

CQF VERSION 0.2 MANUAL

PRZEMYSŁAW KOPROWSKI

CQF is a Magma [1] package for doing computations in algebraic theory of quadratic forms. The package can be downloaded from the author's webpage at <http://www.pkoprowski.eu/cqf>. The package is released under standard MIT license. The full statement of the license is given at the end of this manual.

The package was developed on version 2.23 of Magma. The author has not tested it on other versions (yet), but will appropriate any reports on how it works in different setups. Questions and comments should be sent via email to przemyslaw.koprowski@us.edu.pl.

1. QUICK START GUIDE

We present here a quick glimpse of the use of the package. Full description of the functions used in this tutorial can be found in the following sections. We assume that the reader is at least mildly familiar with Magma. Otherwise we suggest to consult [3] prior to reading this manual.

First we need to load the package (see Section 2):

```
> AttachSpec("path_to_cqf/cqf.spec");
```

Next, we fix a field we will be working with. Let's take $K = \mathbb{Q}(\sqrt{37})$.

```
> K<sqrt37> := QuadraticField(37);
```

Define a quadratic form $q = \langle 3 + 2\sqrt{37}, -7 + \sqrt{37}, 31 - 5\sqrt{37}, 514 - 54\sqrt{37} \rangle$ over K (see Section 4.1):

```
> q := DiagonalQuadraticForm(K,  
>   [3+2*sqrt37, -7+sqrt37, 31-5*sqrt37, 514-54*sqrt37]  
> ); q;  
(2*sqrt37 + 3, sqrt37 - 7, -5*sqrt37 + 31, -54*sqrt37 + 514)
```

and check whether the form is isotropic (see Section 4.3):

```
> IsIsotropic(q);  
true
```

Indeed it is. So, let's see if it happens to be hyperbolic:

```
> IsHyperbolic(q);  
false
```

The isotropic form of dimension 4 that is not hyperbolic must have Witt index equal 1:

```
> WittIndex(q);  
1
```

To round this tutorial off we will determine the Witt class of K . It is known (see e.g. [5, Section 20.2]) that there are precisely seven Witt classes of quadratic fields represented by $\mathbb{Q}(\sqrt{d})$ for $d \in \{-1, \pm 2, \pm 7, \pm 17\}$. The following code finds the one which is Witt equivalent to K :

```
> reps := { QuadraticField(j) : j in [-1,2,-2,7,-7,17,-17] };
> for L in reps do
>   if AreWittEquivalent(K, L) then
>     print "K is Witt equivalent to", L;
>     break;
>   end if;
> end for;
K is Witt equivalent to Quadratic Field with defining polynomial
$.1^2 - 2 over the Rational Field
```

It turns out that K is in the same Witt class as $\mathbb{Q}(\sqrt{2})$. See Section 5 for a description of `AreWittEquivalent`.

2. PACKAGE INSTALLATION AND LOADING

The CQF package can be downloaded for free from the author's web page
<http://www.pkoprowski.eu/cqf>

The package comes in form of a zip archive that must be extracted and put on disc in a place where Magma is able to find it. To test the installation go to the directory where CQF resides and execute

```
magma tests.magma
```

If it prints "All tests passed" this means that the installation is correct.

The CQF package is loaded into Magma by attaching its spec file:

```
> AttachSpec("path_to_cqf/cqf.spec");
```

One can also add this command to Magma startup file (specified by the environment variable `MAGMA_USER_SPEC`) to auto-load the package when Magma starts. For details about Magma startup environment consult [2, Sections 2.3.9 and 4.3].

3. SUMS OF SQUARES

A sum of squares of elements of a given field is a frequent object of research in quadratic form theory. CQF provides the following functions for working with sums of squares.

```
IsSumOfSquares(a)
```

```
IsSOS(a)
```

Checks if a is a sum of squares in its parent field.

Supports: finite fields, rationals, number fields, global rational function fields, global function fields, reals, complex numbers

```
LengthOfSumOfSquares(a)
```

```
LengthOfSOS(a)
```

Given a nonzero element a of a field K , this function determines the minimal number of summands needed to express a as a sum of squares. It returns 0 when a is not a sum of squares in K .

