A formalization of Dedekind domains and class groups of global fields

Anne Baanen
Vrije Universiteit Amsterdam, Netherlands

Sander R. Dahmen
Vrije Universiteit Amsterdam, Netherlands

Ashvni Narayanan
London School of Geometry and Number Theory

Filippo A. E. Nuccio Mortarino Majno di Capriglio
Univ Lyon, Université Jean Monnet Saint-Étienne, CNRS UMR 5208, Institut Camille Jordan, F-42023 Saint-Étienne, France

Abstract

Dedekind domains and their class groups are notions in commutative algebra that are essential in algebraic number theory. We formalized these structures and several fundamental properties, including number theoretic finiteness results for class groups, in the Lean prover as part of the mathlib mathematical library. This paper describes the formalization process, noting the idioms we found useful in our development and mathlib's decentralized collaboration processes involved in this project.

2012 ACM Subject Classification Mathematics of computing → Mathematical software; Security and privacy → Logic and verification

Keywords and phrases formal math, algebraic number theory, commutative algebra, Lean, mathlib

Digital Object Identifier 10.4230/LIPIcs.CVIT.2016.23

Supplementary Material Full source code of the formalization is part of mathlib. Copies of the source files relevant to this paper are available in a separate repository.
Software: `https://github.com/lean-forward/class-number`
archived at `Software Heritage Identifier`

Funding *Anne Baanen*: NWO Vidi grant No. 016.Vidi.189.037, Lean Forward
Sander R. Dahmen: NWO Vidi grant No. 639.032.613, New Diophantine Directions
Ashvni Narayanan: EPSRC, UK

Acknowledgements I want to thank ...

1 Introduction

In its basic form, number theory studies properties of the integers $\mathbb{Z}$ and its fraction field, the rational numbers $\mathbb{Q}$.[1] Both for the sake of generalization, as well as for providing powerful techniques to answer questions about the original objects $\mathbb{Z}$ and $\mathbb{Q}$, it is worthwhile to study finite field extensions of $\mathbb{Q}$, called *number fields*, as well as their so called *rings of integers* (defined in Section 2 below), whose relations mirror the way $\mathbb{Q}$ contains $\mathbb{Z}$ as a subring. These number fields and their rings of integers form the basic objects of study of algebraic number theory, an important brach of modern number theory. In this paper, we describe our project

[1] From a classical point of view, one could even argue that the positive, or perhaps nonnegative, integers and rational numbers are the most basic objects of study of number theory. From an algebraic point of view, this would still quickly lead into studying $\mathbb{Z}$ and $\mathbb{Q}$.

aiming to formalize these notions and some of their important properties. Our goal, however, is not to get to the definitions and properties as quickly as possible, but instead we aim at our formalization as a foundation for future work, as part of a natural and more general theory as we shall explain below.

In particular, our project resulted in formalized definitions and elementary properties of number fields and their rings of integers (described in Section 3.3), Dedekind domains (Section 4), and the ideal class group and class number (Section 7). The main proofs that we formalized show that two definitions of Dedekind domains are equivalent (Section 4.3), that the ring of integers (or more generally: the integral closure of a Dedekind domain in a finite separable field extension) is a Dedekind domain (Section 6) and that the class group of a number field is finite (Section 7).

Apart from the achievement of formalizing a non-trivial amount of mathematical theory, our formal definition of the class number is an essential requirement for the use of theorem provers in modern number theory research.

Our work is developed as part of the mathematical library mathlib [17] for the Lean 3 theorem prover [5]. The formal system of Lean is a dependent type theory based on the calculus of inductive constructions, with a proof-irrelevant impredicative universe `Prop` at the bottom of a noncumulative hierarchy of universes `Prop : Type : Type 1 : Type 2 : ...`.[2] Other important characteristics of Lean as used in mathlib are the use of quotient types, ubiquitous classical reasoning and the use of typeclasses to define the hierarchy of algebraic structures.

Organizationally, mathlib is characterized by a distributed and decentralized community of contributors, a willingness to refactor its basic definitions, and a preference for small yet complete contributions over larger projects added all at once. Our own project, being part of the development of mathlib, follows this philosophy by contributing pieces of our work as they are finished, in turn taking advantage of results contributed by others after the start of the project. At several points, we had just merged a formalization into mathlib that another contributor needed, immediately before they contributed a result that we needed. Due to the decentralized organization and fluid nature of contributions to mathlib, its contents are built up of many different contributions from many different authors. Attributing each formalization to a single set of main authors would not do justice to all others whose additions and tweaks are essential to its current use. Therefore, we will make clear whether a contribution is part of our project or not, but not who we consider to be the main author(s).

The source files of the formalization are currently in the process of being merged into mathlib, an up-to-date branch being available `https://github.com/leanprover-community/mathlib/tree/dedekind-domain-dev`. We also maintain a separate repository containing the files relevant to this paper, available at `https://github.com/lean-forward/class-number`.

2 Mathematical background

Let us now introduce some of the main objects we study, described in a "standard" mathematical way. In the later sections we will detail their formalization in Lean.

A *number field* K is a finite field extension of $\mathbb{Q}$, and as such has the structure of a finite dimensional vector space over $\mathbb{Q}$. The smallest example is $\mathbb{Q}$ itself, and the two-dimensional

[2] In our code samples, we use `Type*` as abbreviation of "`Type u` for an arbitrary choice of `u`".

cases are given by the quadratic number fields

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

where $d \neq 1$ is a squarefree (i.e. not divisible by p^2 for any prime p) integer. A cubic example is

$$K := \{a + b\alpha + c\alpha^2 : a, b, c \in \mathbb{Q}\}$$

where α satisfies $\alpha^3 + \alpha^2 - 2\alpha + 8 = 0$ (it is the unique real number with this property).

The *ring of integers* $\mathcal{O}_K$ of a number field K is defined as the integral closure of $\mathbb{Z}$ in K, which boils down to

$$\mathcal{O}_K := \{x \in K : f(x) = 0 \text{ for some } \textit{monic} \text{ polynomial } f \text{ with integer coefficients}\},$$

where we recall that a polynomial is called *monic* if its leading coefficient equals 1. While it might not be immediately obvious that $\mathcal{O}_K$ forms a ring, this follows form general algebraic properties of integral closures. Some examples of $\mathcal{O}_K$ are as follows. Taking $K = \mathbb{Q}$, we get $\mathcal{O}_K = \mathbb{Z}$ back. For $K = \mathbb{Q}(\sqrt{2})$ we get $\mathcal{O}_K = \mathbb{Z}[\sqrt{2}] = \{a + b\sqrt{2} : a, b \in \mathbb{Z}\}$. But for $K = \mathbb{Q}(\sqrt{5})$ we do *not* simply get $\mathbb{Z}[\sqrt{5}] = \{a + b\sqrt{5} : a, b \in \mathbb{Z}\}$ as $\mathcal{O}_K$, since the golden ratio $\varphi := (1 + \sqrt{5})/2 \notin \mathbb{Z}[\sqrt{5}]$ satisfies the monic polynomial equation $\varphi^2 - \varphi - 1 = 0$, hence by definition $\varphi \in \mathcal{O}_K$; it turns out that $\mathcal{O}_K = \mathbb{Z}[\varphi] = \{a + b\varphi : a, b \in \mathbb{Z}\}$. For quadratic numbers field $\mathbb{Q}(\sqrt{d})$, with d as above, the previous two examples in fact generalize to

