A formalization of Dedekind domains and class groups of global fields

Anne Baanen
Department of Computer Science, Vrije Universiteit Amsterdam, The Netherlands

Sander R. Dahmen
Department of Mathematics, Vrije Universiteit Amsterdam, The 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.ITP.2021.

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`

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 Grant EP/S021590/1 (UK)

Acknowledgements We would like to thank Jasmin Blanchette for useful comments on a first version of the manuscript.
A. N. would like to thank Prof. Kevin Buzzard for his constant support and encouragement, and for introducing her to the other co-authors.
A. N. and F. N. wish to express their deepest gratitude to Anne Baanen for the generosity shown along all stages of the project. Without Anne's never-ending patience, it would have been impossible for them to contribute to this project, and to overcome several difficulties.
Finally, we would like to thank the whole mathlib community for invaluable advice all along the project.

1 Introduction

In its basic form, number theory studies properties of the integers $\mathbb{Z}$ and its fraction field, the rational numbers $\mathbb{Q}$. 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 extensions of $\mathbb{Q}$, called *number fields*, as well as their *rings of integers* (Section 2),

Interactive Theorem Proving 2021 (ITP 2021).
Editors: John Q. Open and Joan R. Access; Article No. ; pp. :1–:18
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

whose relations mirror the way $\mathbb{Q}$ contains $\mathbb{Z}$ as a subring. In this paper, we describe our project aiming at formalizing 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 rather to lay the foundations 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 (Section 3.3), Dedekind domains (Section 4), and the ideal class group and class number (Section 7). Apart form the very basics concerning number fields, these concepts were not formalized before as far as we are aware of. We note that our formal definition of the class number is an essential requirement for the use of theorem provers in modern number theory research. The main proofs that we formalized show that two definitions of Dedekind domains are equivalent (Section 4.3), that the ring of integers is a Dedekind domain (Section 6) and that the class group of a number field is finite (Section 7). In fact, most of our results for number fields are also obtained in the more general setting of *global fields*.

Our work is developed as part of the mathematical library mathlib [20] for the Lean 3 theorem prover [6]. 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 : ...` ; "an arbitrary `Type u`" is abbreviated as `Type*`. 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. In this project, as part of the development of mathlib, we follow this philosophy by contributing pieces of our work as they are finished. We, in turn, use 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 over 100 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 we will not stress whom we consider to be the main authors.

The source files of the formalization are currently in the process of being merged into mathlib. The up-to-date development branch is publically available.[1] We also maintain a repository[2] containing the source code referred to in this paper.

2 Mathematical background

Let us now introduce some of the main objects we study, described informally. We assume some familiarity with basic ring and field theory.

A *number field* K is a finite extension of the field $\mathbb{Q}$, and as such has the structure of a finite dimensional vector space over $\mathbb{Q}$; its dimension is called the *degree* of K. The

[1] https://github.com/leanprover-community/mathlib/tree/dedekind-domain-dev
[2] https://github.com/lean-forward/class-number

easiest example is $\mathbb{Q}$ itself, and the two-dimensional cases are given by the quadratic number fields $\mathbb{Q}(\sqrt{d}) = \{a + b\sqrt{d} : a, b \in \mathbb{Q}\}$ where $d \in \mathbb{Z}$ is not a square. For an interesting cubic example, let α be the unique real number satisfying $\alpha^3 + \alpha^2 - 2\alpha + 8 = 0$. It gives rise to the number field $\mathbb{Q}(\alpha) = \{a + b\alpha + c\alpha^2 : a, b, c \in \mathbb{Q}\}$. In general, taking any root α of an irreducible polynomial of degree n over $\mathbb{Q}$ yields a number field of degree n: $\mathbb{Q}(\alpha) = \{c_0 + c_1\alpha + \ldots + c_{n-1}\alpha^{n-1} : c_0, c_1, \ldots, c_{n-1} \in \mathbb{Q}\}$, and, up to isomorphism, these are all the number fields of degree n.