Supports: finite fields, rationals, number fields, global rational function fields, global function fields, reals, complex numbers

Example. Check that 7 is a sum of four squares in \mathbb{Q} :

```
> IsSOS(7);
true
> LengthOfSOS(7);
4
```

Example. In the next example we take a number field $K = \mathbb{Q}(\theta)$, where θ is a root of the polynomial $x^4 - 5x^2 + 25$. We then show that $\theta + 1$ is a sum of three squares in K .

```
> _<x> := PolynomialRing(Rationals());
> K<theta> := NumberField(x^4 - 5*x^2 + 25);
> IsSumOfSquares(theta + 1);
true
> LengthOfSumOfSquares(theta + 1);
3
```

Level(K)

The *level* of a field is by definition (see e.g. [4, Definition XI.2.1]) the length of -1 . Hence `Level(K)` is equivalent to `LengthOfSOS(K!-1)`. In particular, if K is formally real, then `Level(K)` returns zero.

Supports: finite fields, rationals, number fields, global rational function fields, global function fields, reals, complex numbers

IsFormallyReal(K)

IsReal(K)

A field K is called *formally real* (or shortly *real*), when -1 is not a sum of squares in K . Hence this function is equivalent to `IsSOS(K!-1)`.

Supports: finite fields, rationals, number fields, global rational function fields, global function fields, reals, complex numbers

PythagorasNumber(K)

The *Pythagoras Number* $P(K)$ of a field K is by definition (see [4, Definition XI.5.5]) the supremum of the lengths of all nonzero sums of squares in K . Hence, Pythagoras number equal $n \in \mathbb{N}$ means that every sum of squares in K is in fact a sum of n squares.

Supports: finite fields, rationals, number fields, global rational function fields, global function fields, reals, complex numbers

PythagorasElement(K)

This function returns a sum of squares of maximal length in K . Hence it returns an element $a \in K$ such that the length of a equals the Pythagoras number of K .

Supports: finite fields, rationals, number fields, global rational function fields, global function fields, reals, complex numbers

Example. Take a number field $K = \mathbb{Q}(\theta)$, where θ is a root of the polynomial $x^4 - 5x^2 + 25$. We show that the $P(K) = 3$ and that $\theta + 1$ is an element of the maximal length.

```

> _<x> := PolynomialRing(Rationals());
> K<theta> := NumberField(x^4 - 5*x^2 + 25);
> PythagorasNumber(K);
3
> a := PythagorasElement(K); a;
theta + 1
> LengthOfSOS(a);
3

```

4. QUADRATIC FORMS

Magma has a built-in type `ModTupFld` for representing quadratic spaces. CQF introduced an additional category `DiagQuadFrm` that represents *diagonal* quadratic forms. Objects of this type can be converted to quadratic spaces or multivariate polynomials. Conversely, any quadratic space over a field of odd characteristic can be diagonalized and converted into a diagonal quadratic form.

4.1. Construction.

`DiagonalQuadraticForm(K, [a1, ..., an])`

`DiagonalQuadraticForm([a1, ..., an])`

Constructs a quadratic form $\langle a_1, \dots, a_n \rangle = a_1x_1^2 + \dots + a_nx_n^2$ over a field K . If the field is not specified, it is assumed to be the universe of the list of coefficients.

Example. Construct a form $\langle 1, 2 - 5 \rangle$ over the rationals:

```

> DiagonalQuadraticForm( Rationals(), [1,2,-5] );
(1, 2, -5)

```

`HyperbolicForm(K, n)`

Construct a hyperbolic form over K of dimension n , i.e. an orthogonal sum of $n/2$ copies of $\langle 1, -1 \rangle$. The dimension must be even, otherwise an error is thrown.

`PfisterForm(K, [a1, ..., an])`

`PfisterForm([a1, ..., an])`

Construct an n -fold Pfister form $\langle\langle a_1, \dots, a_n \rangle\rangle = \langle 1, a_1 \rangle \otimes \dots \otimes \langle 1, a_n \rangle$. If the base field K is not specified, it is assumed to be the universe of the list of coefficients. For more information on Pfister forms see [4, Chapter X] or [5, Chapter 17].

Example. Construct a two-fold Pfister form $\langle\langle 2, 3 \rangle\rangle = \langle 1, 2, 3, 6 \rangle$ over $\mathbb{Q}(i)$:

```

> PfisterForm( QuadraticField(-1), [2,3] );
(1, 2, 3, 6)

```

4.2. Conversion.

`QuadraticFormPolynomial(q)`

Converts a diagonal quadratic form $\langle a_1, \dots, a_n \rangle$ into a multivariate quadratic polynomial $a_1x_1^2 + \dots + a_nx_n^2$.