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

Finally, if $K = \mathbb{Q}(\alpha)$ with α as before, then $\mathcal{O}_K = \{a + b\alpha + c(\alpha + \alpha^2)/2 : a, b, c \in \mathbb{Z}\}$, illustrating that explicitly writing down $\mathcal{O}_K$ can quickly become complicated. We could think of $\mathcal{O}_K$ as generalization of $\mathbb{Z}$, and ask for properties of $\mathbb{Z}$ if they still hold in $\mathcal{O}_K$, and if not, if a natural generalization still holds. An important property of $\mathbb{Z}$ is that it is a PID (i.e. a principal ideal domain), and hence a UFD (i.e. unique factorization domain); the latter meaning that every nonzero nonunit element can be written as a (nonempty) finite product of irreducible elements, which is unique up to the order of the elements and changing the elements by multiplication with units (which are ± 1 in $\mathbb{Z}$). For example, 6 can be factorized in exactly 4 ways, namely $6 = 2 \cdot 3 = 3 \cdot 2 = (-2) \cdot (-3) = (-3) \cdot (-2)$. Some well known rings of integers are e.g. the Gaussian integers $\mathbb{Z}[i] = \{a + bi : a, b, \in \mathbb{Z}\}$ (with i some squareroot of -1), the Eisenstein integers $\mathbb{Z}[(1 + \sqrt{-3})/2]$, and the 'real' quadratic ring $\mathbb{Z}[\sqrt{2}]$. They all have in common that they are also UFD's, like $\mathbb{Z}$. However, this is certainly not true for all rings of integers. For example $\mathbb{Z}[\sqrt{-5}]$ is *not* a UFD: $6 = 2 \cdot 3 = (1 + \sqrt{-5})(1 - \sqrt{-5})$ provide two essentially different ways to factor 6 in irredicible elements in $\mathbb{Z}[\sqrt{-5}]$. As it turns out, there is a beautiful way to remedy this. Namely by considering factorization of *ideals* instead of elements: for a number field K, with ring of integers $\mathcal{O}_K$, every nonzero ideal of $\mathcal{O}_K$ can be factored into prime ideals in a unique way, up to the order of the factors.

Although unique factorization in terms of ideals is of great importance and beauty, it is still very interesting, and for many arithmetic applications necessary, to also consider factorization properties in terms of elements. For this, one can consider the nonzero fractional ideals of $\mathcal{O}_K$ modulo the units K^*, which a priori has the structure of a commutative monoid, but actually turns out to be a group, called the *class group* of $\mathcal{O}_K$. An important theorem is that the class group of $\mathcal{O}_K$ is finite. The interpretation is that the order of this group, called

the *class number*, measures how far away $\mathcal{O}_K$ is from being a UFD. In particular, the class group of $\mathcal{O}_K$ is trivial if and only if $\mathcal{O}_K$ is a UFD.

Now talk about the more general setting and describe more explicitly the different theorems concerning Dedekind domains, class groups, etc.

The intrinsic algebraic properties of $\mathcal{O}_K$ are very nice. In particular, every ring of integers $\mathcal{O}_K$ is a *Dedekind domain*. The latter can be defined as a domain D which is Noetherian (i.e. every ideal of D is finitely generated), integrally closed (i.e. if x is in the fraction field of D and a root of a monic polynonial with coefficients in D, then actually $x \in D$), and of Krull dimension at most 1 (i.e. every nonzero prime ideal of D is maximal).

3 Number fields, global fields and rings of integers

A number field is a finite field extension of $\mathbb{Q}$. Number fields are a basic concept in algebraic number theory. Examples: .... We formalized number fields as the following typeclass:

```
class is_number_field (K : Type*) [field K] :=
[cz : char_zero K] [fd : finite_dimensional ℚ K]
```

The condition `[cz : char_zero K]` states that K has characteristic zero, i.e. the canonical ring homomorphism $\mathbb{Z} \to K$ is an embedding. This implies that there is a $\mathbb{Q}$-algebra structure on K (found by typeclass search), this gives the vector space structure used in the `[fd : finite_dimensional ℚ K]` hypothesis.

3.1 Field extensions

The definition of `is_number_field` illustrates our treatment of field extensions. In informal mathematics, a field L containing a subfield K is said to be a field extension L/K. Often we encounter towers of field extensions: we might have that $\mathbb{Q}$ is contained in K, K is contained in L, L is contained in an algebraic closure $\bar{K}$ of K, and $\bar{K}$ is contained in $\mathbb{C}$. We might formalize this situation by viewing $\mathbb{Q}$, K, L and $\bar{K}$ to be sets of complex numbers $\mathbb{C}$ and defining field extensions as subset relations between these subfields. This way, no coercions need to be inserted to map elements of one field into a larger field. In type theory we cannot define $\mathbb{Q}$ as a subset of $\mathbb{C}$ since we need $\mathbb{Q}$ to define $\mathbb{C}$. Thus, some coercion is always needed to go from the original definition of $\mathbb{Q}$ to its image in $\mathbb{C}$; and similar issues arise for other subfields that were not originally defined as such. Moreover, such an approach loses flexibility since we need to fix the top field, of which all others are subfields, at the start of our development and cannot adjoin more elements when needed.

Instead, we formalize results about field extensions by parametrization. The lemma statement is parametrized over abritrary types K and L with a field structure, along with the hypothesis "L is a field extension of K", represented by an instance parameter `[algebra K L]`. This provides us with a canonical ring homomorphism `algebra_map K L` : $K \to L$; this map is injective because K and L are fields. In other words, field extensions are given by their canonical embeddings.

3.2 Scalar towers

The main drawback of using arbitrary embeddings to represent field extensions is that we need to prove that these maps commute. For example, we might start with a field extension $L/\mathbb{Q}$, then define a subfield K of L, resulting in a tower of extensions $L/K/\mathbb{Q}$. In such a tower, the map $\mathbb{Q} \to L$ should be equal to the composition $\mathbb{Q} \to K \to L$. The example has

other maps depend on the map $\mathbb{Q} \to L$, so we cannot arrange the coherence condition by defining $\mathbb{Q} \to L$ after the fact.

The solution in mathlib is to parametrize over all three maps, as long a there is also a proof of coherency: a hypothesis of the form "$L/K/F$ is a tower of field extensions" is translated to three instance parameters `[algebra F K]`, `[algebra K L]` and `[algebra F L]`, along with an additional parameter `[is_scalar_tower F K L]` expressing that the maps commute.

The `is_scalar_tower` typeclass derives its name from its applicability to any three types between which exist scalar multiplication operations:

```
class is_scalar_tower (M N α : Type*) [has_scalar M N] [has_scalar N α]
  [has_scalar M α] : Prop :=
(smul_assoc : ∀ (x : M) (y : N) (z : α), (x • y) • z = x • (y • z))
```

For example, if R is a ring, A is an R-algebra and M an A-module, we can express the fact that M is also an R-module by adding a `[is_scalar_tower R A M]` parameter. Since `x • y` for an R-algebra A is defined as `algebra_map R A x * y`, applying `smul_assoc` for each x with $y = z = 1$ shows that the `algebra_map` s indeed commute.

