Rings and Algebras in MAGMA

Silvia Properzi

January 13, 2026

Outline / Table of Contents

Integers, Residue Rings, and Fields

Rings of integers

- `Integers()`

Residue class rings

- `Integers(n)` creates $\mathbb{Z}/n\mathbb{Z}$
- This is a ring, not a field in general

Fields

- `Rationals()` for $\mathbb{Q}$
- `GF(p)` for $\mathbb{F}_p$

```
Z := Integers();
Z12 := Integers(12);
Q := Rationals();
F5 := GF(5);
```

Polynomial Rings

Univariate polynomial rings

- `R<x> := PolynomialRing(R)`

Multivariate polynomial rings

- `R<x,y> := PolynomialRing(R,2)`

Note

- Polynomial rings are exact rings.
- They are the basis for most constructions.

```
Z := Integers();
R<x,y> := PolynomialRing(Z,2);
```

Fraction Fields and Rational Function Fields

For `R` an integral domain.

Field of fractions

- `F<t> := FieldOfFractions(R)`

Note `FieldOfFractions` does not explicitly construct the inclusion $R \hookrightarrow \mathrm{Frac}(R)$.

```
Z := Integers();
ZX<x> := PolynomialRing(Z);

FieldOfFractions(ZX);
// Univariate rational function field
// over Integer Ring
// Variables: $.1

F<t> := FieldOfFractions(ZX);
F!x; // t
```

Subrings

```
R := Integers();
S,f := sub< R | 2 >;
```

- Subrings are defined by generators
- The output has the inclusion map

Ideals

```
R<x,y> := PolynomialRing(Rationals(), 2);
I := ideal< R | x^2, y^2, x*y >;
```

Note:

```
R<x> := PolynomialRing(Integers(), 1);

//  ERROR: coefficient ring must be a field
I := ideal< R | x^2 - 2 >;

//  Works (symbolic quotient)
Q := quo< R | x^2 - 2 >;
```

- `ideal<R|...>` requires the coefficient ring to be a field
- Over $\mathbb{Z}$, general ideal machinery is unavailable

Homomorphisms via Generators

Ring homomorphisms are defined as maps or by images of generators

```
P<x> := PolynomialRing(Integers());
R := quo< P | x^2 >;
S := Integers(4);
phi := hom< R -> S | 2 >; // x |-> 2
phi(x);
Kernel(phi);
```

Note: Relations are *not checked*

```
// x |-> 3 does NOT respect x^2 = 0
psi := hom< R -> S | 3 >;
psi(x^2) eq psi(x)^2; // false
```

- `map<R->S|x :-> f(x)>` defines a function
- No algebraic properties are checked

Ring Predicates in MAGMA

- `IsCommutative(R)`
- `IsUnitary(R)`
- `IsFinite(R)`
- `IsOrdered(R)`
- `IsIntegralDomain(R)`
- `R eq S`   `R ne S`
- `IsField(R)`
- `IsLocal(R)`
- `IsDivisionRing(R)`
- `IsEuclideanRing(R)`
- `IsMagmaEuclideanRing(R)`
- `IsPID(R)`
- `IsUFD(R)`
- `HasGCD(R)`
- `IsArtinian(R)`
- `IsNoetherian(R)`

Euclidean Ring Distinction in MAGMA: `IsEuclideanRing(R)` tests the mathematical property, while `IsMagmaEuclideanRing(R)` checks if MAGMA can actually run Euclidean algorithms.

Euclidean Rings in MAGMA

Note: A ring may be Euclidean in theory, but not "computably Euclidean" in MAGMA.

```
R<x> := PolynomialRing(Integers());
Q := quo< R | x^2 - 2 >;
IsEuclideanRing(Q); //true
IsMagmaEuclideanRing(Q);// false
```

Explanation:

- $Q = \mathbb{Z}[x]/(x^2 - 2)$ is mathematically Euclidean. Hence `IsEuclideanRing(Q)` returns `true`.
- MAGMA does not implement the necessary Euclidean operations for this quotient. Therefore `IsMagmaEuclideanRing(Q)` returns `false`.

Algebraic Extensions: the Polynomial $x^2 + 5$

Let $f(x) = x^2 + 5$.