Example. Convert a form $q = \langle 1, 2, 2 \rangle$ over \mathbb{F}_{11} into a polynomial $x_1^2 + 2x_2^2 + 2x_3^2$:

```

> q := DiagonalQuadraticForm( GF(11), [1,2,2] ); q;
(1,2,2)
> QuadraticFormPolynomial(q);
$.1^2 + 2*$.2^2 + 2*$.3^2
> P<x1,x2,x3> := PolynomialRing(GF(11), 3);
> P ! QuadraticFormPolynomial(q);
x1^2 + 2*x2^2 + 2*x3^2

```

QuadraticFormMatrix(q)

Convert a diagonal quadratic form into a diagonal matrix representing this form.

Example. We use the same form as in the previous example.

```

> q := DiagonalQuadraticForm( GF(11), [1,2,2] ); q;
(1,2,2)
> QuadraticFormMatrix(q);
[ 1 0 0]
[ 0 2 0]
[ 0 0 2]

```

QuadraticSpace(q)

Converts a diagonal quadratic form into a quadratic space (Magma's object of type ModTupFld).

Diagonalization(V)

Given a quadratic space (V, q) over a field of odd characteristic, this function constructs a diagonal quadratic form isometric with (V, q) .

Example. Construct a quadratic space $V = (\mathbb{F}_{53}^3, q)$, where q is a *non-diagonal* quadratic form $x^2 + xy + 3xz - 2yz + y^2 + z^2$. Diagonalize it to get a *diagonal* quadratic form $q' = \langle 1, 14, 30 \rangle$. Then check that $V' = (\mathbb{F}_{53}^3, q')$ is indeed isometric to V .

```

> P<x,y,z> := PolynomialRing(GF(53), 3);
> q := x^2 + x*y + 3*x*z - 2*y*z + y^2 + z^2;
> V := QuadraticSpace(q);
> q_ := Diagonalization(V); q_;
(1, 14, 30)
> V_ := QuadraticSpace(q_);
> IsIsometric(V, V_);
true Mapping from: ModTupFld: V to ModTupFld: V_ given by a rule

```

4.3. Invariants.

BaseRing(q)

BaseField(q)

Returns the base field of the diagonal quadratic form q .

Dimension(q)

The dimension of the diagonal quadratic form q .

Discriminant(q)

The *discriminant* (a.k.a. *signed determinant*) of the quadratic form. By definition (see e.g. [4, p. 30])

$$\text{disc}\langle a_1, \dots, a_n \rangle := (-1)^{\frac{n \cdot (n-1)}{2}} \cdot a_1 \cdots a_n.$$

`IsNondegenerate(q)`

`IsNondegenerate(V)`

A quadratic form is *degenerate* if there is a nonzero vector orthogonal to the whole space iff the discriminant of the form is zero. This function checks whether a given quadratic form is not degenerate. The argument can be either a diagonal quadratic form or a quadratic space (not necessarily diagonal).

`IsIsotropic(q)`

`IsIsotropic(V)`

A non-degenerate quadratic space (V, q) is called *isotropic* if there is a self-orthogonal vector $v \in V$ i.e. a vector such that $q(v) = 0$. This function checks if the given quadratic form is isotropic. An argument can be either a diagonal quadratic form (`DiagQuadFrm`) or a quadratic space (`ModTupFld`). It generalizes a built-in Magma's function of the same name, that works over the rationals and finite fields only.

Supports: finite fields, rationals, number fields, global rational function fields, global function fields, reals, complex numbers

Example. Take a form $\langle 2, 3\theta, 3\theta, 1, 1 \rangle$ over a number field $K := \mathbb{Q}(\theta)$, where θ is a root of $2x^4 + 3x^3 + x^2 + 3x + 1$. We show that it is isotropic.

```
> _<x> := PolynomialRing(Rationals());
> K<theta> := NumberField(2*x^4 + 3*x^3 + x^2 + 3*x + 1); q;
(2, 3*theta, 3*theta, 1, 1)
> q := DiagonalQuadraticForm(K, [2, 3*theta, 3*theta, 1, 1]);
> IsIsotropic( q );
true
```

`IsHyperbolic(q)`

`IsHyperbolic(V)`

A quadratic space over a field of characteristic $\neq 2$ is called *hyperbolic* if it decomposes into an orthogonal sum of hyperbolic planes (a *hyperbolic plane* is a 2-dimension isotropic space, it is isometric to $\langle 1, -1 \rangle$). Equivalently a quadratic space is hyperbolic if it contains a subspace equal to its orthogonal complement. This function checks if the given quadratic form is hyperbolic. The argument can be either a diagonal quadratic form (`DiagQuadFrm`) or a quadratic space (`ModTupFld`). It generalizes a built-in Magma's function of the same name, that works over the rationals and finite fields only.