The typeclass system is set up to automatically provide common `is_scalar_tower` instances, such as for the maps $R \to S \to A$ when S is a R-subalgebra of S. The effect is that almost all coherence proof obligations are automatically solved from known results or filled in from parameters. In our formalization, we found that the `is_scalar_tower` typeclass translates towers of field extension well.

3.3 Ring of integers

A number ring is defined as a ring whose fraction field is a number field, the ring of integers $\mathcal{O}_K$ is an important example. The ring of integers in K is defined as the integral closure of $\mathbb{Z}$ in K. This is the subring containing those $x : K$ that are the root of a monic polynomials with coefficients in $\mathbb{Z}$:

```
def ring_of_integers (K : Type*) [field K] [is_number_field K] :
  subalgebra ℤ K :=
integral_closure ℤ K
```

where `integral_closure` was previously defined in mathlib as follows:

```
def integral_closure (R A : Type*) [comm_ring R] [comm_ring A]
  [algebra R A] : subalgebra R A :=
{ carrier := { r | is_integral R r },
  .. /- proofs omitted -/ }
```

Some examples of rings of integers include $\mathbb{Z}$ and $\mathbb{Z}[\iota]$. We prove ahead that the ring of integers of a number field is, in fact, a Dedekind domain. Moreover, it is a finitely-generated free $\mathbb{Z}$-module, with rank equal to the degree of the number field over $\mathbb{Q}$.

3.4 Subobjects

The ring of integers are one example of a subobject, such as a subfield, subring or subalgebra, defined through a characteristic predicate. In mathlib, a subobject is defined as a bundled structure comprising the carrier set, along with proofs showing the carrier set is closed under the relevant operations.

Two new subobjects we needed in our development were `subfield` and `intermediate_field`. We define a subfield of a field K as a subset of K that contains 0 and 1 and is closed under addition, negation, multiplication, and taking inverses. If L is a field extension of K, we define an intermediate field as a subfield that is also a subalgebra: a subfield that contains the image of `algebra_map K L`. Other examples of subobjects available in mathlib are submonoids, subgroups and submodules (with ideals as a special case of submodules).

The new definitions found immediate use: soon after we contributed our definition of `intermediate_field` to mathlib, the Berkeley Galois theory group used it in a proof of the primitive element theorem. Soon after the primitive element theorem was merged into mathlib, we used it in our development of the trace form. This anecdote illustrates the decentralized development style of mathlib, with different groups and people building on each other's results in a collaborative process.

By providing a coercion from subobjects to types, sending a subobject S to the subtype of all elements of S, and putting typeclass instances on this subtype, we can reason about inductively defined rings such as $\mathbb{Z}$ and subrings such as `integral_closure ℤ K` uniformly. If S : `subfield` K, the map that sends $x : S$ to K by "forgetting" that $x \in S$ is a ring embedding, and we register this map as a `algebra S K` instance, also allowing us to treat field extensions of the form $\mathbb{Q} \to \mathbb{C}$ and subfields uniformly. Similarly, for F : `intermediate_field K L`, we defined the corresponding `algebra K F`, `algebra F L` and `is_scalar_tower K F L` instances.

3.5 Fields of fractions

The fraction field $\operatorname{Frac} R$ of an integral domain R can be defined explicitly as a quotient type as follows: starting from the set of pairs (a, b) with $a, b \in R$ such that $b \neq 0$, one quotients by the equivalence relation stating that $(\alpha a, \alpha b) \sim (a, b)$ for all $\alpha \neq 0 : R$, writing the equivalence class of (a, b) as $\frac{a}{b}$. It can easily be proved that the ring structure on R extends uniquely to a ring structure turning K into a field. When $R = \mathbb{Z}$, this yields the traditional description of $\mathbb{Q}$ as the set of equivalence classes of fractions, where $\frac{2}{3} = \frac{-4}{-6}$, etc. The drawback of this construction is that there are many other fields that can serve as the field of fractions for the same ring. For instance, although there is an isomorphism of $\operatorname{Frac} \mathbb{C}[\![t]\!]$ with the field

$$\mathbb{C}(\!(t)\!) = \Big\{ \sum_{i=a}^{+\infty} a_i t^i \quad \text{with } a \in \mathbb{Z} \Big\}$$

of Laurent series, there is no (definitional) equality between the types. Another example comes from the field

$$\mathbb{Q}(i) = \{z \in \mathbb{C} : \Re z \in \mathbb{Q}, \Im z \in \mathbb{Q}\}$$

which is isomorphic to $\operatorname{Frac}(\mathbb{Z}[i])$, but not definitionally equal to it. In fact, even the rational numbers in Lean are a counterexample: for computational efficiency, $\mathbb{Q}$ is defined as a subtype where the numerator and denominator are coprime, instead of a quotient by "scalar multiplication". A definition like

```
def fraction_field (R : Type*) : Type* :=
{ab : R × R // ab.2 ≠ 0}
```

would require transferring results across isomorphisms as soon as one needs to handle a different construction of a field isomorphic to $\operatorname{Frac} R$.

The strategy used in mathlib is to rather allow for many different *fraction fields* of our given integral domain R, as fields K along with an injective *fraction map* $f : R \to K$ which witnesses that all elements of K are "fractions" of elements of R,[3]. and to parametrize every result over the choice of f. The conditions on f imply that K is the smallest field containing R, by showing each injective map $g : R \to A$ such that $g(x)$ has a multiplicative inverse for all $x \neq 0 : R$, can be extended uniquely to a map $K \to A$ compatible with f and g. In particular, if $f_1 : R \to K_1$ and $f_2 : R \to K_2$ are fraction maps, they induce an isomorphism $K_1 \simeq K_2$. The construction of $\operatorname{Frac} R$ then results in *a* field of fractions rather than *the* field of fractions.

This came at a price: informally, at any given stage of one's reasoning, the field K is fixed and the map $f\colon R \to K$ is applied implicitly, just viewing every $x : R$ as $x : K$. It is now impossible to view *range* $f \leq K$ as an inclusion of subalgebras, because the map f is needed explicitly to give the R-algebra structure on K. We use a type synonym `f.codomain := K` and instantiate the R-algebra structure given by f on this synonym.

In the following sections, let $f : R \to K$ be a fraction map.