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

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

where we recall that a polynomial is called *monic* if its leading coefficient equals 1. While it might not be immediately obvious that $\mathcal{O}_K$ is a ring, this follows from general algebraic properties of integral closures. Some examples of $\mathcal{O}_K$ are the following. Taking $K = \mathbb{Q}$, we get $\mathcal{O}_K = \mathbb{Z}$ back. For $K = \mathbb{Q}(i) = \mathbb{Q}(\sqrt{-1})$ we get that $\mathcal{O}_K$ is the ring of Gaussian integers $\mathbb{Z}[i] = \{a + bi : a, b \in \mathbb{Z}\}$. 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 $\mathcal{O}_K = \mathbb{Z}[\varphi] = \{a + b\varphi : a, b \in \mathbb{Z}\}$. 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. Further well-known rings of integers are the Eisenstein integers $\mathbb{Z}[(1 + \sqrt{-3})/2]$ and the ring $\mathbb{Z}[\sqrt{2}]$.

Thinking of $\mathcal{O}_K$ as a generalization of $\mathbb{Z}$, it is natural to ask which of its properties still hold in $\mathcal{O}_K$ and, when this fails, if a reasonable weakening does.

An important property of $\mathbb{Z}$ is that it is a principal ideal domain (PID), meaning that every ideal is generated by one element. This implies that every nonzero nonunit element can be written as a finite product of prime elements, which is unique up to reordering and multiplying by ± 1: a ring where this holds is called a unique factorization domain, or UFD. For example, 6 can be factored in primes in 4 equivalent ways, namely $6 = 2 \cdot 3 = 3 \cdot 2 = (-2) \cdot (-3) = (-3) \cdot (-2)$. In fact, the previously mentioned examples of rings of integers are UFDs, but this is certainly not true for all rings of integers. For example, unique factorization *does not* hold in $\mathbb{Z}[\sqrt{-5}]$: it is easy to prove that $6 = 2 \cdot 3$ and $6 = (1 + \sqrt{-5})(1 - \sqrt{-5})$ provide two essentially different ways to factor 6 into prime elements of $\mathbb{Z}[\sqrt{-5}]$.

As it turns out, there is a way to remedy this. Namely, by considering factorization of *ideals* instead of elements: given a number field K, with ring of integers $\mathcal{O}_K$, a beautiful and classical result by Dedekind shows that every nonzero ideal of $\mathcal{O}_K$ can be factored as a product of prime ideals in a unique way, up to reordering.

Although unique factorization in terms of ideals is of great importance, it is still interesting, and sometimes necessary, to also consider factorization properties in terms of elements. We mentioned that unique factorization in $\mathbb{Z}$ follows from the fact that every ideal is generated by a single element. We can extend the monoid of ideals of $\mathbb{Z}$ to a group of *fractional ideals*. These are additive subgroups of $\mathbb{Q}$ of the form $\frac{1}{d}I$ with I an ideal of $\mathbb{Z}$ and d a nonzero integer. When the distinction is important, we refer to an ideal $I \subseteq \mathbb{Z}$ as an *integral ideal*. The nonzero fractional ideals of $\mathbb{Z}$ naturally form a multiplicative group (whereas there is no integral ideal $I \subseteq \mathbb{Z}$ such that $I * (2\mathbb{Z}) = (1)$). The statement that every ideal is generated by a single element translates to the fact that the quotient group of nonzero fractional ideals modulo $\mathbb{Q}^\times$ (where $\frac{a}{b} \in \mathbb{Q}^\times$ corresponds to $\frac{1}{b}a\mathbb{Z}$) is trivial.

It turns out that this quotient group can be defined for every ring of integers $\mathcal{O}_K$. The fundamental theoretical notion beneath this construction is that of Dedekind domain:

these are integral domains D which are Noetherian (every ideal of D is finitely generated), integrally closed (if an element x in the fraction field of D is a root of a monic polynomial with coefficients in D, then actually $x \in D$), and of Krull dimension at most 1 (every nonzero prime ideal of D is maximal). It can be proved that the nonzero fractional ideals of D again form a group, and the quotient of this group by the image of the natural embedding of $(\operatorname{Frac} D)^\times$ is called the (*ideal*) *class group* $\mathcal{Cl}_D$.

