

# IMPERIAL

DEPARTMENT OF MATHEMATICS  
IMPERIAL COLLEGE LONDON

---

## Quadratic Integer Rings in Lean 4 Formalization Report

Student name: *Frankie Feng-Cheng WANG*

Email: *maths@frankie.wang*

GitHub:

*<https://github.com/FrankieeW/ClassificationOfIntegersOfQuadraticNumberFields>*

---

Course: *MATH70040-Formalising Mathematics* – Lecturer: *Dr Bhavik Mehta*

Due date: *February 27, 2026*

### Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
<b>2</b>	<b>Quadratic setup and basic structures (Base.lean)</b>	<b>3</b>
2.1	Definition 2.1 (quadratic parameter package) . . . . .	3
2.2	Definition 2.2 (ambient type) . . . . .	3
2.3	Definition 2.3 (rescaling equivalence) . . . . .	3
2.4	Definition 2.4 (trace and norm abbreviations) . . . . .	4
2.5	Definition 2.5 (rational embedding) . . . . .	4
2.6	Definition 2.6 (nonsquare rational condition) . . . . .	4
2.7	Proposition 2.7 (squarefree nontrivial implies nonsquare in $\mathbb{Z}$ ) . . . . .	5
2.8	Proposition 2.8 (parameter hypothesis gives rational nonsquare) . . . . .	5
2.9	Proposition 2.9 (field structure) . . . . .	5
<b>3</b>	<b>Trace, norm, and quadratic identity (MinimalPolynomial.lean)</b>	<b>6</b>
3.1	Theorem 3.1 (trace formula) . . . . .	6
3.2	Theorem 3.2 (norm formula) . . . . .	6
3.3	Theorem 3.3 (quadratic polynomial annihilation) . . . . .	6
<b>4</b>	<b>Non-isomorphism of distinct quadratic fields (NonIso.lean)</b>	<b>6</b>
4.1	Lemma 4.1 . . . . .	7
4.2	Lemma 4.2 . . . . .	7
4.3	Lemma 4.3 . . . . .	7
4.4	Theorem 4.4 (distinct parameters give non-isomorphic fields) . . . . .	8
<b>5</b>	<b>Half-integral normal form (HalfInt.lean)</b>	<b>9</b>
5.1	Definition 5.1 . . . . .	10
5.2	Theorem 5.2 (trace of half-integral element) . . . . .	10
5.3	Theorem 5.3 (norm of half-integral element) . . . . .	10

<b>6</b>	<b>Mod-4 analysis and parity classification (ModFourCriteria.lean)</b>	<b>10</b>
6.1	Lemma 6.1 . . . . .	11
6.2	Lemma 6.2 . . . . .	11
6.3	Lemma 6.3 . . . . .	11
6.4	Lemma 6.4 . . . . .	11
6.5	Lemma 6.5 (internal equivalence) . . . . .	12
6.6	Theorem 6.6 (main mod-4 criterion) . . . . .	12
6.7	Theorem 6.7 (forcing even–even when $d \not\equiv 1 \pmod{4}$ ) . . . . .	13
6.8	Theorem 6.8 (equivalence in non-1 mod 4 branch) . . . . .	13
6.9	Theorem 6.9 (equivalence in 1 mod 4 branch) . . . . .	14
<b>7</b>	<b>Embedding into <math>\mathbb{Q}(\sqrt{d})</math> and image characterization (ClassificationToZsqrtd.lean)</b>	<b>14</b>
7.1	Definition 7.1 (canonical embedding) . . . . .	14
7.2	Theorem 7.2 (injectivity) . . . . .	15
7.3	Definition 7.3 (equivalence with image) . . . . .	15
7.4	Theorem 7.4 (half-integral image criterion) . . . . .	15
7.5	Theorem 7.5 (classification in $d \not\equiv 1 \pmod{4}$ branch) . . . . .	16
<b>8</b>	<b>Progress and remaining work</b>	<b>16</b>

## 1. Introduction

This report documents a Lean 4 formalization of results from the algebraic-integers chapter (Lecture 2) of Boxer’s notes [1], building on Mathlib [2]. Every definition, lemma, and theorem listed below has been formally verified; the corresponding Lean source is included inline for reference.

The mathematical scope is the quadratic field

$$\mathbb{Q}(\sqrt{d}) = \{a + b\sqrt{d} \mid a, b \in \mathbb{Q}\},$$

and the ring-of-integers prerequisites needed for the classification split by  $d \pmod{4}$ . The central goal is to formalize the standard result (cf. [1, Lecture 2]):

$$\mathcal{O}_{\mathbb{Q}(\sqrt{d})} = \begin{cases} \mathbb{Z}[\sqrt{d}] & \text{if } d \not\equiv 1 \pmod{4}, \\ \mathbb{Z}\left[\frac{1 + \sqrt{d}}{2}\right] & \text{if } d \equiv 1 \pmod{4}. \end{cases}$$

**Structure.** section 2 sets up the ambient quadratic field. section 3 records the trace, norm, and minimal-polynomial identity. section 4 proves non-isomorphism of distinct quadratic fields. sections 5 and 6 develop the half-integral normal form and the mod-4 parity criterion. section 7 combines these into the element-level classification for the  $d \not\equiv 1 \pmod{4}$  branch.

**Build status (February 27, 2026):** lake build succeeds. Two sorry markers remain in ClassificationToZsqrtd.lean<sup>1</sup> (the forward and reverse directions linking the ring-of-integers isomorphism to the mod-4 condition); all other files are sorry-free.

---

<sup>1</sup>It is not in submission ZIP

**Development note (motivation and current strategy).** The current formalization strategy is intentionally pragmatic. Historically, while preparing and submitting PR work around `Zsqrtd`, I noticed that `QuadraticAlgebra` provides a workable ambient path for this project stage. So for the half-integral classification workflow I currently use `QuadraticAlgebra` and define

$$\mathbb{Q}\text{sqrtd}(d) := \text{QuadraticAlgebra } \mathbb{Q} (d : \mathbb{Q}) 0.$$

This is a coarse-grained but effective bridge for now. The reason is that the integral model needed for the  $d \equiv 1 \pmod{4}$  branch, especially the  $\mathbb{Z}[\frac{1+\sqrt{1+4k}}{2}]$ -style object, is not yet packaged as a ready drop-in component in the way this project needs; see the related discussion: Zulip discussion.