3.6 Representing simple field extensions

A number field K is defined as a finite extension of $\mathbb{Q}$, or equivalently (by the primitive element theorem) a field of the form $\mathbb{Q}(\alpha)$ for some algebraic number α. The choice of α and is normally underspecified in informal mathematical usage. We can find $\mathbb{Q}(\alpha)$ by adjoining the root of a polynomial: there is an irreducible polynomial $p \in \mathbb{Q}[X]$ such that $\mathbb{Q}[X]/p \simeq K$; we set α to be the image of X in $\mathbb{Q}[X]/p$. We can also take $\alpha : K$ and let $\mathbb{Q}(\alpha)$ to be the smallest subfield of K containing α; then $K = \mathbb{Q}(\alpha)$ means that $\mathbb{Q}(\alpha)$, as a subfield of K, is equal to the subfield $\top$ containing all elements of K. We could also view K and $\mathbb{Q}(\alpha)$ as subfields of an arbitrary larger field F. Because α is algebraic the smallest subring containing α and $\mathbb{Q}$ will be a field, thus we can add two more representations, replacing "smallest subfield" with "smallest subring". Moreover, all subfields/subrings containing $\mathbb{Q}$ are also $\mathbb{Q}$-algebras, so we can additionally replace "subfield" with "intermediate field" and "subring" with "$\mathbb{Q}$-subalgebra". The same continues to hold if we replace the base field $\mathbb{Q}$ with F, thus considering extensions of the form $F(\alpha)$, now requiring that α be a root of some $p \in F[X]$.

The ability to switch between these representations is important: sometimes K and F are fixed and we want an arbitrary α; sometimes α is fixed and we want an arbitrary type representing $F(\alpha)$. The different constructions of $F(\alpha)$ have already been formalized in mathlib,

To find a uniform way to reason about all these equivalent definitions, we chose to formalize the notion of *power basis* to represent simple field extensions, a basis of the form $1, x, x^2, \ldots, x^{n-1} : K$ (viewing K as a F-vector space)[4]. We call x the *generator* and n the *dimension* of this power basis. We defined the following type of power basis, bundling the information of a power basis:

```
structure power_basis (F K : Type*) [field F] [field K] [algebra F K] :=
(gen : S) (dim : ℕ)
(is_basis : is_basis F (λ (i : fin dim), gen ^ (i : ℕ)))
```

[3] In the definition used by mathlib, a fraction map is a special case of a *localization map*. Different localizations restrict the denominators to different subsets of $R \setminus \{0\}$.

[4] In the formalization we generalize this notion to any algebra A over a commutative ring R

We proved that the various notions of simple field extensions are equivalent to the existence of a power basis.

With the `power_basis` structure, we have the ability to parametrize our results, being able to choose the F and K in a simple field extension K/F, or being able to choose the α generating $F(\alpha)$ (by setting `power_basis.gen pb` equal to α). Specializing a result from an arbitrary K with a power basis over F, to a specific value of K such as $F(\alpha) =$ `algebra.adjoin F {`α`}`, is a matter of applying the result to the power basis generated by α, and rewriting `power_basis.gen (adjoin.power_basis F` α`)` $= \alpha$.

4 Dedekind domains

The aim of this section is to introduce the notion of *Dedekind domain* which, as discussed in Section 2 is the right setting to study algebraic properties of number fields.

4.1 Definitions

There are various equivalent conditions, used at various times, for an integral domain R being a Dedekind domain, of which the following three have been formalized in mathlib:

- `is_dedekind_domain R`: R is a Noetherian integral domain, integrally closed in its fraction field and has Krull dimension at most 1;
- `is_dedekind_domain_inv R`: R is an integral domain and nonzero fractional ideals of R have a multiplicative inverse (we discuss the notion and formalization of fractional ideals in Section 4.2);
- `is_dedekind_domain_dvr R`: R is a Noetherian integral domain and the localization of R at each prime ideal is a discrete valuation ring.

We did not use `is_dedekind_domain_dvr` in our project, so we will not discuss this definition further.

Some authors exclude fields from being Dedekind domains, a convention we initially followed. Since we did not encounter any cases where excluding fields was necessary to prove a theorem, we decided to simplify the definition of a Dedekind domain. It is still possible to exclude fields in a theorem by adding an extra hypothesis `¬ is_field R`.

The "main" definition was chosen to be `is_dedekind_domain`, since this condition is usually the one checked in practice [16]. The other two equivalent definitions were added mathlib, before the proof they are indeed equivalent. Having multiple definitions allowed us to do our work in parallel without depending on unformalized results. For example, the proof of unique ideal factorization in a Dedekind domain initially assumed `is_dedekind_domain_inv R`, and the proof that the ring of integers is a Dedekind domain concluded `is_dedekind_domain (ring_of_integers K)`. After the equivalence between `is_dedekind_domain R` and `is_dedekind_domain_inv R` was formalized, we could painlessly replace usages of `is_dedekind_domain_inv R` with `is_dedekind_domain R`. Separating the different definitions meshed well with the contribution philosophy followed by mathlib, preferring small, standalone additions over in-progress work or entire finished projects.

The conditions `is_dedekind_domain` and `is_dedekind_domain_inv` require a fraction field K, although the truth value of the predicates does not depend on the choice of K. For ease of use, we let the type of `is_dedekind_domain` only depend on the domain R by instantiating K in the definition as `fraction_ring R`.

```
class is_dedekind_domain (R : Type*) [integral_domain R] : Prop :=
```

```
(to_is_noetherian_ring : is_noetherian_ring R)
(dimension_le_one : dimension_le_one R)
(is_integrally_closed : integral_closure R (fraction_ring R) = ⊥)
```

Applications of `is_dedekind_domain` can choose a specific fraction field through the following lemma exposing the alternate definition:

```
lemma is_dedekind_domain_iff (f : fraction_map R K) :
  is_dedekind_domain R ↔
    is_noetherian_ring R ∧ dimension_le_one R ∧
    integral_closure R f.codomain = ⊥
```

We mark `is_dedekind_domain` as a typeclass by using the keyword `class` rather than `structure`, allowing the typeclass system to automatically infer the Dedekind domain structure when an appropriate instance is declared, such as for principal ideal domains or rings of integers.

4.2 Fractional ideals

The notion which is pivotal to the definition of the ideal class group of a Dedekind domain is that of *fractional ideals*: given any integral domain R, these are R-ideals divided by some $x : R$, or equivalently R-submodules J of K such that there is an $x : R$ with $xJ \subseteq R$. The reason for introducing them is that, unlike their subset of proper ideals, they form a group under multiplication. As it should be clear from Section 3.5, this notion depends on the field K as well as on the localization map $f\colon R \to K$ allowing to speak about R-submodules of K and, more importantly, to see an element $x : R$ as the element $fx : K$, so as to be able to write the inclusion $f(x)J \subseteq f(R)$. We formalized the definition of fractional ideals relative to a map $f\colon R \to K$ as a type `fractional_ideal f`. We encoded that the structure of fractional ideals does not depend on the choice of fraction map f, which we formalized as an isomorphism `fractional_ideal.canonical_equiv` between the fractional ideals relative to embeddings $f_1\colon R \to K_1$ and $f_2\colon R \to K_2$.