What is arithmetically crucial is the theorem ensuring that the ring of integers $\mathcal{O}_K$ of every number field K is a Dedekind domain, and that in this case the class group $\mathcal{Cl}_{\mathcal{O}_K}$ is actually *finite*. In particular, $\mathcal{Cl}_{\mathcal{O}_K}$ can be seen as "measuring" how far ideals of $\mathcal{O}_K$ are from being generated by a single element and, consequently, as a measure of the failure of unique factorization. The order of $\mathcal{Cl}_{\mathcal{O}_K}$ is called *the class number* of K. Intuitively, then, the smaller the class number, the fewer factorizations are possible.

The statements in the previous paragraph also hold for *function fields*, namely fields which are finite extensions of $\mathbb{F}_q(t) \simeq \operatorname{Frac} \mathbb{F}_q[t]$, where $\mathbb{F}_q$ is a finite field with q elements. Recall that when q is a prime number, $\mathbb{F}_q$ is simply the field $\mathbb{Z}/q\mathbb{Z}$. A field which is either a number field or a function field is called a *global field*.

In the next sections we will describe the formalization of the above concepts.

3 Number fields, global fields and rings of integers

We refer the reader to Section 2 for the mathematical background needed in this section.

We formalized number fields as the following typeclass:

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

The *class* keyword declares a structure type (in other words, a type of records) and enables typeclass inference for terms of this type. Round brackets mark parameters explicitly supplied by the user, such as `(K : Type*)`, square brackets mark instance parameters inferred by the typeclass system, such as `[field K]`. The condition `[cz : char_zero K]` states that K has characteristic zero, so 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 instance search), endowing K with the $\mathbb{Q}$-vector space structure used in the `[fd : finite_dimensional ℚ K]` hypothesis.

We defined the function fields K over a finite field $\mathbb{F}_q$ using the following typeclass:

```
class is_function_field_over {𝔽q F : Type*} [field 𝔽q] [fintype 𝔽q]
  [field F] (f : fraction_map (polynomial 𝔽q) F) (L : Type*) [field L]
  [algebra f.codomain L] : Prop :=
[fd : finite_dimensional f.codomain L]
```

Curly brackets mark implicit parameters inferred through unification, such as `{𝔽q F : Type*}`. The map `f` witnesses that F is a fraction field of the polynomial ring $\mathbb{F}_q[t]$, the notation `f.codomain` endows F with the $\mathbb{F}_q[t]$-algebra structure of $\mathbb{F}_q(t)$. We present a more detailed analysis of `fraction_map` in Section 3.5.

3.1 Field extensions

The definition of `is_number_field` illustrates our treatment of field extensions. 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 $\overline{K}$ of K, and $\overline{K}$ is contained in $\mathbb{C}$. We might formalize this situation by viewing $\mathbb{Q}$, K, L and $\overline{K}$ as sets of complex numbers $\mathbb{C}$ and defining field extensions as subset relations between these subfields. This way, no coercions need to be inserted in order to map elements of one field into a larger field. Unfortunately, we can only avoid coercions as far as we are able to stay within one largest field. For example, the definition of complex numbers depends on many results for rational numbers, which would need to be proved again, or transported, for the subfield of $\mathbb{C}$ isomorphic to $\mathbb{Q}$.

Instead, we formalized results about field extensions through parametrization. The fields K and L can be arbitrary types and the hypothesis "L is a field extension of K" is represented by an instance parameter `[algebra K L]` denoting a K-algebra structure on L. There are multiple possible K-algebra structures for a field L and Lean does not enforce uniqueness of typeclass instances, but the mathlib maintainers try to ensure all instances that can be inferred are definitionally equal. The `algebra` structure 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 $\mathbb{Q} \to K$ followed by $K \to L$. Such an equality cannot always be achieved by defining the map $\mathbb{Q} \to L$ to be this composition: in the example, the map $\mathbb{Q} \to K$ depends on the map $\mathbb{Q} \to L$.

The solution in mathlib is to parametrize over all three maps, as long a there is also a proof of coherence: a hypothesis of the form "$L/K/F$ is a tower of field extensions" is translated into 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 state 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 $x : K$ with $y = (1 : L)$ and $z = (1 : F)$ shows that the `algebra_maps` indeed commute.

Common `is_scalar_tower` instances are declared in mathlib, such as for the maps $R \to S \to A$ when S is a R-subalgebra of A. The effect is that almost all coherence proof obligations are automated through typeclass instance search.