## 2. Quadratic setup and basic structures (Base.lean)

This section sets up the type-level infrastructure for working with  $\mathbb{Q}(\sqrt{d})$ . Following [1, Lecture 2, §1], we first pin down the admissible parameters  $d$ , then construct the ambient field together with its trace, norm, and embedding.

**2.1. Definition 2.1 (quadratic parameter package).** Lean name: `IsQuadraticParam`.

For  $d \in \mathbb{Z}$ , we define a proposition requiring

$$d \neq 0, \quad d \neq 1, \quad d \text{ squarefree}.$$

These are the standard hypotheses ensuring that  $\mathbb{Q}(\sqrt{d})$  is a genuine quadratic extension of  $\mathbb{Q}$  (cf. [1, Lecture 2, Definition 2.1]).

```

21 class IsQuadraticParam (d : ℤ) : Prop where
22   /-- `d ≠ 0` and `d ≠ 1` ensure `ℚ(√d)` is not just `ℚ`. -/
23   ne_zero : d ≠ 0
24   ne_one : d ≠ 1
25   /-- `Squarefree d` gives a canonical representative for the field. -/
26   squarefree : Squarefree d

```

**2.2. Definition 2.2 (ambient type).** Lean name: `Qsqrtd`.

The type

$$\mathbb{Q}\text{sqrtd}(d) := \text{QuadraticAlgebra } \mathbb{Q} (d : \mathbb{Q}) 0$$

serves as the formal model of  $\mathbb{Q}(\sqrt{d})$ .

```

29 abbrev Qsqrtd (d : ℤ) : Type := QuadraticAlgebra ℚ (d : ℚ) 0

```

**2.3. Definition 2.3 (rescaling equivalence).** Lean name: `rescale`.

Given  $a \in \mathbb{Q}^\times$ , there is an algebra isomorphism

$$\mathbb{Q}(\sqrt{d}) \cong \mathbb{Q}(\sqrt{a^2 d}).$$

In coordinates this is

$$(r, s) \mapsto (r, sa^{-1}), \quad (r, t) \mapsto (r, ta).$$

This captures the classical fact that replacing  $d$  by  $a^2d$  does not change the underlying quadratic field.

```

35 def rescale (d : ℚ) (a : ℚ) (ha : a ≠ 0) :
36   QuadraticAlgebra ℚ d 0 ≈a[ℚ] QuadraticAlgebra ℚ (a ^ 2 * d) 0 := by
37   have h1d : (1 : QuadraticAlgebra ℚ d 0) = ⟨1, 0⟩ := by ext <;> rfl
38   have h1a : (1 : QuadraticAlgebra ℚ (a ^ 2 * d) 0) = ⟨1, 0⟩ := by
39     ext <;> rfl
40   exact AlgEquiv.ofLinearEquiv
41     { toFun := fun x => ⟨x.re, x.im * a-1⟩
42       invFun := fun y => ⟨y.re, y.im * a⟩
43       map_add' := by intro x y; ext <;> simp [add_mul]
44       map_smul' := by intro c x; ext <;> simp [mul_assoc]
45       left_inv := by
46         intro x; ext <;> simp [mul_assoc, inv_mul_cancel₀ ha]
47       right_inv := by
48         intro y; ext <;> simp [mul_assoc, mul_inv_cancel₀ ha] }
49   (by simp [h1d, h1a])
50   (by intro x y; ext <;> simp <;> field_simp)

```

**2.4. Definition 2.4 (trace and norm abbreviations).** Lean names: `trace`, `norm'`.

For  $x \in \mathbb{Q}\text{sqrtd}(d)$ , define

$$\text{tr}(x) := x + \bar{x} \in \mathbb{Q}, \quad N(x) := x\bar{x} \in \mathbb{Q}.$$

These correspond to the trace and norm of the quadratic extension  $\mathbb{Q}(\sqrt{d})/\mathbb{Q}$ , computed via the Galois conjugation  $\sqrt{d} \mapsto -\sqrt{d}$ .

```

53 abbrev trace {d : ℤ} (x : ℚsqrtd d) : ℚ := x.re + (star x).re
54
55 /-- Norm on `ℚsqrtd d`. -/
56 abbrev norm' {d : ℤ} (x : ℚsqrtd d) : ℚ := QuadraticAlgebra.norm x

```

**2.5. Definition 2.5 (rational embedding).** Lean name: `embed`.

The canonical inclusion

$$\mathbb{Q} \hookrightarrow \mathbb{Q}(\sqrt{d})$$

is implemented by the algebra map.

```

59 abbrev embed (r : ℚ) : ℚsqrtd d := algebraMap ℚ (ℚsqrtd d) r

```

**2.6. Definition 2.6 (nonsquare rational condition).** Lean name: `IsNonsquareRat`.

For integer  $d$ , define

$$\forall r \in \mathbb{Q}, \quad r^2 \neq d.$$

```

62 class IsNonsquareRat (d : ℤ) : Prop where
63   nonsquare : ∀ r : ℚ, r ^ 2 ≠ (d : ℚ)

```

**2.7. Proposition 2.7** (squarefree nontrivial implies nonsquare in  $\mathbb{Z}$ ). **Lean name:** `not_isSquare_int`.

Under `IsQuadraticParam` hypotheses,

$$\neg \text{IsSquare}(d) \quad \text{in } \mathbb{Z}.$$

This excludes degeneration of the quadratic extension.

```

66 lemma not_isSquare_int (d : ℤ) [IsQuadraticParam d] : ¬ IsSquare d := by
67   intro hdSq
68   rcases hdSq with <z, hz>
69   by_cases huz : IsUnit z
70   · rcases Int.isUnit_iff.mp huz with hz1 | hz1
71     · have : d = 1 := by simp [hz1] using hz
72       exact (IsQuadraticParam.ne_one (d := d)) this
73     · have : d = 1 := by simp [hz1] using hz
74       exact (IsQuadraticParam.ne_one (d := d)) this
75   · have hsqz2 : Squarefree (z ^ 2) := by
76     simp [hz, pow_two] using (IsQuadraticParam.squarefree (d := d))
77   have h01 : (2 : ℕ) = 0 ∨ (2 : ℕ) = 1 :=
78     Squarefree.eq_zero_or_one_of_pow_of_not_isUnit (x := z) (n := 2)
79   ↪ hsqz2 huz
   norm_num at h01

```

**2.8. Proposition 2.8** (parameter hypothesis gives rational nonsquare). **Lean instance:** `instance (d) [IsQuadraticParam d] : IsNonsquareRat d`.