We defined the addition, multiplication and intersection operations on fractional ideals, by showing the corresponding operations on submodules map fractional ideals to fractional ideals. We also proved that these operations give a commutative semiring structure on the type of fractional ideals. For example, multiplication of fractional ideals is defined as:

```
lemma fractional_mul (I J : fractional_ideal f) :
  is_fractional f (I.1 * J.1) := _ -- proof omitted

instance : has_mul (fractional_ideal f) :=
⟨λ I J, ⟨I.1 * J.1, fractional_mul I J⟩⟩
```

Defining the quotient of two fractional ideals requires slightly more work. Consider any R-algebra A and an injection $R \hookrightarrow A$, allowing to look at $x : R$ as $x : A$: given ideals $I, J \leq R$, the submodule quotient $I/J \leq A$ is characterized by the property

```
lemma submodule.mem_div_iff_forall_mul_mem {x : A} {I J : submodule R A} :
  x ∈ I / J ↔ ∀ y ∈ J, x * y ∈ I
```

In our setting, we consider a field of fractions K, together with a localization map $f\colon R \to K$: we can look at every ideal as the fractional ideal $I/1 \leq K$. The first main theoretical result that we need to define the ideal class group is to show that every non-zero ideal $0 < I \leq R$

becomes invertible when seen as a fractional ideal: this means, by definition, that the equality

$$f(I) * \frac{1}{f(I)} = 1 = f(R) \leq K \tag{1}$$

as R-submodules of K, holds. Beware that the notation $1/I$ might be misleading here: indeed, for general integral domains, equation (1) might not hold. An example comes from the product

$$\frac{1}{(X,Y)} * (X,Y) = (X,Y) < \mathbb{C}[X,Y]$$

of the fractional ideals $1/(X,Y)$ and (X,Y) in the fraction field $\mathbb{C}(X,Y)$ of $\mathbb{C}[X,Y]$. On the other hand, it can be proved that Dedekind domain are precisely the right class of integral domains for which (1) always holds. This was formalised as the following

```
lemma fractional_ideal.is_unit {hR : is_dedekind_domain R}
  (I : fractional_ideal f) (hne : I ≠ ⊥) : is_unit I :=
```

together with

```
noncomputable instance [is_dedekind_domain R] (g : fraction_map R K) :
  has_inv (fractional_ideal g) :=
⟨λ I, 1 / I⟩
```

asserting that the inverse of any fractional ideal I (defined as another fractional ideal J such that $I * J = 1$)—which always exists thanks to the `lemma fractional_ideal.is_unit`—is unique and coincides with $1/I$.

Two remarks are in order. The first is that in `lemma fractional_ideal.is_unit` the hypothesis `(hne : I ≠ ⊥)` that I be non-zero is added, and apparently dropped in the `has_inv` instance: this reflects the existence of the typeclass `group_with_zero` in mathlib, consisting of groups endowed with an extra element `0` whose inverse is again `0`. In particular, the zero fractional ideal is invertible (in the mathlib sense) but is not a unit, leading to the strange phenomenon above. Even more fundamentally, the fact that (1) might fail to hold in certain circumstances shows that, for general domains, $1/I \neq I^{-1}$. Since `a / b` used to have the built-in definition $a/b = a * b^{-1}$, the notation $1/I$, defined for every non-zero I, was conflicting with the fact that I might not be invertible. Since, for Dedekind domains, we wanted to *define* I^{-1} as $1/I$, a major refactor of a core definition was needed. In particular, to break the circularity, we had to weaken the definitional equality to a proposition; this involved many small changes throughout mathlib.

4.3 Equivalence of the definitions

We now describe how we proved and formalized that the two definitions `is_dedekind_domain` and `is_dedekind_domain_inv` of being a Dedekind domain are equivalent.

To show that `is_dedekind_domain_inv` implies `is_dedekind_domain`, we follow the proof given by Fröhlich in [11, Chapter 1,§ 2, Proposition 1] . A constant challenge that was faced while coding this proof was already mentioned in Section 3.5, namely the fact that elements of the ring must be traced along the fixed morphism to the fields of fractions.The proofs for being integrally closed and of dimension being less than or equal to 1 are fairly straightforward.

Proving the Noetherian condition was the most challenging. In the original proof by Fröhlich, he considers elements $a_1, \dots, a_n \in I$ and $b_1, \dots, b_n \in I^{-1}$ for any nonempty fractional ideal I, satisfying $\sum_i a_i b_i = 1$. However, it is quite challenging to prove that an element of the product of two R-submodules M and N must be of the form $\sum_{i=1}^m a_i * b_i$, for $a_i \in A$ and $b_i \in B$ for all $1 \leq i \leq m$. Instead, we show that, for every element of an ideal, there exists a `s : finset R` whose span is contained in the ideal, and which contains the element. This is accomplished by the lemma `submodule.mem_span_mul_finite_of_mem_span_mul`. Now considering an ideal I of the ring R, due to its invertibility (as a fractional ideal), by `submodule.mem_span_mul_finite_of_mem_span_mul`, we obtain finite sets $T \subset I$ and $T' \subset 1/I$ of type `finset R`, such that 1 is contained in the R-span of $T * T'$. With coercions, the actual statement of the latter expression in Lean is `↑T' ⊆ ↑↑(1 / ↑s)`, which reads :

```
(T' : set (localization_map.codomain (fraction_ring.of A)) ) ⊆ (((1 / (s :
     fractional_ideal (fraction_ring.of A))) : submodule A (
    localization_map.codomain (fraction_ring.of A))) set (localization_map.
    codomain (fraction_ring.of A)) )
```

This is then sufficient to show that I is finitely generated, as shown in the lemma `fg_of_one_mem_span_mul`.

The theorem `fractional_ideal.mul_inv_cancel` proves the converse, namely that `is_dedekind_domain` implies `is_dedekind_domain_inv`. The classical proof first shows that every maximal ideal `M : ideal R`, seen as a fractional ideal, is invertible; from this, some work allows to show that every non-zero ideal is inverible, using that it is contained in a maximal ideals; and, finally, the fact that every fractional ideal `J : fractional_ideal f` satisfies $xJ \leq I$ for a suitable `x : R` and `I : ideal R` allows to show that every fractional ideal is invertible, concluding the proof that non-zero fractional ideals form a group. We have found that formalizing the second step, so passing from the case where `M` is maximal to the general case, required more code that directly showing invertibility of arbitrary non-zero ideals. We have coded this in the following

```
lemma coe_ideal_mul_one_div [hR : is_dedekind_domain R] (hNF : ¬ is_field
    R)
  (I : ideal R) (hne : I ≠ ⊥) :
  ↑I * ((1 : fractional_ideal f) / ↑I) = (1 : fractional_ideal f) :=
```

from where it becomes apparent that, over and over again, we had to carefully distinguish between the ideal I, which is a term of type `ideal R`, and its coercion `↑I`, which is of type `fractional_ideal f`, although these objects, from a mathematical point of view, are identical.

The proof of the above result relies mainly on the lemma `exists_not_mem_one_of_ne_bot`, which says that for every non-trivial ideal $0 \lneq I \lneq R$, there exists an element in the field K which is not integral (so, not in `f.range`) but lies in $1/I$. This depends crucially on R being Noetherian, since the proof begins by invoking that every non-zero ideal in a Noetherian ring contains a product of non-zero prime ideals. This result was not previously available in mathlib, and we formalized it as `exists_prime_spectrum_prod_le_and_ne_bot_of_domain`. It is when applying this that the dimension condition shows its full force: the constructed prime ideal, being non-zero, will be maximal because the Krull dimension of R is at most 1; from this, the conclusion follows straightforwardly. Having the above lemma at our disposal, we can prove that every ideal $I \neq 0$ is invertible by arguing by contradiction: if $I * 1/I \lneq R$, we can find an element $x \in K \setminus f(R)$ which is in $1/(1 * 1/I)$ thanks to

`exists_not_mem_one_of_ne_bot` and some easy algebraic manipulation will imply that x is actually integral over R. Since R is integrally closed, it must lie in $f(R)$, contradicting its construction.