3.3 Rings of integers

When K is a number field, the ring $\mathcal{O}_K$ 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 roots of monic polynomials with coefficients in $\mathbb{Z}$, which we formalized as:

```
def number_field.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.

When K is a function field over the finite field $\mathbb{F}_q$, we defined $\mathcal{O}_K$ analogously as `integral_closure (polynomial K) F`. To treat both definitions of ring of integers on an equal footing, we will work with the integral closure of any principal ideal domain when possible.

3.4 Subobjects

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

Two new subobjects that we defined in our development were `subfield` as well as `intermediate_field`. We defined 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 defined an intermediate field as a subfield that is also a K-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 formalization 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 could reason about inductively defined rings such as $\mathbb{Z}$ and subrings such as `integral_closure ℤ K` uniformly. If S : `subfield` K, there is a canonical ring embedding, the map that sends $x : S$ to K by "forgetting" that $x \in S$, and we registered this map as an `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 generated by $(\alpha a, \alpha b) \sim (a, b)$ 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 field structure on $\operatorname{Frac} R$; in mathlib this construction is called `fraction_ring R`. 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. Consider the field $\{z \in \mathbb{C} : \Re z \in \mathbb{Q}, \Im z \in \mathbb{Q}\}$, which is isomorphic to $\mathrm{Frac}(\mathbb{Z}[i])$ but not definitionally equal to it.

The strategy used in mathlib is to rather allow for many different *fraction fields* of our given integral domain R, as fields F along with an injective *fraction map* $f \colon R \to F$ which witnesses that all elements of F are "fractions" of elements of R, and to parametrize every result over the choice of f. In the definition used by mathlib, a fraction map is a special case of a *localization map*. Different localizations restrict the denominators to different multiplicative submonoids of $R \setminus \{0\}$.

The conditions on f imply that F is the smallest field containing R, expressed by the following unique mapping property. If $g \colon R \to A$ is an injective map to a ring A such that $g(x)$ has a multiplicative inverse for all $x \neq 0 : R$, then it can be extended uniquely to a map $F \to A$ compatible with f and g. In particular, if $f_1 \colon R \to F_1$ and $f_2 \colon R \to F_2$ are fraction maps, they induce an isomorphism $F_1 \simeq F_2$. The construction of $\mathrm{Frac}\, R$ then results in *a* field of fractions (with fraction map `fraction_ring.of R`) rather than *the* field of fractions.

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

3.6 Representing monogenic field extensions

In Section 2 we have informally said that every number field K can be written as $K = \mathbb{Q}(\alpha)$ for a root α of an irreducible polynomial $P \in \mathbb{Q}[X]$. This can be made precise in several ways. For instance, one can consider a large field E (of characteristic 0) where P splits completely, then choose a root $\alpha \in E$ and let $\mathbb{Q}(\alpha)$ be the smallest subfield of E containing α. Or, one can consider the quotient ring $\mathbb{Q}[X]/P$ and observe that this is a field where the class $X \pmod{P}$ is a root of P. The assignment $\alpha \mapsto X \pmod{P}$ yields an isomorphism of the two fields, but any other choice of a root $\alpha' \in E$ leads to another isomorphism $\mathbb{Q}(\alpha') \cong \mathbb{Q}[X]/P$. Although mathematically we often tacitly identify the constructions, there is no canonical representation of the *monogenic* extensions of $\mathbb{Q}$, those which can be obtained by adjoining a single root of one polynomial.

The same continues to hold if we replace the base field $\mathbb{Q}$ with another field F, thus considering extensions of the form $F(\alpha)$, now requiring that α be a root of some $P \in F[X]$. Various constructions of $F(\alpha)$ have already been formalized in mathlib. 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 $F(\alpha)$.

To find a uniform way to reason about all these definitions, we chose to formalize the notion of *power basis* to represent monogenic field extensions: this is a basis of the form $1, x, x^2, \ldots, x^{n-1} : K$ (viewing K as a F-vector space). We defined a structure type bundling the information of a power basis. Omitting some generalizations not needed in this paper, the definition reads:

