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

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 $d \bmod 4$. The central goal is to formalize the standard result (cf. [1, Lecture 2]):

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

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`[1] (the forward and reverse directions linking the ring-of-integers isomorphism to the mod-4 condition); all other files are sorry-free.

[1] 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

$$\texttt{Qsqrtd}(d) := \texttt{QuadraticAlgebra}\ \mathbb{Q}\ (d : \mathbb{Q})\ 0.$$

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

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

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]).

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

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

The type

$$\texttt{Qsqrtd}(d) := \texttt{QuadraticAlgebra}\ \mathbb{Q}\ (d : \mathbb{Q})\ 0$$

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

```
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

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

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

```
def rescale (d : ℚ) (a : ℚ) (ha : a ≠ 0) :
    QuadraticAlgebra ℚ d 0 ≃ₐ[ℚ] QuadraticAlgebra ℚ (a ^ 2 * d) 0 := by
  have h1d : (1 : QuadraticAlgebra ℚ d 0) = ⟨1, 0⟩ := by ext <;> rfl
  have h1a : (1 : QuadraticAlgebra ℚ (a ^ 2 * d) 0) = ⟨1, 0⟩ := by
    ext <;> rfl
  exact AlgEquiv.ofLinearEquiv
    { toFun := fun x => ⟨x.re, x.im * a⁻¹⟩
      invFun := fun y => ⟨y.re, y.im * a⟩
      map_add' := by intro x y; ext <;> simp [add_mul]
      map_smul' := by intro c x; ext <;> simp [mul_assoc]
      left_inv := by
        intro x; ext <;> simp [mul_assoc, inv_mul_cancel₀ ha]
      right_inv := by
        intro y; ext <;> simp [mul_assoc, mul_inv_cancel₀ ha] }
    (by simp [h1d, h1a])
    (by intro x y; ext <;> simp <;> field_simp)
```

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

For $x \in \texttt{Qsqrtd}(d)$, define

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

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

```
abbrev trace {d : ℤ} (x : Qsqrtd d) : ℚ := x.re + (star x).re

/-- Norm on `Qsqrtd d`. -/
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.

```
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

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

```
class IsNonsquareRat (d : ℤ) : Prop where
  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,

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

This excludes degeneration of the quadratic extension.

```
lemma not_isSquare_int (d : ℤ) [IsQuadraticParam d] : ¬ IsSquare d := by
  intro hdSq
  rcases hdSq with ⟨z, hz⟩
  by_cases huz : IsUnit z
  · rcases Int.isUnit_iff.mp huz with hz1 | hz1
    · have : d = 1 := by simpa [hz1] using hz
      exact (IsQuadraticParam.ne_one (d := d)) this
    · have : d = 1 := by simpa [hz1] using hz
      exact (IsQuadraticParam.ne_one (d := d)) this
  · have hsqz2 : Squarefree (z ^ 2) := by
      simpa [hz, pow_two] using (IsQuadraticParam.squarefree (d := d))
    have h01 : (2 : ℕ) = 0 ∨ (2 : ℕ) = 1 :=
      Squarefree.eq_zero_or_one_of_pow_of_not_isUnit (x := z) (n := 2)
        ↪  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

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

```
instance (d : ℤ) [IsQuadraticParam d] : IsNonsquareRat d := by
  refine ⟨?_⟩
  intro r hr
  have hsqQ : IsSquare ((d : ℤ) : ℚ) := ⟨r, by simpa [pow_two] using
   ↪  hr.symm⟩
  have hsqZ : IsSquare d := (Rat.isSquare_intCast_iff).1 hsqQ
  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.

```
instance {d : ℤ} [IsNonsquareRat d] : Field (Qsqrtd d) := by
  letI : Fact (∀ r : ℚ, r ^ 2 ≠ (d : ℚ) + 0 * r) := ⟨by
    intro r hr
    exact (IsNonsquareRat.nonsquare (d := d) r) (by simp [hr])
  ⟩
  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 $x = a + b\sqrt{d} \in \mathbb{Q}(\sqrt{d})$,

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

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

3.2. Theorem 3.2 (norm formula). Lean name: `norm_eq_sqr_minus_d_sqr`.

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

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

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