The final step, when we prove that invertibility of ideals implies that of fractional ones as well, was easy: the material developed for the general theory of `fractional_ideals f` allowed to smootly deduce that a fractional ideal J must be invertible as soon as a certain multiple xJ of it is, and since there always exists a `x : R` satisfying the latter condition (because xJ can be made into a "usual" ideal), this leads to the final `lemma fractional_ideal.is_unit` quoted above.

5 Principal ideal domains are Dedekind

As an example of our definitions, we will discuss in some detail our formalization of the fact that a principal ideal domain is a Dedekind domain. A principal ideal domain (PID) is an integral domain R such that each ideal is generated by one element. There is no explicit definition of PIDs in mathlib, rather it is split up into two hypotheses. One uses `[integral domain R] [is_principal_ideal_ring R]` to denote a PID R, where `is_principal_ideal_ring` is a typeclass defined for all commutative rings:

```
class is_principal_ideal_ring (R : Type*) [comm_ring R] : Prop :=
(principal : ∀ (S : ideal R), S.is_principal)
```

Our proof that the hypotheses `[integral_domain A] [is_principal_ideal_ring A]` imply `is_dedekind_domain A` is relatively short:

```
instance principal_ideal_ring.to_dedekind_domain (A : Type*)
  [integral_domain A] [is_principal_ideal_ring A] :
  is_dedekind_domain A :=
⟨principal_ideal_ring.is_noetherian_ring,
 dimension_le_one.principal_ideal_ring _,
 unique_factorization_monoid.integrally_closed (fraction_ring.of A)⟩
```

Making this an `instance` instead of a `lemma` ensures that the typeclass system can now automatically infer a Dedekind domain structure whenever a principal ideal structure is already available.

The Noetherian property of a Dedekind domain follows easily by the previously defined lemma `principal_ideal_ring.is_noetherian_ring`, since, by definition, each ideal in a principal ideal ring is finitely generated (by a single element).

The lemma `dimension_le_one.principal_ideal_ring` is an instantiation of the existing result `is_prime.to_maximal_ideal` showing a nonzero prime ideal in a PID is maximal. The latter lemma uses the characterization that I is a maximal ideal if and only if any strictly larger ideal $J > I$ is the full ring $\top$. If I is a nonzero prime ideal and $J > I$ in the PID R, we have that the generator j of J is a divisor of the generator i of I. Since I is prime, this implies that either $j \in I$, contradicting the assumption that $J > I$, $i = 0$, contradicting that I is nonzero, or that j is a unit, implying $J = \top$ as desired.

The final condition of a PID being integrally closed is the most challenging. We use the previously defined instance `principal_ideal_ring.to_unique_factorization_monoid` that a PID is a unique factorisation monoid (UFM), to instantiate our proof that every UFM is integrally closed. In the same way that principal ideal domains are generalized to principal ideal rings, mathlib generalizes unique factorization domains to unique factorization monoids.

A commutative monoid R with an absorbing element 0 and injectivity of multiplication is defined to be a UFM, if the relation "x properly divides y" is well-founded (implying each element can be factored as a product of irreducibles) and an element of R is prime if and only if it is irreducible (implying the factorization is unique). The first condition is satisfied for a PID since the Noetherian property implies that the division relation is well-founded. The second condition follows from `principal_ideal_ring.irreducible_iff_prime`. To prove that an irreducible element p is prime, the proof uses that prime elements generate prime ideals and irreducible elements of a PID generate maximal ideals. Since all maximal ideals are prime ideals, the ideal generated by p is maximal, hence prime, thus p is prime. The lemma `irreducible_of_prime` proves the converse holds in any commutative monoid with zero.

In order to show that a UFM is integrally closed, we first proved the Rational Root Theorem, named `denom_dvd_of_is_root`, which states that for polynomial $p : R[X]$ and x an element of the fraction field K such that $p(x) = 0$, the denominator of x divides the leading coefficient of p. If x is integral with minimal polynomial p, the leading coefficient is 1, therefore the denominator is a unit and x is an element of R. This gives us the required lemma `unique_factorization_monoid.integrally_closed`, which states that the integral closure of A in its fraction field is A itself.

6 Rings of integers are Dedekind domains

An important class of Dedekind domains consists of the rings of integers of number fields.Recall that we defined the ring of integers of a number field K as the integral closure of $\mathbb{Z}$ in K. We proved a stronger result: give a Dedekind domain R with fraction field K, if L is a finite separable extension of K, then the integral closure of R in L is a Dedekind domain with fraction field L. Our approach adapts [16], Theorem 3.1. Throughout this section, R will be an integral domain with a field of fractions K (given by the map $f\colon R \to K$), L a field extension of K and S will denote the integral closure of R in L.

The first step is to show that L is a field of fractions for the integral closure, i.e. that there is a map `fraction_map_of_finite_extension f L : fraction_map S L`. We formalized the following definition, which implies the desired result:

```
def fraction_map_of_algebraic (alg : is_algebraic R L)
  (inj : function.injective (algebra_map R L)) :
  fraction_map S L
```

The main content of `fraction_map_of_algebraic` consists of showing that all elements $x : L$ can be written as y/z for elements $y \in S$, $z \in R \subseteq S$; the standard proof of this fact (e.g. [6, Theorem 15.29]) formalizes readily.

Now we are ready to show that the integral closure of R in L is a Dedekind domain, by proving it is integrally closed in L, has Krull dimension at most 1 and is Noetherian. The fact that the integral closure is integrally closed is immediate.

To show the Krull dimension is at most 1, we needed to develop basic going-up theory for ideals. In particular, we show that an ideal I in an integral extension is maximal if it lies over a maximal ideal, and use a result already available in mathlib that a prime ideal I in an integral extension lies over a prime ideal.

```
lemma is_maximal_of_is_integral_of_is_maximal_comap
  {S : Type*} [integral_domain S] [algebra R S]
  (hRS : algebra.is_integral R S) (I : ideal S) [I.is_prime]
```

```
  (hI : is_maximal (I.comap (algebra_map R S))) : is_maximal I

theorem is_prime.comap [hK : K.is_prime] : (comap f K).is_prime
```

The final condition, that the integral closure S of R in L is a Noetherian ring, requires the most work. We start by following the first half of [6, Theorem 15.29], so that it suffices to find a nondegenerate bilinear form B such that all integral $x, y : L$ satisfy $B(x, y) \in$ `integral_closure` R L. We formalized the results in [16], 2.5–2.8, to show the *trace form* is a bilinear form satisfying these requirements.

6.1 The trace form

If L/K is a field extension, we have a bilinear form `lmul` $= \lambda xy : S, xy$. The trace of the linear map `lmul x` is called the *algebra trace* $\mathrm{Tr}_{L/K}(x)$ of x We define the algebra trace as a linear map from L to K:

```
noncomputable def trace : L →ₗ[K] K :=
(linear_map.trace K L).comp (lmul K L).to_linear_map
```

This definition is marked noncomputable since `linear_map.trace` makes a case distinction on the existence of a basis, choosing an arbitrary basis if one exists and returning 0 otherwise. This latter case does not occur in our development.