```
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)))
```

We formalized that the previously defined notions of monogenic field extensions are equivalent to the existence of a power basis.

With the `power_basis` structure, we gained the ability to parametrize our results, being able to choose the F and K in a monogenic field extension K/F, or being able to choose the α generating $F(\alpha)$ (by setting `power_basis.gen pb` equal to α). To specialize 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}`, one can apply the result to the power basis generated by α and rewrite `power_basis.gen (adjoin.power_basis F α)` $= \alpha$.

4 Dedekind domains

The right setting to study algebraic properties of number fields are *Dedekind domains*. We formalized fundamental results on Dedekind domains, including the equivalence of two definitions of Dedekind domain.

4.1 Definitions

There are various equivalent conditions, used at various times, for an integral domain D to be a Dedekind domain. The following three have been formalized in mathlib:

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

Note that fields are Dedekind domains according to these conventions.

The mathlib community chose `is_dedekind_domain` as the main definition, since this condition is usually the one checked in practice [17]. The other two equivalent definitions were added to mathlib, but before formalizing the proof that 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 D`, and the proof that the ring of integers $\mathcal{O}_K$ is a Dedekind domain concluded `is_dedekind_domain (ring_of_integers K)`. After the equivalence between `is_dedekind_domain D` and `is_dedekind_domain_inv D` was formalized, we could easily replace usages of `is_dedekind_domain_inv` with `is_dedekind_domain`.

The conditions `is_dedekind_domain` and `is_dedekind_domain_inv` require a fraction field F, although the truth value of the predicates does not depend on the choice of F. For ease of use, we let the type of `is_dedekind_domain` depend only on the domain D by instantiating F in the definition as `fraction_ring D`. From now on, we fix a fraction map $f\colon D \to F$.

```
class is_dedekind_domain (D : Type*) [integral_domain D] : Prop :=
(to_is_noetherian_ring : is_noetherian_ring D)
(dimension_le_one : dimension_le_one D)
(is_integrally_closed : integral_closure D (fraction_ring D) = ⊥)
```

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 D F) :
  is_dedekind_domain D ↔
```

```
    is_noetherian_ring D ∧ dimension_le_one D ∧
    integral_closure D f.codomain = ⊥
```

We marked `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 PIDs or rings of integers.

4.2 Fractional ideals

The notion that is pivotal to the definition of the ideal class group of a Dedekind domain is that of *fractional ideals*: given any integral domain R with a field of fractions F, these are R-submodules J of F such that there is an $x : R$ with $xJ \subseteq R$. For a Dedekind domain, they form a group under multiplication. As seen in Section 3.5, this notion depends on the field F as well as on the fraction map $f\colon R \to F$. A more precise way of stating the above condition is then $f(x)J \subseteq f(R)$. We formalized the definition of fractional ideals relative to a map $f\colon R \to F$ as a type `fractional_ideal f`. The structure of fractional ideals does not depend on the choice of a fraction map, which we formalized as an isomorphism `fractional_ideal.canonical_equiv` between the fractional ideals relative to fraction maps $f_1\colon R \to F_1$ and $f_2\colon R \to F_2$.

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 formalized 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 $R \hookrightarrow A$. 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
```

Beware that the notation $1/I$ might be misleading here: indeed, for general integral domains, the equality $I * 1/I = 1$ might not hold. As an example, one can consider the ideal (X, Y) in $\mathbb{C}[X, Y]$. On the other hand, we formalized that this equality holds for Dedekind domains (Section 4.3) as the following lemma:

```
lemma fractional_ideal.is_unit {hD : is_dedekind_domain D}
  (I : fractional_ideal f) (hne : I ≠ ⊥) : is_unit I
```

This justifies the notation $I^{-1} = 1/I$. In fact, we define this notation even for the ideal 0, by declaring that $0^{-1} = 0$. 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`.

Moreover, mathlib used to define $a/b := a * b^{-1}$, but our definition of $I^{-1} = 1/I$ would cause circularity. This led us to a major refactor of this core definition. In particular, 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. Let D be a Dedekind domain, and $f\colon D \to F$ a fraction map to a field of fractions F of D.

To show that `is_dedekind_domain_inv` implies `is_dedekind_domain`, we follow the proof given by Fröhlich in [11, Chapter 1, § 2, Proposition 1.2.1]. 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 fraction map. The proofs for being integrally closed and of dimension being less than or equal to 1 are fairly straightforward.

Formalizing the Noetherian condition was the most challenging. Fröhlich considers elements $a_1, \ldots, a_n \in I$ and $b_1, \ldots, b_n \in I^{-1}$ 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 D-submodules A and B 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 $A * B$, there are finite sets $T \subseteq A$, $T' \subseteq B$ such that `x : span (T * T')`, formalized as `submodule.mem_span_mul_finite_of_mem_mul`. Now considering a nonzero integral ideal I of the ring D, its invertibility allows to write `1 : (1 : fractional_ideal f) = I * 1 / I`. Hence, we obtain finite sets $T \subset I$ and $T' \subset 1/I$ such that 1 is contained in the D-span of $T * T'$. We used `norm_cast` to resolve most coercions, however, this tactic did not solve coercions coming from the fraction map. With coercions, the actual statement of the latter expression in Lean is `↑T’ ⊆ ↑↑(1 / ↑I)`, which reads