Supports: finite fields, rationals, number fields, global rational function fields, global function fields, reals, complex numbers

Example. Take the same form $\langle 2, 3\theta, 3\theta, 1, 1 \rangle$ as in the previous example. We show that it is not hyperbolic.

```

> _<x> := PolynomialRing(Rationals());
> K<theta> := NumberField(2*x^4 + 3*x^3 + x^2 + 3*x + 1);
> q := DiagonalQuadraticForm(K, [2, 3*theta, 3*theta, 1, 1]);
> IsHyperbolic( q );
false

```

WittIndex(q)

WittIndex(V)

AnisotropicDimension(q)

AnisotropicDimension(V)

Witt decomposition theorem states that every non-degenerate quadratic space (V, q) decomposes into an orthogonal sum of a hyperbolic and anisotropic subspaces. The dimensions of both these summands is an invariant of the quadratic space and is independent of the actual decomposition. The function `AnisotropicDimension` returns the dimension of the anisotropic part. On the other hand, the number of hyperbolic planes contained in the given quadratic space (i.e. half of the dimension of the hyperbolic part) is called the *Witt index* (see e.g. [4, Definition I.4.3]) of the quadratic space and denoted $\text{ind } V$. The anisotropic dimension and the Witt index are connected by the formula

$$\text{ind } V = \frac{1}{2} \cdot (\dim V - \dim V_a),$$

where V_a is the anisotropic part of V . Both `WittIndex` and `AnisotropicDimension` can be used with either diagonal quadratic forms (`DiagQuadFrm`) or quadratic spaces (`ModTupFld`).

Supports: finite fields, rationals, number fields, global rational function fields, global function fields, reals, complex numbers

Example. Once again we use consider the same quadratic form $\langle 2, 3\theta, 3\theta, 1, 1 \rangle$ as in the previous two examples. We know it is not hyperbolic but it is isotropic. It contains a single hyperbolic plane, so its Witt index is 1 and consequently the dimension of its anisotropic part is 3.

```

> _<x> := PolynomialRing(Rationals());
> K<theta> := NumberField(2*x^4 + 3*x^3 + x^2 + 3*x + 1);
> q := DiagonalQuadraticForm(K, [2, 3*theta, 3*theta, 1, 1]);
> WittIndex( q );
1
> AnisotropicDimension( q );
3

```

AnisotropicDimension(q, P)

This function computes the dimension of the anisotropic part (see the previous command for explanation) of the localization $q \otimes K_P$ of the diagonal quadratic form q at the prime P . Here, P can be either a place or a prime ideal.

Supports: rationals, number fields, global rational function fields, global function fields

Example. Take a quadratic form q over $K = \mathbb{Q}(\sqrt{7})$ and check that the form is anisotropic (see function `IsIsotropic` above):

```
> K<sqrt7> := QuadraticField(7);
> q := DiagonalQuadraticForm(K, [2*sqrt7, 2*sqrt7, 1-sqrt7, 1-sqrt7]);
> IsIsotropic(q);
false
```

Take the unique (ramified) place P of K that divides 2 and compute the dimension of the anisotropic part of the localization $q \otimes K_P$:

```
> P := Decomposition(K, 2)[1][1];
> AnisotropicDimension(q, P);
0
```

The result is zero, hence the localization must be hyperbolic. Indeed it is (see function `IsHyperbolic` above):

```
> IsHyperbolic(q, P);
true
```

```
AreIsometric(q1, q2)
```

```
AreIsometric(q1, V2)
```

```
AreIsometric(V1, q2)
```

```
AreIsometric(V1, V2)
```

This function checks if two quadratic spaces (V_1, q_1) , (V_2, q_2) are isometric, i.e. if there is a linear isomorphism $\varphi : V_1 \rightarrow V_2$ such that $q_2(\varphi(v)) = q_1(v)$ for every $v \in V_1$. The arguments can be any combination of diagonal quadratic forms (`DiagQuadFrm`) and quadratic spaces (`ModTupFld`).