The *trace form* is a K-bilinear form on L, mapping $x, y : L$ to $\mathrm{Tr}(xy)$.

```
noncomputable def trace_form : bilin_form K L :=
{ bilin := λ x y, trace K L (x * y), .. /- proofs omitted -/ }
```

In fact, we define the trace and trace form for any algebra over a commutative ring. For simplicity of exposition in this paper we will only consider finite field extensions. In the following, let $K/L/F$ be a tower of finite field extensions, i.e. we assume `[algebra K L] [algebra L F] [algebra K F] [is_scalar_tower K L F]`, as described in Section 3.2.

The value of the trace depends on the choice of K and L; we formalized this as lemmas `trace_algebra_map x : trace K L (algebra_map K L x) = findim K L • x` and `trace_comp L x : trace K F x = trace K L (trace L F x)`. These results follow by direct computation.

To compute $\mathrm{Tr}_{K:L}(x)$ it therefore suffices to consider the trace of x in the smallest field containing x and K, which is the simple extension $K(x)$ discussed in Section 3.6. There is a nice formula for the trace in $K(x)$, although the terms in this formula are elements in a larger field F (such as the *splitting field* of the minimal polynomial of x). In formalizing this formula, we must first map the trace to F using the canonical embedding `algebra_map K F`, giving the following lemma statement:

```
lemma power_basis.trace_gen_eq_sum_roots (pb : power_basis K L)
  (h : polynomial.splits (algebra_map K F) pb.minpoly_gen) :
  algebra_map K F (trace K L pb.gen) =
    (pb.minpoly_gen.map (algebra_map K F)).roots.sum
```

We formulate the lemma in terms of the power basis, since we will need to use it for $K(x)$ here and for an arbitary finite separable extension L/K later in the proof.

The elements of `(pb.minpoly_gen.map (algebra_map K F)).roots` are called *conjugates* of x in F. Each conjugate of x is integral since it is a root of (the same) monic polynomial, and integer multiples and sums of integral elements are integral. Combining

`trace_gen_eq_sum_roots` and `trace_algebra_map` together shows that the trace of x is an integer multiple (namely `findim K(x) L`) of a sum of conjugate roots, hence we conclude that the trace (and trace form) of an integral element is also integral.

Finally, we show the trace form is nondegenerate, following [16], Proposition 2.8. Since L/K is a finite, separable field extension, it has a power basis `pb` generated by x. Letting x_k denote the k-th conjugate of x in an algebraically closed field $F/L/K$, the main difficulty lies in checking the equality $\sum_k x_k^{i+j} = \mathrm{Tr}_{L/K}(x^{i+j})$. Directly applying `trace_gen_eq_sum_roots` is tempting, since we have a sum over conjugates of powers on both sides. However, the two expressions will not precisely match: the left hand side is a sum of conjugates of x, where each conjugate is raised to the power $i+j$, while the conclusion of `trace_gen_eq_sum_roots` results in a sum over conjugates of x^{i+j}.

Instead, the informal proof switches here to an equivalent definition of conjugate: the conjugates of x in F are the images (counted with multiplicity) of x under each embedding $\sigma\colon K(x) \to F$ that fixes K. This equivalence between the two notions of conjugate was contributed to mathlib by the Berkeley group in the week before we realized we needed it. Mapping `trace_gen_eq_sum_roots` through the equivalence gives $\mathrm{Tr}_{L/K}(x) = \sum_{\sigma : L \to_a [K]F} \sigma x$. Since σ is a ring homomorphism, $\sigma\ x^{i+j} = (\sigma\ x)^{i+j}$, so the conjugates of x^{i+j} are the $(i+j)$-th powers of conjugates of x, concluding the proof.

7 Class number

The class group is the quotient `units (fractional_ideal f)` modulo the principal fractional ideals, or equivalently the ideals of R (except 0) modulo the elements of R (except 0).

We are interested in the class group because ...

An important property of the ring of integers in a number field is that the class group is finite. The number of elements of the class group is called the class number.

We prove that the class group is finite using a simplified form of the traditional proof, ...

8 Discussion

8.1 Related work

Broadly speaking, one could see the formalization work as part of number theory. There are several formalization result in this direction; see e.g. [4, Section 6], most notably the formalization in Isabelle/HOL of a substantial part of analytic number theory [7]. Narrowing somewhat in on our more algebraic setting, we are not aware of any other formal developments of fractional ideals, Dedekind domains or class groups of global fields. Since our project touches upon the theories of field extensions, ideals, number fields and number rings, we provide here a partial overview of formalizations in these areas.

There are many libraries formalizing basic notions of commutative algebra such as field extensions and ideals, including the Mathematical Components library in Coq [14], the algebraic library for Isabelle/HOL [9], the `set.mm` database for MetaMath [15] and the Mizar Mathematical Library [13]. The field of algebraic numbers, or more generally algebraic closures of arbitrary fields, are also available in many provers, for example Coq [3, 14], Isabelle/HOL [19], MetaMath [1], and Mizar [20]. To our knowledge, the Coq Mathematical Components library is the only formal development except ours specifically dealing with number fields [14, `field/algnum.v`].

Apart from the general theory of algebraic numbers, there are formalizations of specific number rings: the Gaussian integers $\mathbb{Z}[i]$ are available in Isabelle/HOL [8], MetaMath [2] and Mizar [12]. The Isabelle/HOL formalization deserves special mention since it introduces techniques from algebraic number theory, defining the integer-valued norm on $\mathbb{Z}[i]$ and classifying the prime elements of $\mathbb{Z}[i]$.

8.2 Future directions

Having formalized the basics of algebraic number theory, there are several natural directions for future formalization work. These include the following.

- Finiteness of the class group for the ring of integers in all global fields. This would entail dropping the separability condition in the result mentioned in the third line of Section 6, and consequently adapt some of the details in the final steps for the finiteness of the class group in the admissible case. Some basic prerequisites would be setting up some field theory dealing with (finite) inseparable field extensions, especially the purely inseparable ones. Al in all this seems a tedious though reasonable project.
- Finite generation of the group of units of the ring of integers in a number field, or slightly stronger, Dirichlet's unit theorem [16, Theorem 7.4]. This seems a somewhat more involved, but still reasonable, project. The finite generation result also holds for function fields, so actually it would be nice (and doable) to consider all global fields (which would involve finite inseparable field extensions, as in the previous item).
- Other finiteness results in algebraic number theory, most notably Hermitte's theorem about the existence of only finitely many number fields (up to isomorphism, or embedded in a fixed algebraically closed field containing $\mathbb{Q}$, e.g. $\mathbb{C}$) with discriminant below a given bound [16, Theorem 2.16]. This would be significantly more involved than the previous items and would require, amongst other things, setting up a lot of ramification theory (which is very important for algebraic number theory). As usual, there are analogue results in the function field setting, though they are less straightforward. One reason being that the nondegenerateness of the trace form from Section 6.1 does not hold any more when the separability condition is dropped.
- Computational aspect. Our formalization lays some foundations to the verification of number theoretic software, such as KASH/KANT [10] and PARI/GP [18]. Verifying computations for class groups, or just class numbers, in the case of 'small' (e.g. some quadratic) number fields, looks within reach. Of course, getting really efficient algorithms (or certificates), is a hard research topic.
- Number theoretic applications. All of the above items consider theoretical of computational aspects within algebraic number theory itself. There are many applications of these, e.g. solving Diophantine equations. Solving Mordell equations, i.e. for a given nonzero integer D determining all pairs of integers (x, y) such that $y^2 = x^3 + D$, could be an interesting first case study (dealing with some values of D where elementary methods fail).