From the integer nonsquare result and transfer lemmas between  $\mathbb{Z}$  and  $\mathbb{Q}$ , one obtains

$$\forall r \in \mathbb{Q}, r^2 \neq d.$$

```

81 instance (d : ℤ) [IsQuadraticParam d] : IsNonsquareRat d := by
82   refine <?_>
83   intro r hr
84   have hsqQ : IsSquare ((d : ℤ) : ℚ) := <r, by simp [pow_two] using
85     ↪ hr.symm>
86   have hsqZ : IsSquare d := (Rat.isSquare_intCast_iff).1 hsqQ
87   exact (not_isSquare_int d) hsqZ

```

**2.9. Proposition 2.9** (field structure). **Lean instance:** `Field (Qsqrtd d)` under `IsNonsquareRat d`.

If  $d$  is rationally nonsquare, then  $\mathbb{Q}(\sqrt{d})$  is a field. This is the key structural fact that upgrades the ring `Qsqrtd(d)` to a number field.

```

88 instance {d : ℤ} [IsNonsquareRat d] : Field (Qsqrtd d) := by
89   letI : Fact (∀ r : ℚ, r ^ 2 ≠ (d : ℚ) + 0 * r) := <by
90     intro r hr
91     exact (IsNonsquareRat.nonsquare (d := d) r) (by simp [hr])
92   >
93   infer_instance

```

### 3. Trace, norm, and quadratic identity (MinimalPolynomial.lean)

With the ambient field in place, we now establish the explicit trace and norm formulas and verify the characteristic polynomial identity. These are the algebraic prerequisites for the integrality criterion used in section 6.

#### 3.1. Theorem 3.1 (trace formula). Lean name: trace\_eq\_two\_re.

For  $x = a + b\sqrt{d} \in \mathbb{Q}(\sqrt{d})$ ,

$$\text{tr}(x) = 2a.$$

```

8 theorem trace_eq_two_re {d : ℤ} (x : Qsqrtd d) : trace x = 2 * x.re := by
9   have star_re : (star x).re = x.re := by
10     simp [star, star]
11     simp [trace, star_re]
12   ring

```

#### 3.2. Theorem 3.2 (norm formula). Lean name: norm\_eq\_sqr\_minus\_d\_sqr.

Writing  $x = a + b\sqrt{d}$ , one has

$$N(x) = a^2 - db^2.$$

```

15 theorem norm_eq_sqr_minus_d_sqr {d : ℤ} (x : Qsqrtd d) :
16   norm' x = x.re ^ 2 - (d : ℚ) * x.im ^ 2 := by
17     simp [norm', QuadraticAlgebra.norm]
18   ring
19

```

#### 3.3. Theorem 3.3 (quadratic polynomial annihilation). Lean name: aeval\_eq\_zero\_of\_quadratic.

Each  $x \in \mathbb{Q}(\sqrt{d})$  satisfies

$$x^2 - \text{tr}(x)x + N(x) = 0.$$

In other words, every element of the quadratic extension is a root of the polynomial  $T^2 - \text{tr}(x)T + N(x)$ , which is its minimal polynomial over  $\mathbb{Q}$  when  $x \notin \mathbb{Q}$ .

```

21 theorem aeval_eq_zero_of_quadratic (d : ℤ) (x : Qsqrtd d) :
22   x * x - (algebraMap ℚ (Qsqrtd d) (x.trace)) • x + (algebraMap ℚ (Qsqrtd
23     ↪ d) (norm' x)) = 0 := by
24   ext <;> simp [trace, norm', QuadraticAlgebra.norm, star, smul_eq_mul] <;>
25   ↪ ring

```

### 4. Non-isomorphism of distinct quadratic fields (NonIso.lean)

Before turning to the ring-of-integers classification, we address a natural prerequisite: distinct squarefree parameters really do give distinct fields. This section formalizes the classical result that if  $d_1 \neq d_2$  are both valid quadratic parameters, then  $\mathbb{Q}(\sqrt{d_1})$  and  $\mathbb{Q}(\sqrt{d_2})$  are not isomorphic as  $\mathbb{Q}$ -algebras.

**4.1. Lemma 4.1. Lean name:** `not_isSquare_neg_one_rat`.

$-1$  is not a square in  $\mathbb{Q}$ .

```

8 isomorphic (as `Q`-algebras) whenever `d₁ ≠ d₂` and both are squarefree
9 nonzero integers.
10
11 The proof strategy is:

```

**4.2. Lemma 4.2. Lean name:**

`nat_eq_one_of_squarefree_intcast_of_isSquare`.