1. Quotient $\mathbb{Z}[x]/(x^2 + 5)$

```
Z := Integers();
R<x> := PolynomialRing(Z);
Q := quo<R | x^2 + 5>;
Type(Q);    // RngUPolRes
IsDomain(Q);
IsUFD(Q); //fails
IsPrime(Q!x); // fails
```

Algebraic Extensions: the Polynomial $x^2 + 5$

2. Algebraic extensions $\mathbb{Q}(\sqrt{-5})$ and $\mathbb{Z}[\sqrt{-5}]$

```
K2<a> := ext<Rationals() | x^2 + 5>;
Type(K2);  // FldNum
Za:= ext<Integers() | x^2 + 2>;
Type(Za); // RngOrd
 K2 eq NumberField(x^2 + 5); // false
```

Note: `Za<a> := ext<Integers() |` $x^2 + 5$`>;` ERROR

Possible solution:

```
Zx<x> := PolynomialRing(Integers());
Za := ext<Integers() | x^2 + 5>;
a := Za.1; // a = 1
b := Za.2  // b = sqrt(-5)
alpha := 3 + 2*b;
IsPrime(alpha);
IsIrreducible(alpha);
```

Prime and Irreducible elements: number fields and ring of integers

3. Number Field $\mathbb{Q}(\sqrt{-5})$ and its ring of integers

```
Qx<x> := PolynomialRing(Rationals());
K<a> := NumberField(x^2 + 5);
Type(K);    // FldNum
OK:=Integers(K);
Type(OK); // RngOrd
```

If we now consider the integers,

```
OK := Integers(K);
p := OK!a;
IsIrreducible(p); // fails
IsPrime(p); //fails
I := ideal<OK | a>;
IsPrime(I); // works
IsMaximal(I); // fails
```

Local and Series Rings

***p*-adic fields**

- `pAdicField(p)`

Power and Laurent series

- `PowerSeriesRing(R)`
- `LaurentSeriesRing(R)`

Note

- These are approximate rings with finite precision.

Free and Finitely Presented Algebras

Free associative algebras

- `FreeAlgebra(R,n)`

Finitely presented algebras

- Quotients of free algebras

```
k := GF(3);
F<x,y> := FreeAlgebra(k,2);
A := quo<F | x^2, y^2, x*y>;
```

- `MatrixAlgebra(R,n)`

Group algebras

- `GroupAlgebra(R,G)`

Jacobson radical

`JacobsonRadical` works for finite-dimensional algebras over fields.

```
M := MatrixAlgebra(Rationals(),2);
X := M![1,0,0,0];
Y := M![0,1,0,0];
A := sub<M | X,Y>;
Dimension(A);    // 2
JacobsonRadical(A);
// Matrix Algebra [ideal of A] of degree 2
// and dimension 1 over Rational Field
```

It fails for finitely presented algebras (even if finite-dimensional).

```
F<x,y> := FreeAlgebra(Rationals(),2);
B := quo<F |x^2,y^2,x*y-y*x>;
Dimension(B); // 4
JacobsonRadical(B); // fails:
// Runtime error in 'JacobsonRadical':
// Bad argument types Argument types given:
//AlgFP
```

Additive Group of a Ring

AdditiveGroup returns the additive group as an abelian group, along with a map to the ring.

```
R := Integers(12);
A, phi := AdditiveGroup(R);
phi; // map from A to R
AdditiveGroup(Integers());
//Abelian Group isomorphic to Z
AdditiveGroup(GF(16));
// Abelian Group isomorphic to
// Z/2 + Z/2 + Z/2 + Z/2
```

Note: It doesn't work for infinite fields or polynomial rings.

```
AdditiveGroup(Rationals()); //fails

P:=PolynomialRing(Integers())
AdditiveGroup(P); // fails
```

Units of a Ring

```
R  := Integers(12);
U, f:= UnitGroup(R);
f; // map from U to R
U<u,v> := UnitGroup(R);
Generators(U) eq {u,v}; // true
```

Note: It doesn't work for infinite fields or polynomial rings.

```
UnitGroup(Rationals()); // fails
Zx:=PolynomialRing(Integers())
UnitGroup(Zx); // fails
Qx:=PolynomialRing(Rationals())
UnitGroup(Qx); // fails
```

Units in Matrix Rings

```
M  := MatrixRing(Integers(9), 2);
A  := M![1,2,3,4];
IsUnit(A);      //true
Inverse(A);     // fails
A^-1;           // works
UnitGroup(M); //fails
```

Note: `UnitGroup` is only available for matrices over finite fields.

```
UnitGroup(MatrixRing(Rationals(), 2););
// Runtime error:
// Base field for algebra must be finite
```

```
UnitGroup(MatrixRing(Integers(7), 2););//fails
M := MatrixRing(GF(7), 2);
b, G := UnitGroup(M);//b=true, G=GL(2, GF(7))
G; // prints the group and the two generators
b, G<A,B>:= UnitGroup(M); // A, B are the gens
```

Changing the Base Ring

- Magma supports coercion between polynomial rings

```
P<x> := PolynomialRing(Integers(), 1);
Q<y> := PolynomialRing(Rationals(), 1);
f := P!(x^2 + 2);
g := Q!f; // Change base ring to Q
```

- ChangeRing allows base extension for algebras

```
k := FiniteField(3);
F<x,y> := FreeAlgebra(k, 2);
I := ideal< F | x^2, y^2, x*y >;
A := quo< F | I >;
L := FiniteField(9);
AL := ChangeRing(A, L);
```