```
(T’ : set (fraction_ring.of D).codomain) ⊆
  (((1 / (I : fractional_ideal (fraction_ring.of D)))
    : submodule D (fraction_ring.of D).codomain)
    : set (fraction_ring.of D).codomain
```

The lemma `fg_of_one_mem_span_mul` then shows that I is finitely generated, concluding the proof.

The theorem `fractional_ideal.mul_inv_cancel` proves the converse, namely that `is_dedekind_domain` implies `is_dedekind_domain_inv`. The classical proof consists of three steps: first, every maximal ideal $M \subseteq D$, seen as a fractional ideal, is invertible; secondly, every nonzero ideal is invertible, using that it is contained in a maximal ideal; thirdly, the fact that every fractional ideal J satisfies $xJ \leq I$ for a suitable $x \in D$ and an ideal $I \subseteq D$ implies that every fractional ideal is invertible, concluding the proof that nonzero fractional ideals form a group. The third step was easy, building upon the material developed for the general theory of `fractional_ideals f`. Concerning the first two, we found that passing from the case where M is maximal to the general case required more code than directly showing invertibility of arbitrary nonzero ideals. The formal statement reads

```
lemma coe_ideal_mul_one_div [hD : is_dedekind_domain D]
  (I : ideal D) (hne : I ≠ ⊥) :
  ↑I * ((1 : fractional_ideal f) / ↑I) = (1 : fractional_ideal f)
```

from where it becomes apparent that we had to repeatedly distinguish between `I : ideal D`, and its coercion `↑I : fractional_ideal f` although these objects, from a mathematical point of view, are identical.

The formal proof of the above result relies on the lemma `exists_not_mem_one_of_ne_bot`, which says that for every non-trivial ideal $0 \subsetneq I \subsetneq D$, there exists an element in the field F which is not integral (so, not in $f(D)$) but lies in $1/I$. The proof begins by invoking that every nonzero ideal in the Noetherian ring D contains a product of nonzero prime ideals. This result was not previously available in mathlib. The dimension condition shows its full force when applying this lemma: each prime ideal in the product, being nonzero, will be maximal because the Krull dimension of D is at most 1; from this, exists_not_mem_one_of_ne_bot follows easily. Having the above lemma at our disposal, we were able to prove that every ideal $I \neq 0$ is invertible by arguing by contradiction: if $I * 1/I \lneq D$, we can find an element $x \in F \setminus f(R)$ 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 D. Since D is integrally closed, it must lie in $f(D)$, contradicting the construction of x. Combining these results gives the equivalence between the two conditions for being a Dedekind domain.

5 Principal ideal domains are Dedekind

As an example of our definitions, we discuss in some detail our formalization of the fact that a principal ideal domain is a Dedekind domain. 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 : ∀ (I : ideal R), is_principal I)
```

Our proof that the hypotheses [integral_domain R] [is_principal_ideal_ring R] imply is_dedekind_domain R was relatively short:

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

The instance keyword marks the declaration for inference by the typeclass system.

The Noetherian property of a Dedekind domain followed 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).

We proved the lemma dimension_le_one.principal_ideal_ring, which 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 $J \supsetneq I$ is the full ring R. If I is a nonzero prime ideal and $J \supsetneq I$ 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 $J \supsetneq I$, $i = 0$, contradicting that I is nonzero, or that j is a unit, implying $J = R$ as desired.