The function is called `AreIsometric` rather than `IsIsometric` to avoid collision with the built-in function of the latter name, which works solely over finite fields but returns also the isometry φ .

Supports: finite fields, rationals, number fields, global rational function fields, global function fields, reals, complex numbers

```
HasseMinkowskiInvariant(q, P)
```

```
WittInvariant(q,P)
```

Let q be a diagonal quadratic form over a global field K of characteristic $\neq 2$. This function computes the Hasse invariant (see e.g. [4, Definition V.3.17]) of the localization $q \otimes K_P$ in the completion of K at P . Here P can be given either as a place or a prime ideal of K . These functions generalize built-in functions of the same names that work only over \mathbb{Q} .

Be aware that the name `WittInvariant` can be a bit misleading since it is **not** the Witt invariant in the sense of [4, p. 117]. The relation between these two notions is explained in [4, Proposition V.3.20]. We use the above name for consistency with the built-in function.

Supports: rationals, number fields, global rational function fields, global function fields

4.4. Other functions.

```
OrthogonalSum(q1, q2)
```

```
q1 + q2
```

The orthogonal sum $q_1 \perp q_2$ of two diagonal quadratic forms q_1, q_2 over the same field.

`q1 - q2`

The “orthogonal difference” i.e. a sum $q_1 \perp (-q_2)$ of two diagonal quadratic forms q_1, q_2 over the same field.

`TensorProduct(q1, q2)`

The tensor product $q_1 \otimes q_2$ of two diagonal quadratic forms q_1, q_2 over the same field.

Example. Let us compute the tensor product

$$\langle 1, 2, 3 \rangle \otimes \langle 1, -1 \rangle = \langle 1, 2, 3, -1, -2, -3 \rangle.$$

```
> q1 := DiagonalQuadraticForm( Rational(), [1, 2, 3]); q1;
(1, 2, 3)
> q2 := HyperbolicForm( Rational(), 2 ); q2;
(1, -1)
> TensorProduct(q1, q2);
(1, 2, 3, -1, -2, -3)
```

5. WITT RING AND WITT EQUIVALENCE

Recall that two non-degenerate quadratic forms q_1, q_2 are said to be *similar* if there are non-negative integers m, n such that $q_1 \perp nH$ is isometric to $q_2 \perp nH$, where H stands for a hyperbolic plane. The set WK of similarity classes of non-degenerate forms posses a ring structure, were addition and multiplication are induced by orthogonal sum and tensor product, respectively. The ring WK is called the *Witt ring* of K . CQF provides two new data-types `RngWittQF` and `RngWittQFelt` that represent a Witt ring of a field and a similarity class (i.e. element of the Witt ring), respectively.

5.1. Ring construction.

`WittRing(K)`

This function constructs the Witt ring WK of the field K . Although K can be *any* field that Magma is able to work with, for most fields very little can be done with the resulting object. Full functionality is limited to the fields supported by CQF, these are: finite field, rationals, number fields, global function fields, reals and the field of complex numbers.

Warning: There is a built-in function of the same name `WittRing(K, n)` that constructs the ring of *Witt vectors* of length n over K . This is a completely different object! Be careful not to mistake these two commands. `WittRing` with just *one* argument creates the Witt ring of similarity classes of quadratic forms and with *two* arguments constructs the ring of Witt vectors.

Example. Let’s see that WF_7 is isomorphic to \mathbb{Z}_4 .

```
> W := WittRing(GF(7));
> print W : Maximal;
Witt ring of Finite field of size 7, isomorphic to Residue class ring
of integers modulo 4
```

5.2. Working with elements.

`W ! q`

`W ! a`

Let W be the Witt ring of a field K . If q is a diagonal quadratic form over K , this operator constructs the class $[q]$ of q in the Witt ring (i.e. the similarity class of q). In a an element of K , this operator constructs the Witt class of the 1-dimensional form $\langle a \rangle$.

`Zero(W)`

This function constructs the zero element of the Witt ring W , i.e. the class of hyperbolic forms.

`One(W)`

This function constructs the unit element of the Witt ring W , i.e. the class of the form $\langle 1 \rangle$. This function is equivalent to $W!1$.

`IsZero(q)`

This function checks if q is the null element of the Witt ring.

Supports: finite fields, rationals, number fields, global rational function fields, global function fields, reals, complex numbers

`IsOne(q)`

This function checks if q is the unit element of the Witt ring.

