1 Algebraic Structures in SageMath

- Building the car: other libraries
- Basic rings: $\mathbb{Z}, \mathbb{Q}, \mathbb{R}$
- Polynomials and power series
- Parents and coercion
- Finite fields
- And beyond: linear algebra, symbolics, creating new parents

1.1 Building the car: other libraries

Part of the original philosophy of Sage is to take advantage of the best existing open source algebra systems, tying them together through Python.

- Using OSCAR from Sage and vice versa via PyJulia
- Determining libraries used using `get_systems` (requires building Sage with SAGE_PROFILE=yes):

[2]: R.<x,y> = ZZ[]
from sage.misc.citation import get_systems
get_systems('x+y+1')

[2]: ['Singular']

- A list of all libraries used by Sage is available here.
- Some systems are used as libraries (e.g. libgap, pari), others by a Pexpect interface (e.g. magma, fricas). Pexpect is slower but easier: passes strings.
1.2 Basic rings

: Integer arithmetic uses GMP (as of 9.3.beta2), preparses to get Integer objects rather than Python ints. Various number theoretic methods.

\[ (2^{512}).\text{next\_prime()} \]

\[ (2^{256}+17).\text{factor()} \]

\[ 3 \times 2549 \times 1762903 \times 858931867307207720967237044861984560661754999503936575763779105033 \]

: Rational numbers use GMP's mpq_t type.

\[ \text{bernoulli}(70).\text{support()} \]

\[ [2, 3, 5, 7, 11, 71, 688531, 20210499584198062453, 3090850068676441179447] \]

: Several models of real numbers. Python floats, RDF using C doubles, MPFR for multi-precision floating point, MPFI for interval arithmetic, Arb for arbitrary precision balls.

\[ \text{RR}.\pi() \]

\[ 3.14159265358979 \]

\[ \text{RDF}.\pi() \]

\[ 3.141592653589793 \]

\[ \text{import math} \]
\[ \text{type(math.pi)} \]

\[ \text{<class 'float'>} \]

\[ \text{RealIntervalField(200)}.\pi() \]

\[ 3.1415926538979323846264338327950288419716939937510582097495? \]

\[ \text{RealBallField(200)}.\pi() \]

\[ [3.1415926538979323846264338327950288419716939937510582097494 +/- 5.73e-60] \]

: Multiple models of complex numbers as well. Python complex type, CDF using C doubles, MPC for multi-precision, MPFI for rectangles, Arb for balls.

\[ \text{complex}(1,1)^3 \]
1.3 Polynomials and power series

- Generic code supports arbitrary base rings (both sparse and dense).
- Univariate and multivariate polynomials behave quite differently (term orders for multivariate)
- Libraries used for specific base fields give big speedups and additional functionality
- FLINT, NTL, Arb, MPFR for univariate; Singular, Macaulay2 for multivariate; Polybori for boolean polynomials

```
[20]: R.<x> = ZZ[] # preparser syntax
(\texttt{x}^{18} - 1).\texttt{factor}()

[20]: (x - 1) \ast (x + 1) \ast (x^{2} - x + 1) \ast (x^{2} + x + 1) \ast (x^{6} - x^{3} + 1) \ast (x^{6} + x^{3} + 1)

[21]: (x^{6} - x^{3} + 1).is_cyclotomic()

[21]: True

[22]: 1/(1 - x + x^{2} + \texttt{O}(x^{10}))

[22]: 1 + x - x^{3} - x^{4} + x^{6} + x^{7} - x^{9} + \texttt{O}(x^{10})

[39]: S.<x,y> = PolynomialRing(QQ, order='\texttt{degrevlex}')
I = S.ideal([-3*x^{2} - x*y - y^{2} + y, -x^{2} + x*y - y^{2} - 9*x])
\texttt{sorted}(I.\texttt{groebner\_basis}())
```
[39]: \[ \begin{align*}
    &x*y - 1/2*y^2 - 27/4*x - 1/4*y, \\
    &x^2 + 1/2*y^2 + 9/4*x - 1/4*y, \\
    &y^3 - 3/4*y^2 + 261/8*x + 101/24*y
\end{align*} \]