```

3.3. Theorem 3.3 (quadratic polynomial annihilation). Lean name: `aeval_eq_zero_of_quadratic`.

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

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

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

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

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

```
isomorphic (as `ℚ`-algebras) whenever `d₁ ≠ d₂` and both are squarefree
nonzero integers.

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

$$m = 1.$$

```
3. Expanding in the basis `{1, √d₂}`, the imaginary part equation forces
   `x.re * x.im = 0`, giving two cases.
4. In either case we derive that `d₁/d₂` is a square in `ℚ`.
5. A denominator argument using squarefreeness shows `d₂ ∣ d₁` and `d₁ ∣
↪  d₂`,
   so `d₁` and `d₂` are associates in `ℤ`.
6. Associates in `ℤ` means `d₁ = d₂` or `d₁ = -d₂`; the former contradicts
   `d₁ ≠ d₂`, and the latter implies `-1` is a square in `ℚ`, a
    ↪  contradiction.
-/

namespace ClassificationOfIntegersOfQuadraticNumberFields
namespace Qsqrtd

/-- `-1` is not a square in `ℚ`. Any rational square `r * r` is nonneg, but
↪  `-1 < 0`. -/
lemma not_isSquare_neg_one_rat : ¬ IsSquare (- (1 : ℚ)) := by
  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.

```

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

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

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.

```
    -- (squarefree means every factor appears at most once, but z appears twice).
    have hsqz2 : Squarefree (z ^ 2) := by simpa [hz, pow_two] using hsm
    have h01 : (2 : ℕ) = 0 ∨ (2 : ℕ) = 1 :=
      Squarefree.eq_zero_or_one_of_pow_of_not_isUnit (x := z) (n := 2) hsqz2 huz
    norm_num at h01

/-- If `d₂` is squarefree and nonzero, and `d₁ / d₂` is a square in `ℚ`, then `d₂ ∣ d₁`.

    The key idea: write `d₁/d₂` in lowest terms as `p/q`. The denominator `q` divides `d₂`,
    so `q` is also squarefree. But `d₁/d₂` being a square forces `q` to be a square as well.
    A squarefree square must be 1, meaning `d₂` divides `d₁`. -/
lemma int_dvd_of_ratio_square (d₁ d₂ : ℤ) (hd₂ : d₂ ≠ 0)
    (hsq_d₂ : Squarefree d₂) (hr : IsSquare ((d₁ : ℚ) / (d₂ : ℚ))) : d₂ ∣ d₁ := by
  -- The denominator of d₁/d₂ (in lowest terms) is a square.
  have hsq_den_nat : IsSquare (((d₁ : ℚ) / (d₂ : ℚ)).den) := (Rat.isSquare_iff.mp hr).2
  -- Lift the squareness to ℤ.
  have hsq_den_int : IsSquare ((((d₁ : ℚ) / (d₂ : ℚ)).den : ℤ)) := by
    rcases hsq_den_nat with ⟨n, hn⟩
    refine ⟨(n : ℤ), by exact_mod_cast hn⟩
  -- The denominator of d₁/d₂ divides d₂.
  have hden_dvd : ((((d₁ : ℚ) / (d₂ : ℚ)).den : ℤ)) ∣ d₂ := by
    simpa [← Rat.divInt_eq_div] using (Rat.den_dvd d₁ d₂)
  -- Since it divides the squarefree d₂, the denominator is itself squarefree.
  have hsqf_den_int : Squarefree ((((d₁ : ℚ) / (d₂ : ℚ)).den : ℤ)) :=
    Squarefree.squarefree_of_dvd hden_dvd hsq_d₂
  -- Squarefree + square □ denominator = 1, so d₂ ∣ d₁.
  have hden1_nat : ((d₁ : ℚ) / (d₂ : ℚ)).den = 1 :=
    nat_eq_one_of_squarefree_intcast_of_isSquare _ hsqf_den_int hsq_den_int
  exact (Rat.den_div_intCast_eq_one_iff d₁ d₂ hd₂).1 hden1_nat
```

```lean

/-- **Main theorem**: Distinct squarefree parameters define non-isomorphic quadratic fields.