Supports: finite fields, rationals, number fields, global rational function fields, global function fields, reals, complex numbers

`q1 + q2`

`q1 - q2`

`q1 * q2`

`-q1`

`n * q1`

Standard ring arithmetic operators.

Warning: The result is not simplified. Hence the same Witt class constructed by two different sequences of operators can have different representations! Use the operators `eq` and `ne` to check equality.

`q1 eq q2`

`q1 ne q2`

Standard comparison operators.

Supports: finite fields, rationals, number fields, global rational function fields, global function fields, reals, complex numbers

Example. Construct the Witt ring $W\mathbb{Q}$ of the field of rational numbers:

```
> QQ := Rational();
> WQ := WittRing(QQ); WQ;
Witt ring of Rational Field
```

Take the Witt class of the form $\langle 1, 2, -7 \rangle$ and add the class of $\langle 1 \rangle$. Check that the result is not the null element of $W\mathbb{Q}$:

```

> q := DiagonalQuadraticForm(QQ, [1,2,-7]);
> _q := WQ ! q; _q;
[(1, 2, -7)]
> _q := _q - One(WQ); _q;
[(1, 2, -7, -1)]
> IsZero(_q);
false

```

Alternatively the last line can be

```

> _q eq Zero(WQ);
false

```

IsTorsion(q)

Returns true if and only if q is a torsion element of the Witt ring. For an example use, see the example following the command `Height` below.

Supports: finite fields, rationals, number fields, global rational function fields, global function fields, reals, complex numbers

Parent(q)

Given an element $q \in W$ of a Witt ring, this function returns the Witt ring W .

BaseField(q)

BaseRing(q)

Given an element q of a Witt ring WK of a field K , this function returns K .

5.3. Invariants.

W

Returns the cardinality of the Witt ring W .

Supports: finite fields, rationals, number fields, global rational function fields, global function fields, reals, complex numbers

Example. The Witt ring of \mathbb{Q} is infinite, while the Witt ring of \mathbb{C} has just two elements:

```

> WQ := WittRing(Rationals());
> #WQ;
Infinity
> WC := WittRing(ComplexField());
> #WC;
2

```

BaseField(W)

BaseRing(W)

Given a Witt ring WK of a field K , this function returns K .

Height(K)

This function computes the *height* of K , which by definition (see [4, Definition XI.5.4]) is the exponent of the torsion part of WK .

Supports: finite fields, rationals, number fields, global rational function fields, global function fields, reals, complex numbers

Example. The Witt ring $W\mathbb{R}$ of the reals is isomorphic to \mathbb{Z} , hence torsion-free. Consequently the height of \mathbb{R} equals 1;

```
> Height( RealField() );
4
```

Example. The height of \mathbb{Q} is 4 since $4 \cdot q = 0$ for every torsion element $[q] \in W\mathbb{Q}$ while for $n < 4$ the form $n \cdot \langle 1, -7 \rangle$ is not hyperbolic. To see this construct the Witt ring $W\mathbb{Q}$ of the rationals:

```
> QQ := Rational();
> WQ := WittRing(QQ); WQ;
Witt ring of Rational Field
```

Now, construct the Witt class of $\langle 1, -7 \rangle$:

```
> _q := WQ ! DiagonalQuadraticForm(QQ, [1, -7]);
> _q;
[(1, -7)]
```

See that it is a torsion element:

```
> IsTorsion(_q);
true
```

Compute the height h of \mathbb{Q} and see that $h \cdot [(1, -7)]$ is the null element of the Witt ring:

```
> h := Height(QQ); h;
4
> IsZero(h*_q);
true
```

Finally, let's see that $k \cdot [(1, -7)]$ is nonzero for $0 < k < h$ (for explanation of the command exists see [2, Section 9.7]):

```
> exists{ k : k in [1..h-1] | IsZero(k*_q) };
false
```

5.4. Witt equivalence.

AreWittEquivalent(K, L)

Two fields K, L are *Witt equivalent* if their Witt rings are isomorphic. This function checks whether two fields are Witt equivalent.