If  $m \in \mathbb{N}$ ,  $(m : \mathbb{Z})$  is squarefree, and  $(m : \mathbb{Z})$  is a square, then

$$m = 1.$$

```

14 3. Expanding in the basis `{1, √d₂}`, the imaginary part equation forces
15 `x.re * x.im = 0`, giving two cases.
16 4. In either case we derive that `d₁/d₂` is a square in `Q`.
17 5. A denominator argument using squarefreeness shows `d₂ | d₁` and `d₁ |
   ↪ `d₂`,
18 so `d₁` and `d₂` are associates in `Z`.
19 6. Associates in `Z` means `d₁ = d₂` or `d₁ = -d₂`; the former contradicts
20 `d₁ ≠ d₂`, and the latter implies `-1` is a square in `Q`, a
   ↪ contradiction.
21 -/
22
23 namespace ClassificationOfIntegersOfQuadraticNumberFields
24 namespace QsqrtD
25
26 /-- `-1` is not a square in `Q`. Any rational square `r * r` is nonneg, but
   ↪ `-1 < 0`. -/
27 lemma not_isSquare_neg_one_rat : ¬ IsSquare (- (1 : Q)) := by
28   rintro <r, hr>

```

**4.3. Lemma 4.3. Lean name:** `int_dvd_of_ratio_square`.

Let  $d_2 \neq 0$ , with  $d_2$  squarefree. If

$$\frac{d_1}{d_2} \in \mathbb{Q}$$

is a square in  $\mathbb{Q}$ , then

$$d_2 \mid d_1.$$

This is the key divisibility extraction used in the non-isomorphism argument.

```

31
32 /-- If `m : N` is squarefree as an integer and is a square in `Z`, then `m
   ↪ ` = 1`.
33 A squarefree integer that is also a perfect square must be a unit. -/
34 lemma nat_eq_one_of_squarefree_intcast_of_isSquare (m : N)
35   (hsm : Squarefree (m : Z)) (hsq : IsSquare (m : Z)) : m = 1 := by
36   rcases hsq with <z, hz>
37   -- Case split: is the square root `z` a unit (i.e. ±1)?
38   by_cases huz : IsUnit z

```

```

39 · -- If z is a unit, then m = z * z = (±1)² = 1.
40 rcases Int.isUnit_iff.mp huz with hz1 | hz1
41 · have hmz : (m : ℤ) = 1 := by simpa [hz1] using hz
42   norm_num at hmz
43   exact hmz
44 · have hmz : (m : ℤ) = 1 := by simpa [hz1] using hz

```

**4.4. Theorem 4.4 (distinct parameters give non-isomorphic fields).** Lean name: `quadratic_fields_not_iso`.

Assume  $d_1, d_2$  satisfy `IsQuadraticParam` and  $d_1 \neq d_2$ . Then

$$\mathbb{Q}(\sqrt{d_1}) \not\cong_{\mathbb{Q}} \mathbb{Q}(\sqrt{d_2}).$$

The proof follows a standard reduction: an assumed isomorphism forces square-ratio conditions implying divisibility both ways (via Lemma 4.3); associatedness yields either equality or sign flip; the sign-flip branch reduces to  $-1$  being a rational square (Lemma 4.1), contradiction.

```

46   exact hmz
47 · -- If z is not a unit, then z² being squarefree is impossible
48   -- (squarefree means every factor appears at most once, but z appears
49   ↪ twice).
49   have hsqz2 : Squarefree (z ^ 2) := by simpa [hz, pow_two] using hsm
50   have h01 : (2 : ℕ) = 0 ∨ (2 : ℕ) = 1 :=
51     Squarefree.eq_zero_or_one_of_pow_of_not_isUnit (x := z) (n := 2)
52     ↪ hsqz2 huz
53   norm_num at h01
54 /-- If `d₂` is squarefree and nonzero, and `d₁ / d₂` is a square in `ℚ`,
55   ↪ then `d₂ | d₁`.
56   The key idea: write `d₁/d₂` in lowest terms as `p/q`. The denominator
57   ↪ `q` divides `d₂`,
58   so `q` is also squarefree. But `d₁/d₂` being a square forces `q` to be
59   ↪ a square as well.
60   A squarefree square must be 1, meaning `d₂` divides `d₁`. -/
61 lemma int_dvd_of_ratio_square (d₁ d₂ : ℤ) (hd₂ : d₂ ≠ 0)
62   (hsq_d₂ : Squarefree d₂) (hr : IsSquare ((d₁ : ℚ) / (d₂ : ℚ))) : d₂ |
63   ↪ d₁ := by
64   -- The denominator of d₁/d₂ (in lowest terms) is a square.
65   have hsq_den_nat : IsSquare (((d₁ : ℚ) / (d₂ : ℚ)).den) :=
66     ↪ (Rat.isSquare_iff.mp hr).2
67   -- Lift the squareness to ℤ.
68   have hsq_den_int : IsSquare (((d₁ : ℚ) / (d₂ : ℚ)).den : ℤ) := by
69     rcases hsq_den_nat with <n, hn>
70     refine <(n : ℤ), by exact_mod_cast hn>
71   -- The denominator of d₁/d₂ divides d₂.
72   have hden_dvd : (((d₁ : ℚ) / (d₂ : ℚ)).den : ℤ) | d₂ := by
73     simpa [← Rat.divInt_eq_div] using (Rat.den_dvd d₁ d₂)
74   -- Since it divides the squarefree d₂, the denominator is itself
75   ↪ squarefree.
76   have hsqf_den_int : Squarefree (((d₁ : ℚ) / (d₂ : ℚ)).den : ℤ) :=
77     Squarefree.squarefree_of_dvd hden_dvd hsq_d₂
78   -- Squarefree + square denominator = 1, so d₂ | d₁.
79   have hden1_nat : ((d₁ : ℚ) / (d₂ : ℚ)).den = 1 :=