    If `d₁ ≠ d₂` are both squarefree nonzero integers, then there is no `ℚ`-algebra
    isomorphism `ℚ(√d₁) ≃ₐ[ℚ] ℚ(√d₂)`. -/
theorem quadratic_fields_not_iso
    (d1 d2 : ℤ) [IsQuadraticParam d1] [IsQuadraticParam d2]
    (hneq : d1 ≠ d2) :
    ¬ Nonempty (Qsqrtd d1 ≃ₐ[ℚ] Qsqrtd d2) := by
  rintro ⟨e⟩
  -- Let x = e(√d₁) ∈ ℚ(√d₂). Write x = a + b√d₂ for some a, b ∈ ℚ.
  let x : Qsqrtd d2 := e ⟨0, 1⟩
  -- Since e is a ring homomorphism and (√d₁)² = d₁, we get x² = d₁ in ℚ(√d₂).
  have hx : x * x = (d1 : Qsqrtd d2) := by
    change e ⟨0, 1⟩ * e ⟨0, 1⟩ = (d1 : Qsqrtd d2)
    calc
      e ⟨0, 1⟩ * e ⟨0, 1⟩ = e ((⟨0, 1⟩ : Qsqrtd d1) * ⟨0, 1⟩) := by
        symm
        exact e.map_mul _ _
      _ = e (d1 : Qsqrtd d1) := by
        congr 1
        ext <;> simp [Qsqrtd]
      _ = (d1 : Qsqrtd d2) := by simp
  -- Expanding x² = (a + b√d₂)² = (a² + b²d₂) + 2ab√d₂, the imaginary part gives 2ab = 0.
  have him0 : (x * x).im = 0 := by
    have him := congrArg QuadraticAlgebra.im hx
    simpa [Qsqrtd] using him
  have hsum : x.re * x.im + x.im * x.re = 0 := by
    simpa [Qsqrtd, mul_assoc, mul_comm, mul_left_comm] using him0
  -- So either a = 0 or b = 0.
  have hprod : x.re * x.im = 0 := by nlinarith [hsum]
  -- In either case, d₁/d₂ is a perfect square in ℚ.
  have hratio : IsSquare ((d1 : ℚ) / (d2 : ℚ)) := by
    rcases mul_eq_zero.mp hprod with hre | him
    · -- Case a = 0: then x = b√d₂, so x² = b²d₂ = d₁, giving d₁/d₂ = b².
      refine ⟨x.im, ?_⟩
      have hre0 : (x * x).re = d1 := by
        have hre' := congrArg QuadraticAlgebra.re hx
        simpa [Qsqrtd] using hre'
      have hmain : (d2 : ℚ) * (x.im ^ 2) = d1 := by
        simpa [Qsqrtd, hre, pow_two, mul_assoc, mul_comm, mul_left_comm] using hre0
      have hd2Q : (d2 : ℚ) ≠ 0 := by
        exact_mod_cast (IsQuadraticParam.ne_zero (d := d2))
```

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

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

```
def halfInt (d : ℤ) (a' b' : ℤ) : Qsqrtd d :=
  ⟨a' / 2, b' / 2⟩
```

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

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

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

```

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 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.

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

```

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.

```
lemma squarefree_int_not_dvd_four (d : ℤ) (hd : Squarefree d) : ¬ (4 : ℤ) ∣ d := by
  intro h
  have h22 : (2 : ℤ) * 2 ∣ d := by
    obtain ⟨k, hk⟩ := h
    exact ⟨k, by omega⟩
  have hunit : IsUnit (2 : ℤ) := hd 2 h22
  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.