[41]:
\[
T.<x,y> = PolynomialRing(QQ, order='lex')
\]
\[
I = T.ideal([-3*x^2 - x*y - y^2 + y, -x^2 + x*y - y^2 - 9*x])
\]
\[
sorted(I.groebner_basis())
\]

[41]: \[y^4 - 15/2*y^3 + 307/12*y^2 - 81/4*y, x + 8/261*y^3 - 2/87*y^2 + 101/783*y\]

1.4 Parents and coercion

- Many objects in Sage fall into one of three levels: an element (e.g. a polynomial), a parent (e.g. the ring ) or a category (e.g. commutative rings with unity).
- A parent is simultaneously a set of elements and an object in a specific category.
- The integer 2 and the rational number 2/1 have different parents but we want to consider them equal.
- Solution: create a commuting family of homomorphisms between parents in Sage, called coercions.

[42]:
\[
f = QQ.coerce_map_from(ZZ)
\]
f

[42]: Natural morphism:
\[
\text{From: Integer Ring} \\
\text{To: Rational Field}
\]

[44]:
\[
f(2), f(2).parent()
\]

[44]: (2, Rational Field)

[45]: 2 + (1/2)

[45]: 5/2

- Most coercions are injective, but some are quotient maps.
- Note that coercions only depend on the parent and should be homomorphisms (in an appropriate category).

[52]:
\[
\text{Zmod(12).coerce_map_from(ZZ)}
\]

[52]: Natural morphism:
\[
\text{From: Integer Ring} \\
\text{To: Ring of integers modulo 12}
\]

- Sometimes there is no coercion map in either direction but there is a natural pushout: another parent that both coerce to.
R = PolynomialRing(QQ, 'x')
S = PolynomialRing(ZZ, 'x,y')
print(R.has_coerce_map_from(S))
print(S.has_coerce_map_from(R))

False
False

x = R.gen(0)
y = S.gen(1)
(x+y).parent()

Multivariate Polynomial Ring in x, y over Rational Field

from sage.categories.pushout import pushout
pushout(R, S)

Multivariate Polynomial Ring in x, y over Rational Field

• Pushouts are determined using the construction() method on each parent.

R.construction()

(Poly[x], Rational Field)

S.construction()

(MPoly[x,y], Integer Ring)

1.5 Finite fields

• Sage uses six different implementations of finite fields, depending on $q$.

# prime fields, $q < \sqrt{2^{31}-1}$
type(GF(5)(1))

<class 'sage.rings.finite_rings.integer_mod.IntegerMod_int'>

# prime fields, $\sqrt{2^{31}-1} < q < 2^{32}$
type(GF(next_prime(2**16))(1))

<class 'sage.rings.finite_rings.integer_mod.IntegerMod_int64'>

# prime fields, $q > 2^{32}$
type(GF(next_prime(2**32))(1))

<class 'sage.rings.finite_rings.integer_mod.IntegerMod_gmp'>
When \( q \) is not prime, there are different methods for choosing the modulus since there are conflicting requirements.

- We want the computation of the modulus to be fast. Good approach: Adleman-Lenstra algorithm.
- We want maps between fields of the same cardinality to be fast. Good approach: if \( b \) generator for \( p^m \) and \( m|n \) ask that \( b^{((p^n-1)/(p^m-1))} \) satisfies modulus for \( p^m \). To make unique, choose lexicographically least polynomials (Conway polynomial)

Sage decides which to prioritize based on whether you specify a **variable name**

```python
# If \( q \) is small enough to be contained in the database of Conway polynomials then
# you'll get a Conway polynomial either way (unless you specify an algorithm)
GF(3^12).modulus()
```

```python
x^12 + x^6 + x^5 + x^4 + x^2 + 2
```

```python
(GF(3^24).gen()^(((3^24-1)/(3^12-1))).minimal_polynomial()
```

```python
x^12 + x^6 + x^5 + x^4 + x^2 + 2
```

```python
# Above the cutoff, if you give a variable name it prioritizes finding the_
# modulus quickly
GF(997^12, 'a').modulus()
```

```python
x^12 + 7*x^11 + 15*x^10 + x^9 + 972*x^8 + 24*x^7 + 152*x^6 + 208*x^5 + 184*x^4 +
144*x^3 + 78*x^2 + 51*x + 71
```

```python
# If you don't give a variable name, it chooses compatibly with divisors
k = GF(997^12)
k.modulus()
```
If you will be working outside the regime where Conway polynomials are known, choosing polynomials compatibly can get quite slow. If you can choose a large finite field in advance, it should be faster to fix a modulus there and then construct subfields.

```python
k.<a> = GF(3^240)
S = k.subfields()
```