```



```

75   nat_eq_one_of_squarefree_intcast_of_isSquare _ hsqf_den_int hsq_den_int
76   exact (Rat.den_div_intCast_eq_one_iff d1 d2 hd2).1 hden1_nat
77
78 /-- **Main theorem**: Distinct squarefree parameters define non-isomorphic
79 ↪ quadratic fields.
80
81   If  $d_1 \neq d_2$  are both squarefree nonzero integers, then there is no
82 ↪  $\mathbb{Q}$ -algebra
83 isomorphism  $\mathbb{Q}(\sqrt{d_1}) \simeq_a [\mathbb{Q}] \mathbb{Q}(\sqrt{d_2})$ . -/
84 theorem quadratic_fields_not_iso
85   (d1 d2 :  $\mathbb{Z}$ ) [IsQuadraticParam d1] [IsQuadraticParam d2]
86   (hneq : d1  $\neq$  d2) :
87    $\neg$  Nonempty (Qsqrtd d1  $\simeq_a$  [ $\mathbb{Q}$ ] Qsqrtd d2) := by
88   rintro <e>
89   -- Let  $x = e(\sqrt{d_1}) \in \mathbb{Q}(\sqrt{d_2})$ . Write  $x = a + b\sqrt{d_2}$  for some  $a, b \in \mathbb{Q}$ .
90   let x : Qsqrtd d2 := e <0, 1>
91   -- Since  $e$  is a ring homomorphism and  $(\sqrt{d_1})^2 = d_1$ , we get  $x^2 = d_1$  in
92   ↪  $\mathbb{Q}(\sqrt{d_2})$ .
93   have hx : x * x = (d1 : Qsqrtd d2) := by
94     change e <0, 1> * e <0, 1> = (d1 : Qsqrtd d2)
95     calc
96       e <0, 1> * e <0, 1> = e ((<0, 1> : Qsqrtd d1) * <0, 1>) := by
97         symm
98         exact e.map_mul _ _
99       _ = e (d1 : Qsqrtd d1) := by
100         congr 1
101         ext <;> simp [Qsqrtd]
102       _ = (d1 : Qsqrtd d2) := by simp
103   -- Expanding  $x^2 = (a + b\sqrt{d_2})^2 = (a^2 + b^2d_2) + 2ab\sqrt{d_2}$ , the imaginary part
104   ↪ gives  $2ab = 0$ .
105   have him0 : (x * x).im = 0 := by
106     have him := congrArg QuadraticAlgebra.im hx
107     simpa [Qsqrtd] using him
108   have hsum : x.re * x.im + x.im * x.re = 0 := by
109     simpa [Qsqrtd, mul_assoc, mul_comm, mul_left_comm] using him0
110   -- So either  $a = 0$  or  $b = 0$ .
111   have hprod : x.re * x.im = 0 := by nlinarith [hsum]
112   -- In either case,  $d_1/d_2$  is a perfect square in  $\mathbb{Q}$ .
113   have hratio : IsSquare ((d1 :  $\mathbb{Q}$ ) / (d2 :  $\mathbb{Q}$ )) := by
114     rcases mul_eq_zero.mp hprod with hre | him
115     · -- Case  $a = 0$ : then  $x = b\sqrt{d_2}$ , so  $x^2 = b^2d_2 = d_1$ , giving  $d_1/d_2 = b^2$ .
116       refine <x.im, ?_>
117       have hre0 : (x * x).re = d1 := by
118         have hre' := congrArg QuadraticAlgebra.re hx
119         simpa [Qsqrtd] using hre'
120       have hmain : (d2 :  $\mathbb{Q}$ ) * (x.im ^ 2) = d1 := by
121         simpa [Qsqrtd, hre, pow_two, mul_assoc, mul_comm, mul_left_comm]
122       ↪ using hre0

```

## 5. Half-integral normal form (HalfInt.lean)

Every element of  $\mathbb{Q}(\sqrt{d})$  that is integral over  $\mathbb{Z}$  can be written in the form  $\frac{a'+b'\sqrt{d}}{2}$  with  $a', b' \in \mathbb{Z}$  (cf. [1, Lecture 2, proof of Theorem 2.5]). This section sets up the half-integral representation and computes its trace and norm explicitly.

**5.1. Definition 5.1.** Lean name: `halfInt`.

For integers  $a', b', d$ , define

$$\text{halfInt}(d, a', b') := \frac{a' + b'\sqrt{d}}{2} \in \mathbb{Q}(\sqrt{d}).$$

```
8 def halfInt (d : ℤ) (a' b' : ℤ) : ℚsqrtd d :=
9   <a' / 2, b' / 2>
```

**5.2. Theorem 5.2** (trace of half-integral element). Lean name: `trace_halfInt`.

$$\text{tr}\left(\frac{a' + b'\sqrt{d}}{2}\right) = a'.$$

```
12 theorem trace_halfInt (d a' b' : ℤ) : trace (halfInt d a' b') = a' := by
13   have : (halfInt d a' b').re = a' / 2 := rfl
14   rw [trace_eq_two_re, this]
15   ring
16
```

**5.3. Theorem 5.3** (norm of half-integral element). Lean name: `norm_halfInt`.

$$N\left(\frac{a' + b'\sqrt{d}}{2}\right) = \frac{a'^2 - db'^2}{4}.$$

An element  $\frac{a' + b'\sqrt{d}}{2}$  is integral over  $\mathbb{Z}$  if and only if both  $\text{tr}$  and  $N$  are integers, which by the formulas above reduces to  $a' \in \mathbb{Z}$  (automatic) and  $4 \mid (a'^2 - db'^2)$ . This divisibility condition is the starting point for the mod-4 analysis in section 6.

```
18 theorem norm_halfInt (d a' b' : ℤ) :
19   norm' (halfInt d a' b') = (a' ^ 2 - (d : ℚ) * b' ^ 2) / 4 := by
20   have re_eq : (halfInt d a' b').re = a' / 2 := rfl
21   have im_eq : (halfInt d a' b').im = b' / 2 := rfl
22   rw [norm_eq_sqr_minus_d_sqr, re_eq, im_eq]
23   ring
24
```

**6. Mod-4 analysis and parity classification** (`ModFourCriteria.lean`)

This section reduces the integrality condition  $4 \mid (a'^2 - db'^2)$  to explicit congruence constraints on  $a'$ ,  $b'$ , and  $d$ . The analysis mirrors the key exercise pattern from [1, Lecture 2]: a case split on the parities of  $a'$  and  $b'$ , combined with the mod-4 residue of  $d$ .