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

6.3. Lemma 6.3. Lean name: `Int.sq_emod_four_of_even`.
If $2 \mid n$, then

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

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

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

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

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

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

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

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 $d \equiv 1 \pmod 4$.

```
theorem dvd_four_sub_sq_iff_even_even_or_odd_odd_mod_four_one (d a' b' : ℤ)
↪  (hd : Squarefree d) :
    4 ∣ (a' ^ 2 - d * b' ^ 2) ↔
      (2 ∣ a' ∧ 2 ∣ b') ∨ (¬ 2 ∣ a' ∧ ¬ 2 ∣ b' ∧ d % 4 = 1) := by
  have hd4 := squarefree_int_emod_four d hd
  constructor
  · intro hdvd
    have hmod : (a' ^ 2 - d * b' ^ 2) % 4 = 0 := (div4_iff_mod a' b' d).1
    ↪  hdvd
    have even_odd_impossible (ha : 2 ∣ a') (hb : ¬ 2 ∣ b') : False := by
      have hmod' := hmod
      have ha_eq : a' ^ 2 = 4 * (a' ^ 2 / 4) := by
        have ha2 : a' ^ 2 % 4 = 0 := Int.sq_emod_four_of_even a' ha
        omega
      have hb_eq : b' ^ 2 = 4 * (b' ^ 2 / 4) + 1 := by
        have hb2 : b' ^ 2 % 4 = 1 := Int.sq_emod_four_of_odd b' hb
        omega
      rw [hb_eq] at hmod'
      ring_nf at hmod'
      rcases hd4 with hd1 | hd2 | hd3 <;> omega
    have odd_even_impossible (ha : ¬ 2 ∣ a') (hb : 2 ∣ b') : False := by
      have hmod' := hmod
      have ha_eq : a' ^ 2 = 4 * (a' ^ 2 / 4) + 1 := by
        have ha2 : a' ^ 2 % 4 = 1 := Int.sq_emod_four_of_odd a' ha
        omega
      have hb_eq : b' ^ 2 = 4 * (b' ^ 2 / 4) := by
        have hb2 : b' ^ 2 % 4 = 0 := Int.sq_emod_four_of_even b' hb
        omega
      rw [ha_eq, hb_eq] at hmod'
      ring_nf at hmod'
      rcases hd4 with hd1 | hd2 | hd3 <;> omega
    have odd_odd_mod_four_one (ha : ¬ 2 ∣ a') (hb : ¬ 2 ∣ b') : d % 4 = 1
    ↪  := by
      have hmod' := hmod
      have ha_eq : a' ^ 2 = 4 * (a' ^ 2 / 4) + 1 := by
```

```
        have ha2 : a' ^ 2 % 4 = 1 := Int.sq_emod_four_of_odd a' ha
        omega
      have hb_eq : b' ^ 2 = 4 * (b' ^ 2 / 4) + 1 := by
        have hb2 : b' ^ 2 % 4 = 1 := Int.sq_emod_four_of_odd b' hb
        omega
      rw [ha_eq, hb_eq] at hmod'
      ring_nf at hmod'
      omega
    by_cases ha : 2 ∣ a' <;> by_cases hb : 2 ∣ b'
    · exact Or.inl ⟨ha, hb⟩
    · exfalso
      exact even_odd_impossible ha hb
    · exfalso
      exact odd_even_impossible ha hb
    · exact Or.inr ⟨ha, hb, odd_odd_mod_four_one ha hb⟩
  · intro h
    rcases h with ⟨ha, hb⟩ | ⟨ha, hb, hd1⟩
    · obtain ⟨p, rfl⟩ := ha
      obtain ⟨q, rfl⟩ := hb
      exact ⟨p ^ 2 - d * q ^ 2, by ring⟩
    · have ha_eq : a' = 2 * (a' / 2) + 1 := by omega
      have hb_eq : b' = 2 * (b' / 2) + 1 := by omega
      rw [ha_eq, hb_eq]
      ring_nf
      have hd_eq : d = 4 * (d / 4) + 1 := by omega
      rw [hd_eq]
      ring_nf
      omega
```

6.7. Theorem 6.7 (forcing even–even when $d \not\equiv 1 \bmod 4$). 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

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

```
theorem even_even_of_dvd_four_sub_sq_of_ne_one_mod_four (d a' b' : ℤ) (hd : Squarefree d)
    (hd4 : d % 4 ≠ 1) (h : 4 ∣ (a' ^ 2 - d * b' ^ 2)) :
    2 ∣ a' ∧ 2 ∣ b' := by
  rcases (dvd_four_sub_sq_iff_even_even_or_odd_odd_mod_four_one d a' b' hd).mp h with
      hab | ⟨_, _, hd1⟩
  · exact hab
  · 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

$$4 \mid (a'^2 - db'^2) \iff \big(2 \mid a' \;\&\; 2 \mid b'\big).$$