The final condition of a PID being integrally closed was the most challenging. We used 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 followed 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. We proved the lemma `irreducible_of_prime`, which shows the converse holds in any commutative monoid with zero.

To show that a UFM is integrally closed, we first formalized the Rational Root Theorem, named `denom_dvd_of_is_root`, which states that for a polynomial $p : R[X]$ and an element of the fraction field $x : \mathrm{Frac}\, R$ 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 gave us the required lemma `unique_factorization_monoid.integrally_closed`, which states that the integral closure of R in its fraction field is R itself.

6 Rings of integers are Dedekind domains

An important classical result in algebraic number theory is that the ring of integers of a number field K, defined as the integral closure of $\mathbb{Z}$ in K, is a Dedekind domain. We formalized a stronger result: given a Dedekind domain D and a field of fractions F, if L is a finite separable extension of F, then the integral closure of D in L is a Dedekind domain with fraction field L. Our approach was adapted from Neukirch [17, Theorem 3.1]. Throughout this section, let D be a Dedekind domain with a field of fractions F (given by the map $f\colon D \to F$), L a finite, separable field extension of F and let S denote the integral closure of D in L.

The first step was to show that L is a field of fractions for the integral closure, namely, there is a map `fraction_map_of_finite_extension f L : fraction_map S L`. The main content of `fraction_map_of_finite_extension` consisted of showing that all elements $x : L$ can be written as y/z for elements $y \in S$, $z \in D \subseteq S$; the standard proof of this fact (see [7, Theorem 15.29]) formalized readily.

We could then show that the integral closure of D 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 was immediate.

To show the Krull dimension is at most 1, we needed to develop basic going-up theory for ideals. In particular, we showed that an ideal I in an integral extension is maximal if it lies over a maximal ideal, and used 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
  (I : ideal S) [is_prime I]
  (hI : is_maximal (comap f I)) : is_maximal I
theorem is_prime.comap (I : ideal S) [hI : is_prime I] :
  is_prime (comap f I)
```

The final condition, that the integral closure S of D in L is a Noetherian ring, required the most work. We started by following the first half of [7, Theorem 15.29], so that it sufficed to find a nondegenerate bilinear form B such that all integral $x, y : L$ satisfy $B(x, y) \in$ `integral_closure` D L. We formalized the results in Neukirch [17, §§ 2.5–2.8], and showed that the *trace form* is a bilinear form satisfying these requirements.

6.1 The trace form

In the notation from the previous section, consider the bilinear form `lmul := λ x y : L, x * y`. The trace of the linear map `lmul x` is called the *algebra trace* $\mathrm{Tr}_{L/F}(x)$ of x. We defined the algebra trace as a linear map, in this case from L to F:

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

This definition was 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 did not occur in our development.

We defined the *trace form* to be an F-bilinear form on L, mapping $x, y : L$ to $\mathrm{Tr}_{L/F}(xy)$.

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

In the following, let $E/L/F$ be a tower of finite extensions of fields, namely we assumed `[algebra E L] [algebra L F] [algebra E F] [is_scalar_tower E L F]`, as described in Section 3.2.

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

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

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

We formulated the lemma in terms of the power basis, since we needed to use it for $F(x)$ here and for an arbitrary finite separable extension L/F later in the proof.

The elements of `(pb.minpoly_gen.map (algebra_map F E)).roots` are called *conjugates* of x in E. 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` showed that the trace of x is an integer multiple (namely `findim F(x) L`) of a sum of conjugate roots, hence we concluded that the trace (and trace form) of an integral element is also integral.

Finally, we showed that the trace form is nondegenerate, following Neukirch [17, Proposition 2.8]. Since L/F 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 $E/L/F$, the main difficulty was in checking the equality $\sum_k x_k^{i+j} = \mathrm{Tr}_{L/F}(x^{i+j})$. Directly applying `trace_gen_eq_sum_roots` was tempting, since we had a sum over conjugates of powers on both sides. However, the two expressions did 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` resulted in a sum over conjugates of x^{i+j}.

Instead, the paper proof switched here to an equivalent definition of conjugate: the conjugates of x in E are the images (counted with multiplicity) of x under each embedding $\sigma\colon F(x) \to E$ that fixes F. 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 gave $\mathrm{Tr}_{L/F}(x) = \sum_\sigma \sigma x$. Since each 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, which concluded the proof.

7 Class group and class number

Given a Dedekind domain with fraction map $f\colon D \to F$, we formalized the notion of class group in Lean by defining a map `to_principal_ideal f:units f.codomain → units (fractional_ideal f)`, and defined the class group as