**6.1. Lemma 6.1. Lean name:** `squarefree_int_not_dvd_four`.

If  $d \in \mathbb{Z}$  is squarefree, then

$$4 \nmid d.$$

Indeed,  $4 = 2^2$  dividing  $d$  would contradict squarefreeness.

```

9 lemma squarefree_int_not_dvd_four (d : ℤ) (hd : Squarefree d) : ¬ (4 : ℤ) ∣
  ↪ d := by
10   intro h
11   have h22 : (2 : ℤ) * 2 ∣ d := by
12     obtain ⟨k, hk⟩ := h
13     exact ⟨k, by omega⟩
14   have hunit : IsUnit (2 : ℤ) := hd 2 h22
15   exact absurd (Int.isUnit_iff.mp hunit) (by omega)

```

**6.2. Lemma 6.2. Lean name:** `squarefree_int_emod_four`.

If  $d$  is squarefree, then

$$d \bmod 4 \in \{1, 2, 3\}.$$

This is an immediate corollary of Lemma 6.1.

```

18 lemma squarefree_int_emod_four (d : ℤ) (hd : Squarefree d) :
19   d % 4 = 1 ∨ d % 4 = 2 ∨ d % 4 = 3 := by
20   have hnd : ¬ (4 : ℤ) ∣ d := squarefree_int_not_dvd_four d hd
21   omega

```

**6.3. Lemma 6.3. Lean name:** `Int.sq_emod_four_of_even`.

If  $2 \mid n$ , then

$$n^2 \equiv 0 \pmod{4}.$$

```

24 lemma Int.sq_emod_four_of_even (n : ℤ) (h : 2 ∣ n) : n ^ 2 % 4 = 0 := by
25   obtain ⟨k, rfl⟩ := h
26   ring_nf
27   omega

```

**6.4. Lemma 6.4. Lean name:** `Int.sq_emod_four_of_odd`.

If  $2 \nmid n$ , then

$$n^2 \equiv 1 \pmod{4}.$$

Lemmas 6.3 and 6.4 together show that the mod-4 residue of a square is determined entirely by its parity, which is the key arithmetic input for the main criterion.

```

30 lemma Int.sq_emod_four_of_odd (n : ℤ) (h : ¬ 2 ∣ n) : n ^ 2 % 4 = 1 := by
31   set k := n / 2
32   have hk : n = 2 * k + 1 := by omega
33   rw [hk]
34   ring_nf
35   omega

```

**6.5. Lemma 6.5 (internal equivalence).** Lean name: `div4_iff_mod` (private).For integers  $a', b', d$ ,

$$4 \mid (a'^2 - db'^2) \iff (a'^2 - db'^2) \bmod 4 = 0.$$

```

37 private lemma div4_iff_mod (a' b' d : ℤ) :
38   4 ∣ (a' ^ 2 - d * b' ^ 2) ↔ (a' ^ 2 - d * b' ^ 2) % 4 = 0 := by
39   omega

```

**6.6. Theorem 6.6 (main mod-4 criterion).** Lean name:`dvd_four_sub_sq_iff_even_even_or_odd_odd_mod_four_one`.Assume  $d$  squarefree. Then

$$4 \mid (a'^2 - db'^2) \iff (2 \mid a' \ \& \ 2 \mid b') \vee (2 \nmid a' \ \& \ 2 \nmid b' \ \& \ d \equiv 1 \pmod{4}).$$

The proof proceeds by exhaustive case analysis on the parities of  $a'$  and  $b'$ : the mixed-parity cases are ruled out by Lemmas 6.3–6.4 and the constraint  $d \bmod 4 \in \{1, 2, 3\}$ ; the odd–odd case forces  $d \equiv 1 \pmod{4}$ .

```

42 theorem dvd_four_sub_sq_iff_even_even_or_odd_odd_mod_four_one (d a' b' : ℤ)
43   ↪ (hd : Squarefree d) :
44   4 ∣ (a' ^ 2 - d * b' ^ 2) ↔
45     (2 ∣ a' ^ 2 ∨ 2 ∣ b') ∨ (¬ 2 ∣ a' ^ 2 ∧ ¬ 2 ∣ b' ^ 2 ∧ d % 4 = 1) := by
46   have hd4 := squarefree_int_emod_four d hd
47   constructor
48   · intro hdvd
49     have hmod : (a' ^ 2 - d * b' ^ 2) % 4 = 0 := (div4_iff_mod a' b' d).1
50     ↪ hdvd
51     have even_odd_impossible (ha : 2 ∣ a') (hb : ¬ 2 ∣ b') : False := by
52       have hmod' := hmod
53       have ha_eq : a' ^ 2 = 4 * (a' ^ 2 / 4) := by
54         have ha2 : a' ^ 2 % 4 = 0 := Int.sq_emod_four_of_even a' ha
55         omega
56       have hb_eq : b' ^ 2 = 4 * (b' ^ 2 / 4) + 1 := by
57         have hb2 : b' ^ 2 % 4 = 1 := Int.sq_emod_four_of_odd b' hb
58         omega
59       rw [hb_eq] at hmod'
60       ring_nf at hmod'
61       rcases hd4 with hd1 | hd2 | hd3 <|> omega
62     have odd_even_impossible (ha : ¬ 2 ∣ a') (hb : 2 ∣ b') : False := by
63       have hmod' := hmod
64       have ha_eq : a' ^ 2 = 4 * (a' ^ 2 / 4) + 1 := by
65         have ha2 : a' ^ 2 % 4 = 1 := Int.sq_emod_four_of_odd a' ha
66         omega
67       have hb_eq : b' ^ 2 = 4 * (b' ^ 2 / 4) := by
68         have hb2 : b' ^ 2 % 4 = 0 := Int.sq_emod_four_of_even b' hb
69         omega
70       rw [ha_eq, hb_eq] at hmod'
71       ring_nf at hmod'
72       rcases hd4 with hd1 | hd2 | hd3 <|> omega
73     have odd_odd_mod_four_one (ha : ¬ 2 ∣ a') (hb : ¬ 2 ∣ b') : d % 4 = 1
74     ↪ := by
75       have hmod' := hmod
76       have ha_eq : a' ^ 2 = 4 * (a' ^ 2 / 4) + 1 := by