8.3 Conclusion

In this project, we found that the rule holds that the hardest part of formalization is to get the definitions just right. Once this is accomplished, the informal proof almost always translated to a formal proof without too much effort. In particular, we regularly had to invent abstractions to treat instances the "same" situation uniformly. Instead of fixing a canonical representation such $K \subseteq L \subseteq F$ as subfields, $\operatorname{Frac} F$ as the field of fractions, or

$K(\alpha)$ as the simple field extension, we find that making the essence of the situation an explicit parameter, as in `is_scalar_tower`, `fraction_map` or `power_basis`, treats equivalent viewpoints uniformly without the need for transferring results.

The formalization efforts described in this paper cannot be cleanly separated from the development of mathlib as a whole. The decentralized organization and highly integrated design of mathlib meant we could contribute our formalizations as we completed them, resulting in a quick integration into the rest of the library. Other contributors building on these results often extended them to meet our requirements, before we could identify that we needed them, as the anecdote in Section 3.4 illustrates. In other words, the low barriers for contributions ensured mutually beneficial collaboration.

The formalization project described in this paper resulted in the contribution of thousands of lines of Lean code involving hundreds of declarations. We validated existing design choices used in mathlib, refactored those that did not scale well and contributed our own set of designs. The real achievement was not to complete each proof, but to build a better foundation for formal mathematics.

References

1. Mario Carneiro. Definition `df-aa`. `http://us.metamath.org/mpeuni/df-aa.html`.
2. Mario Carneiro. Definition `df-gz`. `http://us.metamath.org/mpeuni/df-gz.html`.
3. Cyril Cohen. Construction of real algebraic numbers in Coq. In Lennart Beringer and Amy P. Felty, editors, *Interactive Theorem Proving - Third International Conference, ITP 2012, Princeton, NJ, USA, August 13-15, 2012. Proceedings*, volume 7406 of *Lecture Notes in Computer Science*, pages 67–82. Springer, 2012. `doi:10.1007/978-3-642-32347-8\_6`.
4. Sander R. Dahmen, Johannes Hölzl, and Robert Y. Lewis. Formalizing the Solution to the Cap Set Problem. In John Harrison, John O'Leary, and Andrew Tolmach, editors, *10th International Conference on Interactive Theorem Proving (ITP 2019)*, volume 141 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 15:1–15:19, Dagstuhl, Germany, 2019. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. URL: `http://drops.dagstuhl.de/opus/volltexte/2019/11070`, `doi:10.4230/LIPIcs.ITP.2019.15`.
5. Leonardo de Moura, Soonho Kong, Jeremy Avigad, Floris van Doorn, and Jakob von Raumer. The lean theorem prover (system description). In A. P. Felty and A. Middeldorp, editors, *Automated Deduction - CADE-25*, volume 9195 of *LNCS*, pages 378–388. Springer, Cham, 2015. `doi:10.1007/978-3-319-21401-6_26`.
6. David S. Dummit and Richard M. Foote. *Abstract algebra*. John Wiley & Sons, Inc., Hoboken, NJ, third edition, 2004.
7. Manuel Eberl. Nine chapters of analytic number theory in Isabelle/HOL. In John Harrison, John O'Leary, and Andrew Tolmach, editors, *Interactive Theorem Proving, ITP 2019*, volume 141 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 16:1–16:19. Schloss Dagstuhl, Leibniz-Zentrum fuer Informatik, 2019. `doi:10.4230/LIPIcs.ITP.2019.16`.
8. Manuel Eberl. Gaussian integers. *Archive of Formal Proofs*, April 2020. `https://isa-afp.org/entries/Gaussian_Integers.html`, Formal proof development.
9. Clemens Ballarin (editor), Jesús Aransay, Martin Baillon, Paulo Emílio de Vilhena, Stephan Hohe, Florian Kammüller, and Lawrence C Paulson. The Isabelle/HOL algebra library. `http://isabelle.in.tum.de/dist/library/HOL/HOL-Algebra/index.html`.
10. M. E. Pohst et al. The computer algebra system KASH/KANT. `http://www.math.tu-berlin.de/~kant`.
11. A. Fröhlich. Local fields. In *Algebraic Number Theory (Proc. Instructional Conf., Brighton, 1965)*, pages 1–41. Thompson, Washington, D.C., 1967.

12 Y. Futa, D. Mizushima, and H. Okazaki. Formalization of Gaussian integers, Gaussian rational numbers, and their algebraic structures with Mizar. In *2012 International Symposium on Information Theory and its Applications*, pages 591–595, 2012.

13 Adam Grabowski, Artur Kornilowicz, and Christoph Schwarzweller. On algebraic hierarchies in mathematical repository of mizar. In M. Ganzha, L. Maciaszek, and M. Paprzycki, editors, *Proceedings of the 2016 Federated Conference on Computer Science and Information Systems*, volume 8 of *ACSIS*, pages 363–371, 2016.

14 Assia Mahboubi and Enrico Tassi. *The Mathematical Components Libraries*. `https://math-comp.github.io/mcb/`, 2017.

15 Norman D. Megill and David A. Wheeler. *Metamath: A Computer Language for Mathematical Proofs*. Lulu Press, Morrisville, North Carolina, 2019. `http://us.metamath.org/downloads/metamath.pdf`.

16 Jürgen Neukirch. *Algebraic number theory*, volume 322 of *Grundlehren der Mathematischen Wissenschaften [Fundamental Principles of Mathematical Sciences]*. Springer-Verlag, Berlin, 1999. Translated from the 1992 German original and with a note by Norbert Schappacher, With a foreword by G. Harder. `doi:10.1007/978-3-662-03983-0`.

17 The mathlib Community. The lean mathematical library. In J. Blanchette and C. Hriţcu, editors, *CPP 2020*, page 367–381. ACM, 2020. `doi:10.1145/3372885.3373824`.

18 The PARI Group, Univ. Bordeaux. *PARI/GP version 2.11.2*, 2019. available from `http://pari.math.u-bordeaux.fr/`.

19 René Thiemann, Akihisa Yamada, and Sebastiaan Joosten. Algebraic numbers in Isabelle/HOL. *Archive of Formal Proofs*, December 2015. `https://isa-afp.org/entries/Algebraic_Numbers.html`, Formal proof development.

20 Yasushige Watase. Algebraic numbers. *Formalized Mathematics*, 24(**4**):291–299, 2016. `doi:10.1515/forma-2016-0025`.