Supports: finite fields, rationals, number fields, global rational function fields, global function fields, reals, complex numbers

Example. Let's see that \mathbb{C} and \mathbb{F}_2 are Witt equivalent. In the Witt rings of both fields are isomorphic to \mathbb{Z}_2 :

```
> AreWittEquivalent( ComplexField(), GF(2) );
true
> WittRing( ComplexField() ) : Maximal;
Witt ring of Complex field of precision 30, isomorphic to Residue
class ring of integers modulo 2
```

```
> WittRing( GF(2) ) : Maximal;
Witt ring of Finite field of size 2, isomorphic to Residue class
ring of integers modulo 2
```

Example. We will now consider a more complicated example. Take two quartic fields $K = \mathbb{Q}(\theta)$ where $\theta^4 - 2\theta^3 - 5\theta^2 + 6\theta - 8 = 0$ and $L = \mathbb{Q}(\eta)$ where $\eta^4 - 2\eta^3 - \eta^2 + 2\eta + 8 = 0$:

```
> _<x> := PolynomialRing(Rationals());
> K<theta> := NumberField(x^4-2*x^3-5*x^2+6*x-8);
> L<eta> := NumberField(x^4-2*x^3-x^2+2*x+8);
```

Compute the heights of both fields:

```
> Height(K);
4
> Height(L);
8
```

It follows that the torsion parts of their Witt rings differ in size. In particular the fields cannot be Witt equivalent. Indeed, they are not:

```
> AreWittEquivalent(K, L);
false
```

6. EXTENSIONS TO MAGMA'S BUILT-IN FUNCTIONS

HilbertSymbol(a, b, P)

This function computes the Hilbert symbol $(a, b)_P$, where P is either a place or a prime ideal and a, b are two elements of the same global field K . It generalizes to global function fields a built-in function of the same name.

Supports: global rational function fields, global function fields

Example. Take a function field of an elliptic curve $y^2 - x^3 + 3x + 1 = 0$ over \mathbb{F}_{257} :

```
> F := GF(257);
> FX<X> := FunctionField(F);
> _<Y> := PolynomialRing(Fx);
> K<y> := FunctionField(y^2 - x^3 + 3*x + 1);
> x := K!X;
```

Take a random place P of degree 2 and compute the Hilbert symbol $(x, y)_P$:

```
> _, P := HasRandomPlace(K, 2);
> HilbertSymbol(x, y, P);
1
```

IsLocalSquare(a, P)

Given an element a of a global field K , this function checks whether a is a square in the completion K_P . This is the same as the built in command of the same name, but in addition it allows P to be:

- an ideal in \mathbb{Z} ,
- a place of a global field (either number field or function field).

Supports: rationals, number fields, global rational function fields, global function fields

`LegendreSymbol(a, P)`

Let K be a global field, $a \in K$ its element and P a prime of K . This function computes the Legendre symbol $\left(\frac{a}{P}\right)$, i.e. it checks whether a is a quadratic residue modulo P . The function returns 0 if $a \in P$, -1 if a is not a quadratic residue, and 1 if A is a quadratic residue modulo P . The argument P can be either a place or a prime ideal of K .

Magma has a function of the same name that works over the rationals. CQF extends it to other global fields.

Supports: number fields, global rational function fields, global function fields

Example. Take a quadratic field $K = \mathbb{Q}(\sqrt{-37})$. We will compute the Legendre symbol $\left(\frac{a}{P}\right)$, where $a = 2 + \sqrt{-37}$ and P is the unique dyadic place of K .

```
> K<theta> := QuadraticField(-37);
> P := Decomposition(K, 2)[1][1];
> LegendreSymbol(2+theta, P);
1
```

REFERENCES

- [1] Wieb Bosma, John Cannon, and Catherine Playoust. The Magma algebra system. I. The user language. *J. Symbolic Comput.*, 24(3-4):235–265, 1997. Computational algebra and number theory (London, 1993).
- [2] John Cannon, Wieb Bosma, Claus Fieker, and Allan Steel. *Handbook Of Magma Functions*. School of Mathematics and Statistics, University of Sydney, 2015.
- [3] John Cannon and Catherine Playoust. *First Steps in Magma*. School of Mathematics and Statistics, University of Sydney, 1996.
- [4] T. Y. Lam. *Introduction to quadratic forms over fields*, volume 67 of *Graduate Studies in Mathematics*. American Mathematical Society, Providence, RI, 2005.
- [5] Kazimierz Szyciczek. *Bilinear algebra*, volume 7 of *Algebra, Logic and Applications*. Gordon and Breach Science Publishers, Amsterdam, 1997. An introduction to the algebraic theory of quadratic forms.

7. MIT LICENSE

Copyright (c) 2020 Przemysław Koprowski

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION

WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.