```

```

74     have ha2 : a' ^ 2 % 4 = 1 := Int.sq_emod_four_of_odd a' ha
75     omega
76     have hb_eq : b' ^ 2 = 4 * (b' ^ 2 / 4) + 1 := by
77       have hb2 : b' ^ 2 % 4 = 1 := Int.sq_emod_four_of_odd b' hb
78       omega
79       rw [ha_eq, hb_eq] at hmod'
80       ring_nf at hmod'
81       omega
82     by_cases ha : 2 ∣ a' <|> by_cases hb : 2 ∣ b'
83     · exact Or.inl <ha, hb>
84     · exfalse
85       exact even_odd_impossible ha hb
86     · exfalse
87       exact odd_even_impossible ha hb
88     · exact Or.inr <ha, hb, odd_odd_mod_four_one ha hb>
89   · intro h
90     rcases h with <ha, hb> | <ha, hb, hd1>
91     · obtain <p, rfl> := ha
92       obtain <q, rfl> := hb
93       exact <p ^ 2 - d * q ^ 2, by ring>
94     · have ha_eq : a' = 2 * (a' / 2) + 1 := by omega
95       have hb_eq : b' = 2 * (b' / 2) + 1 := by omega
96       rw [ha_eq, hb_eq]
97       ring_nf
98       have hd_eq : d = 4 * (d / 4) + 1 := by omega
99       rw [hd_eq]
100      ring_nf
101      omega

```

**6.7. Theorem 6.7 (forcing even-even when  $d \not\equiv 1 \pmod{4}$ ).** Lean name: even\_even\_of\_dvd\_four\_sub\_sq\_of\_ne\_one\_mod\_four.

If  $d$  is squarefree and  $d \not\equiv 1 \pmod{4}$ , then

$$4 \mid (a'^2 - db'^2) \implies 2 \mid a' \text{ and } 2 \mid b'.$$

```

104 theorem even_even_of_dvd_four_sub_sq_of_ne_one_mod_four (d a' b' : ℤ) (hd :
105   ↪ Squarefree d)
106   (hd4 : d % 4 ≠ 1) (h : 4 ∣ (a' ^ 2 - d * b' ^ 2)) :
107   2 ∣ a' ^ 2 ∣ b' := by
108   rcases (dvd_four_sub_sq_iff_even_even_or_odd_odd_mod_four_one d a' b'
109     ↪ hd).mp h with
110     hab | <_, _, hd1>
111   · exact hab
112   · exact absurd hd1 hd4

```

**6.8. Theorem 6.8 (equivalence in non-1 mod 4 branch).** Lean name: dvd\_four\_sub\_sq\_iff\_even\_even\_of\_ne\_one\_mod\_four.

If  $d$  is squarefree and  $d \not\equiv 1 \pmod{4}$ , then

$$4 \mid (a'^2 - db'^2) \iff (2 \mid a' \text{ \& } 2 \mid b').$$

```

113 theorem dvd_four_sub_sq_iff_even_even_of_ne_one_mod_four (d a' b' : ℤ) (hd
114   ↪ Squarefree d)
115   (hd4 : d % 4 ≠ 1) :

```

```

115   4 | (a' ^ 2 - d * b' ^ 2) ↔ (2 | a' ^ 2 | b') := by
116   constructor
117   · intro h
118     exact even_even_of_dvd_four_sub_sq_of_ne_one_mod_four d a' b' hd hd4 h
119   · intro h
120     exact (dvd_four_sub_sq_iff_even_even_or_odd_odd_mod_four_one d a' b'
      ↪ hd).2 (Or.inl h)

```

**6.9. Theorem 6.9 (equivalence in  $1 \bmod 4$  branch).** Lean name: `dvd_four_sub_sq_iff_same_parity_of_one_mod_four`.

If  $d$  is squarefree and  $d \equiv 1 \pmod{4}$ , then

$$4 \mid (a'^2 - db'^2) \iff a' \equiv b' \pmod{2}.$$

Together with section 6.6, Theorems 6.8 and 6.9 give the complete parity characterization: in the  $d \not\equiv 1$  branch only the “both even” case survives, while in the  $d \equiv 1$  branch the “same parity” condition captures both the even–even and odd–odd cases.

```

122 /-- If `d % 4 = 1`, divisibility by `4` is equivalent to same parity. -/
123 theorem dvd_four_sub_sq_iff_same_parity_of_one_mod_four (d a' b' : ℤ) (hd :
      ↪ Squarefree d)
124   (hd4 : d % 4 = 1) :
125   4 | (a' ^ 2 - d * b' ^ 2) ↔ a' % 2 = b' % 2 := by
126   rw [dvd_four_sub_sq_iff_even_even_or_odd_odd_mod_four_one d a' b' hd]
127   constructor
128   · rintro (⟨ha, hb⟩ | ⟨ha, hb, _⟩) <;> omega
129   · intro h
130     by_cases ha : 2 | a'
131     · left
132       exact ⟨ha, by omega⟩
133     · right
134       exact ⟨ha, by omega, hd4⟩
135

```

## 7. Embedding into $\mathbb{Q}(\sqrt{d})$ and image characterization (ClassificationToZsqrtd.lean)

With the mod-4 arithmetic in hand, we now connect it to the algebraic structure. The strategy is to embed Mathlib’s  $\mathbb{Z}[\sqrt{d}]$  (the type `Zsqrtd`) into our ambient field model and characterize its image in terms of the half-integral representation from section 5.

**7.1. Definition 7.1 (canonical embedding).** Lean name: `zsqrtdToQsqrtd`.

Define the ring map

$$\iota_d : \mathbb{Z}[\sqrt{d}] \longrightarrow \mathbb{Q}(\sqrt{d}), \quad m + n\sqrt{d} \longmapsto m + n\sqrt{d},$$

with coefficients interpreted in  $\mathbb{Q}$ .

```

12 /-- The canonical embedding `ℤ[√d] → ℚ(√d)` into the rational quadratic
13 ↪ algebra model. -/
14 def zsqrtdToQsqrtd (d : ℤ) : ℤ[√d] →+* Qsqrtd d where

```

```

15   toFun := fun z => ⟨(z.re : ℚ), (z.im : ℚ)⟩
16   map_one' := by
17     ext <;> rfl
18   map_mul' := by
19     intro z w
20     ext <;> simp [mul_assoc, mul_comm, mul_left_comm]

