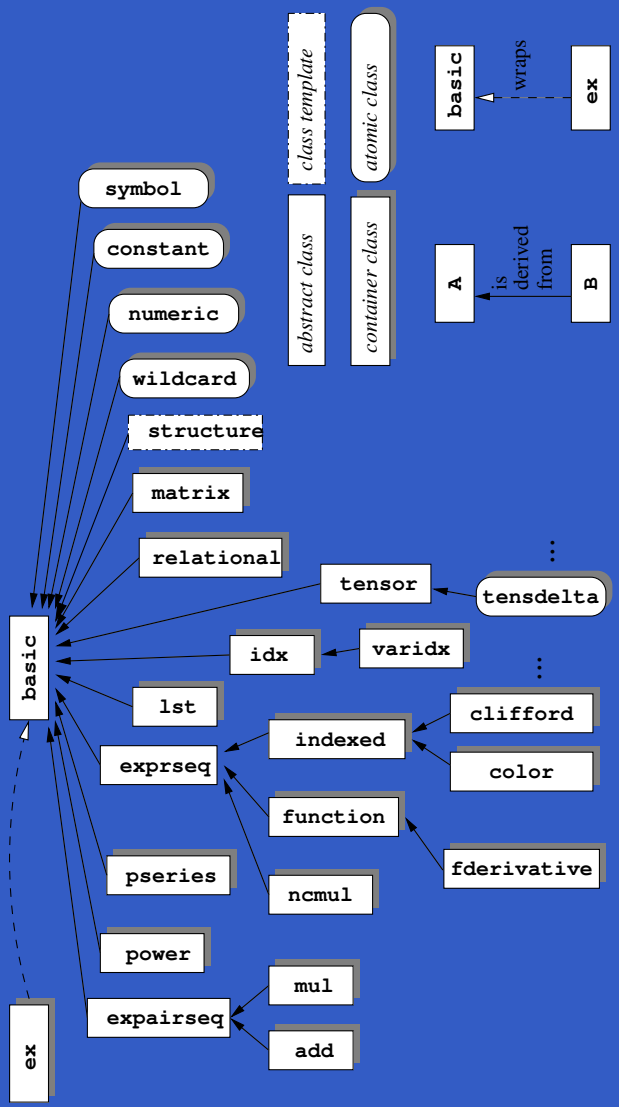
Declaration:

```
symbol x("x");
symbol eps("epsilon")
```

What else?

- Numbers 1.34, 3/4, 2i, ...
- Operations +, -, diff, ...
- Mathematical functions and other objects

Class hierarchy



hermite.cpp

```
#include <iostream>
using namespace std;
#include <ginac/ginac.h>
using namespace GiNaC;

ex HermitePoly(const symbol& x, int n)
{
    ex HKer = exp(-pow(x, 2));
    return normal(pow(-1, n) * diff(HKer, x, n) / HKer);
}

int main(int argc, char* argv[])
{
    if (argc != 2) return 1;
    int i = atoi(argv[1]);

    symbol z("z");
    cout << "H_" << i << "(z) == " << HermitePoly(z, i) << endl;

    return 0;
}
```

Mathematical functions

- sin, cos, atan, asinh, ...
- sqrt, pow, exp, log
- tgamma, psi, beta, factorial, binomial
- Li, G, H, zeta
- abs, csgn, conjugate, gcd, lcm

Expression manipulation

- expand

```
> expand((x+y)^5);
```

Expression manipulation

- expand

```
> expand((x+y)^5);
```

- collect

```
> f=expand((x+y)*(a-b));
x*a-y*b-b*x+y*a
> collect(f,x);
x*(-b+a)-y*b+y*a
```

Expression manipulation

- expand

```
> expand((x+y)^5);
```

- collect

```
> f=expand((x+y)*(a-b));
x*a-y*b-b*x+y*a
> collect(f,x);
x*(-b+a)-y*b+y*a
```

- normal

```
> t1=(x^2 + 2*x + 1) / (x + 1);
> t2=(sin(x)^2 + 2*sin(x) + 1) / (sin(x) + 1);
> normal(t1*t2);
(1+sin(x))^(-1)*(1+x)
```

Expression manipulation

- subs

```
> f=Li(5,x);
> subs(f, x==-1);
-15/16*zeta(5)
```

Expression manipulation

- subs

```
> f=Li(5,x);
> subs(f, x==-1);
-15/16*zeta(5)
```

- evalf

```
> evalf(%);
-0.97211977044690930 59
```

Series expansion

- diff

```
> diff(H({3,2,1},x), x, 3);
(1-x)^(-1)*x^(-2)*S(1,2,x)+2*H(2,2,1,x)*x^(-3)
-3*H(1,2,1,x)*x^(-3)
```

Series expansion

- diff

```
> diff(H({3,2,1},x), x, 3);
(1-x)^(-1)*x^(-2)*S(1,2,x)+2*H(2,2,1,x)*x^(-3)
-3*H(1,2,1,x)*x^(-3)
```

- series

```
> series(tgamma(1-eps), eps, 4);
1+(Euler)*eps+(1/12*Pi^2+1/2*Euler^2)*eps^2
+Order(eps^3)
```

Objects with indices

- Symbolic matrices
- Tensors
- Color algebra, Dirac algebra

```
symbol A("A");
idx i(symbol("i"), 3), j(symbol(j), 3);
ex e = indexed(A, i, j) * delta_tensor(i, j);
e = e.simplify_indexed();

varidx mu(symbol("mu"), D);
e = indexed(A, mu.toggle_variance());
```

SU(3) color algebra

- `color_T(a)` $\leftrightarrow t_a$
- `color_f(a,b,c)` $\leftrightarrow [t_a, t_b] = i f_{abc} t_c$
- `color_d(a,b,c)` $\leftrightarrow \{t_a, t_b\} = \frac{1}{3}\delta_{ab} + d_{abc} t_c$

$\rightarrow$ color.cpp

- $f_{lab} f_{abk}$
- $t_k t_a t_b t_k$
- trace$(4 t_a t_b t_k)$

Dirac algebra

$$\{\gamma_\mu, \gamma_\nu\} = 2g_{\mu\nu}$$

- `dirac_gamma(mu)`
- `dirac_ONE()`
- `dirac_gamma5()`
- `dirac_L()`
- `dirac_R()`
- `dirac_slash(p, dim)`
- `lorentz_g(mu, nu)`
- `dirac_trace(...)`

Calculating $|M|^2$

$\longrightarrow$ m.cpp

www.ginac.de