ValueError Traceback (most recent call last)

```python
<ipython-input-92-86b9f5197bd8> in <module>
----> 1 S = k.subfields()
```

```
1559     pairs = []
1560     for m in divisors:
-> 1561         K = self.subfield(m, name=name[m])
1562         pairs.append((K, self.coerce_map_from(K)))
1563     return pairs
```

```
1472         else:
1473             gen = self.gen()**((self.order() - 1)//(p**degree - 1))
```
K = GF(p**degree, modulus=gen.minimal_polynomial(),
name=name)
try: # to register a coercion map, embedding of K to self
    self.register_coercion(K.hom([gen], codomain=self,
check=False))

~/sage/sage-9.2.beta12/local/lib/python3.8/site-packages/sage/structure/
factory.pyx in sage.structure.factory.UniqueFactory.__call__ (build/cythonized/
sage/structure/factory.c:2181)()
365 False
366 ""
--> 367 key, kwds = self.create_key_and_extra_args(*args, **kwds)
368 version = self.get_version(sage_version)
369 return self.get_object(version, key, kwds)

~/sage/sage-9.2.beta12/local/lib/python3.8/site-packages/sage/rings/
finite_rings/finite_field_constructor.py in create_key_and_extra_args(self,
order, name, modulus, names, impl, proof, check_irreducible, prefix, repr,
elem_cache, base, **kwds)
659 if modulus is not None:
660     if modulus.degree() != relative_degree:
--> 661     raise ValueError("degree of the modulus does not
match the relative degree of finite field")
662 if check_irreducible and not modulus.is_irreducible():
663     raise ValueError("finite field modulus must be irreducible but it is not")

ValueError: degree of the modulus does not match the relative degree of finite field

For number fields, one can construct relative extensions. Applications include finding minimal polynomials, norms and traces in this relative extension. Work is in progress to add relative extensions of finite fields.

Obviously there are many more algebraic structures in Sage.

1.5.1 Linear algebra

You can work with matrices or modules

[93]: M = matrix(3,3,range(9))

[94]: M.kernel()
Free module of degree 3 and rank 1 over Integer Ring
Echelon basis matrix:
\[
\begin{bmatrix}
1 & -2 & 1 \\
0 & 1 & 2 \\
0 & 0 & 0
\end{bmatrix}
\]

\[M = \text{M.rref()}
\]

\[V = \mathbb{Q}^3
\]
\[W = V.\text{subspace}([[1,2,3],[4,5,6]])
\]

Vector space of degree 3 and dimension 2 over Rational Field
Basis matrix:
\[
\begin{bmatrix}
1 & 0 & -1 \\
0 & 1 & 2
\end{bmatrix}
\]

\[X = V.\text{subspace}([[1,2,1],[7,8,9]])
\]
\[X.\text{intersection}(W)
\]

Vector space of degree 3 and dimension 1 over Rational Field
Basis matrix:
\[
\begin{bmatrix}
1 & 8/7 & 9/7
\end{bmatrix}
\]

1.6 Symbolics

Symbolics (symbolic functions, integration and differentiation, solving equations) is mostly separated from the rings and fields in Sage (though the “symbolic ring” is a parent).

One warning: x is predefined in Sage as a symbolic variable, and you can also get symbolics by taking roots of integers for example. Unless you’re doing calculus or solving symbolic equations, you should usually try to avoid the symbolic ring, since it’s much slower than polynomial rings for arithmetic.

\[\text{x = var('x')}
\]
\[f = e^x*\cos(x)
\]
\[f.\text{derivative(x)}
\]

\[\cos(x)*e^x - e^x*\sin(x)
\]

\[f.\text{integral(x)}
\]

\[1/2*(\cos(x) + \sin(x))*e^x
\]

\[f.\text{integrate(x, 0, pi)}
\]
1.7 Creating parents

When you need an algebraic structure not yet in Sage, you’ll generally need to implement a parent class and an element class. See Simon King’s guide.