```

**7.2. Theorem 7.2 (injectivity).** Lean name: `zsqrtdToQsqrtd_injective`.

$$\iota_d(z_1) = \iota_d(z_2) \implies z_1 = z_2.$$

```

23
24   /-- The canonical embedding `ℤ√d → ℚ(√d)` is injective. -/
25   theorem zsqrtdToQsqrtd_injective (d : ℤ) : Function.Injective
26     ↪ (zsqrtdToQsqrtd d) := by
27     intro z w hzw
28     ext

```

**7.3. Definition 7.3 (equivalence with image).** Lean name: `zsqrtdEquivRangeQsqrtd`.

There is a ring isomorphism

$$\mathbb{Z}[\sqrt{d}] \cong \text{im}(\iota_d).$$

```

30
31   /-- `ℤ√d` is ring-isomorphic to its image in `ℚ(√d)`. -/
32   noncomputable def zsqrtdEquivRangeQsqrtd (d : ℤ) : ℤ√d ≃+* (zsqrtdToQsqrtd
33     ↪ d).range := by
34     refine RingEquiv.ofBijective (zsqrtdToQsqrtd d).rangeRestrict ?_
35     constructor
36     · intro z w hzw
37       exact zsqrtdToQsqrtd_injective d (Subtype.ext_iff.mp hzw)
38     · intro x
39       rcases x.property with ⟨z, hz⟩

```

**7.4. Theorem 7.4 (half-integral image criterion).** Lean name: `halfInt_mem_range_zsqrtdToQsqrtd_iff_even_even`.

For integers  $a', b'$ ,

$$\frac{a' + b'\sqrt{d}}{2} \in \text{im}(\iota_d) \iff 2 \mid a' \text{ and } 2 \mid b'.$$

This is the bridge between the half-integral normal form and the  $\mathbb{Z}[\sqrt{d}]$ -representability question.

```

41
42   /-- A half-integral element is in the image of `ℤ√d` iff both numerators
43     ↪ are even. -/
44   theorem halfInt_mem_range_zsqrtdToQsqrtd_iff_even_even (d a' b' : ℤ) :
45     (∃ z : ℤ√d, zsqrtdToQsqrtd d z = halfInt d a' b') ↔ (2 ∣ a' ∧ 2 ∣ b')
46     ↪ := by
47     constructor
48     · rintro ⟨z, hz⟩

```

```

47   have hm : (a' : ℚ) / 2 = z.re := by
48     simp [halfInt, zsqrtToQsqrt] using congrArg QuadraticAlgebra.re
49     ↪ hz.symm
49   have hn : (b' : ℚ) / 2 = z.im := by
50     simp [halfInt, zsqrtToQsqrt] using congrArg QuadraticAlgebra.im
51     ↪ hz.symm
51   have ha : 2 ∣ a' := by
52     refine ⟨z.re, ?_⟩
53     have hq : (a' : ℚ) = 2 * z.re := by nlinarith [hm]
54     exact_mod_cast hq
55   have hb : 2 ∣ b' := by
56     refine ⟨z.im, ?_⟩
57     have hq : (b' : ℚ) = 2 * z.im := by nlinarith [hn]
58     exact_mod_cast hq
59   exact ⟨ha, hb⟩
60   · rintro ⟨ha, hb⟩
61   rcases ha with ⟨m, hm⟩
62   rcases hb with ⟨n, hn⟩

```

**7.5. Theorem 7.5 (classification in  $d \not\equiv 1 \pmod{4}$  branch).** Lean name: `dvd_four_sub_sq_iff_exists_zsqrtImage_of_ne_one_mod_four`.  
 If  $d$  is squarefree and  $d \not\equiv 1 \pmod{4}$ , then

$$4 \mid (a'^2 - db'^2) \iff \exists z \in \mathbb{Z}[\sqrt{d}], \iota_d(z) = \frac{a' + b'\sqrt{d}}{2}.$$

Combining the mod-4 criterion (Theorem 6.8) with the image criterion (Theorem 7.4), this establishes that an integral element in half-integral form lies in  $\mathbb{Z}[\sqrt{d}]$  precisely when  $d \not\equiv 1 \pmod{4}$ . This completes the  $d \not\equiv 1 \pmod{4}$  branch of the classification at the element-level.

```

68   theorem dvd_four_sub_sq_iff_exists_zsqrtImage_of_ne_one_mod_four
69     (d a' b' : ℤ) (hd : Squarefree d) (hd4 : d % 4 ≠ 1) :
70     4 ∣ (a' ^ 2 - d * b' ^ 2) ↔ ∃ z : ℤ√d, zsqrtToQsqrt d z = halfInt d
71     ↪ a' b' := by
72   rw [dvd_four_sub_sq_iff_even_even_of_ne_one_mod_four d a' b' hd hd4]
73   rw [halfInt_mem_range_zsqrtToQsqrt_iff_even_even]

```

## 8. Progress and remaining work

**Current progress.** The formalization now fully covers:

- The quadratic field setup and basic algebraic infrastructure (section 5 and preceding sections).
- The complete parity/divisibility mechanism reducing integrality to congruence conditions (section 6).
- The embedding and image characterization of  $\mathbb{Z}[\sqrt{d}]$  inside  $\mathbb{Q}(\sqrt{d})$ , completing the element-level classification for the  $d \not\equiv 1 \pmod{4}$  branch.
- The non-isomorphism theorem for distinct quadratic fields.



**Remaining (not yet formalized).** The explicit open item is the  $d \equiv 1 \pmod{4}$  structural branch, expected to use  $\mathbb{Z}[\frac{1+\sqrt{d}}{2}]$  as the integral model. Concretely, the next milestones are:

1. Formalize the missing  $\mathbb{Z}[\frac{1+\sqrt{1+4k}}{2}]$ -style construction needed for the  $1 \pmod{4}$  branch.
2. Complete the two remaining sorry markers in `ClassificationToZsqrtd.lean`, linking the element-level classification to the ring-of-integers isomorphism.
3. Unify both congruence branches into a final ring-of-integers classification theorem with a single API-level statement.

## References

- [1] George Boxer. *Algebraic Number Theory: Lecture Notes*. Imperial College London. 2024.
- [2] Mathlib Community. *Mathlib4: The Math Library for Lean 4*. 2024. URL: <https://github.com/leanprover-community/mathlib4>.