```
theorem dvd_four_sub_sq_iff_even_even_of_ne_one_mod_four (d a' b' : ℤ) (hd : Squarefree d)
    (hd4 : d % 4 ≠ 1) :
```

```
    4 ∣ (a' ^ 2 - d * b' ^ 2) ↔ (2 ∣ a' ∧ 2 ∣ b') := by
  constructor
  · intro h
    exact even_even_of_dvd_four_sub_sq_of_ne_one_mod_four d a' b' hd hd4 h
  · intro h
    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.

```
/-- If `d % 4 = 1`, divisibility by `4` is equivalent to same parity. -/
theorem dvd_four_sub_sq_iff_same_parity_of_one_mod_four (d a' b' : ℤ) (hd : Squarefree d)
    (hd4 : d % 4 = 1) :
    4 ∣ (a' ^ 2 - d * b' ^ 2) ↔ a' % 2 = b' % 2 := by
  rw [dvd_four_sub_sq_iff_even_even_or_odd_odd_mod_four_one d a' b' hd]
  constructor
  · rintro (⟨ha, hb⟩ | ⟨ha, hb, _⟩) <;> omega
  · intro h
    by_cases ha : 2 ∣ a'
    · left
      exact ⟨ha, by omega⟩
    · right
      exact ⟨ha, by omega, hd4⟩

```

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

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

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

```

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

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

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

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

```

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

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

There is a ring isomorphism

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

```

/-- `ℤ√d` is ring-isomorphic to its image in `ℚ(√d)`. -/
noncomputable def zsqrtdEquivRangeQsqrtd (d : ℤ) : ℤ√d ≃+* (zsqrtdToQsqrtd d).range := by
  refine RingEquiv.ofBijective (zsqrtdToQsqrtd d).rangeRestrict ?_
  constructor
  · intro z w hzw
    exact zsqrtdToQsqrtd_injective d (Subtype.ext_iff.mp hzw)
  · intro x
    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',

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

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

```

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

```
    have hm : (a' : ℚ) / 2 = z.re := by
      simpa [halfInt, zsqrtdToQsqrtd] using congrArg QuadraticAlgebra.re hz.symm
    have hn : (b' : ℚ) / 2 = z.im := by
      simpa [halfInt, zsqrtdToQsqrtd] using congrArg QuadraticAlgebra.im hz.symm
    have ha : 2 ∣ a' := by
      refine ⟨z.re, ?_⟩
      have hq : (a' : ℚ) = 2 * z.re := by nlinarith [hm]
      exact_mod_cast hq
    have hb : 2 ∣ b' := by
      refine ⟨z.im, ?_⟩
      have hq : (b' : ℚ) = 2 * z.im := by nlinarith [hn]
      exact_mod_cast hq
    exact ⟨ha, hb⟩
  · rintro ⟨ha, hb⟩
    rcases ha with ⟨m, hm⟩
    rcases hb with ⟨n, hn⟩
```

7.5. Theorem 7.5 (classification in $d \not\equiv 1 \bmod 4$ branch). Lean name: `dvd_four_sub_sq_iff_exists_zsqrtd_image_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 $d \not\equiv 1$ branch of the classification at the element-level.

```
theorem dvd_four_sub_sq_iff_exists_zsqrtd_image_of_ne_one_mod_four
    (d a' b' : ℤ) (hd : Squarefree d) (hd4 : d % 4 ≠ 1) :
    4 ∣ (a' ^ 2 - d * b' ^ 2) ↔ ∃ z : ℤ√d, zsqrtdToQsqrtd d z = halfInt d a' b' := by
  rw [dvd_four_sub_sq_iff_even_even_of_ne_one_mod_four d a' b' hd hd4]
  rw [halfInt_mem_range_zsqrtdToQsqrtd_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 $\mathbb{Z}\left[\frac{1+\sqrt{1+4k}}{2}\right]$-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`.