```
def class_group := quotient_group.quotient (to_principal_ideal (range f))
```

In general, Dedekind domains can have infinite class groups. However, as discussed in Section 2, the rings of integers of global fields have finite class groups.

We let K be a number field and K' be a function field, with ring of integers $\mathcal{O}_K$ and $\mathcal{O}_{K'}$ (w.r.t. a fixed $\mathbb{F}_q[t]$), respectively. Most proofs of the finiteness of $\mathcal{Cl}_{\mathcal{O}_K}$ one finds in a modern textbook (see [17, Theorems 4.4, 5.3, 6.3]) depend on Minkowski's lattice point theorem, a result from the geometry of numbers (which has been formalized in Isabelle/HOL [8]). Extending this proof to show the finiteness of $\mathcal{Cl}_{\mathcal{O}_{K'}}$ is quite involved and does not result in a uniform proof for $\mathcal{Cl}_{\mathcal{O}_K}$ and $\mathcal{Cl}_{\mathcal{O}_{K'}}$. Our formalization adapted and generalized a classical approach to the finiteness of $\mathcal{Cl}_{\mathcal{O}_K}$, where the use of Minkowski's theorem is replaced by the pigeonhole principle. For an informal writeup of the proof, used in the formalization efforts, see `https://github.com/lean-forward/class-number/blob/main/FiniteClassGroup.pdf`. The classical approach seems to go back to Kronecker and can be found, for instance, in [14]. We note that some other "uniform" approaches can be found in [1] and [19].

Let D be an Euclidean domain: in particular, it will be a PID and hence a Dedekind domain. Given a fraction map $f\colon D \to F$, let L be a finite separable field extension of F. We formalized, in the theorem `class_group.finite_of_admissible`, that the integral closure of D in L has a finite class group if D has an "admissible" absolute value `abs`. Very informally, the admissibility conditions require that the remainder operator produces values that are not too far apart. Formally, we defined the type of admissible absolute values on D as follows, where `to_fun` stands for an application of the absolute value operator:

```
structure admissible_absolute_value (D : Type*) [euclidean_domain D]
  extends euclidean_absolute_value D ℤ :=
(card : ℝ → ℕ) (exists_partition :
  ∀ (n : ℕ) (ε > (0 : ℝ)) (b ≠ (0 : D)) (A : fin n → D),
  ∃ (t : fin n → fin (card ε)), ∀ i₀ i₁, t i₀ = t i₁ →
  (to_fun (A i₁ % b - A i₀ % b) : ℝ) < to_fun b · ε)
```

The above condition formalizes and generalizes an intermediate result in paper finiteness proofs; the different proofs for number fields and function fields (still assuming L/F separable) become the same after this point. We used division with remainder to replace the *fractional part* operator on F in the classical proof, which was essential to incorporate function fields, and at the same time allowing our proof to stay entirely within D to avoid coercions.

The absolute value extends to a norm `abs_norm f abs:integral_closure D L → ℤ`. We used the admissibility of `abs` to find a finite set `finset_approx L f abs` of elements of D, such that the following generalization of [14, Theorem 12.2.1] holds.

```
theorem exists_mem_finset_approx' (a b : integral_closure D L) :=
  ∃ (q : integral_closure D L) (r ∈ finset_approx L f abs),
  abs_norm f abs (r · a - q * b) < abs_norm f abs b
```

After this, the classical approach mentioned above formalized smoothly.

It remained to define an admissible absolute value for $\mathbb{Z}$ and $\mathbb{F}_q[t]$. On $\mathbb{Z}$, it was straightforward to formalize that the usual Archimedean absolute value fulfils the requirements. For $\mathbb{F}_q[t]$, we showed that $|f|_{\deg} := q^{\deg f}$ for $f \in \mathbb{F}_q[t]$ is the required admissible absolute value; observe that this was somewhat more involved to formalize. We concluded that when K is a global field, restricting to *separable* extensions of $\mathbb{F}_q(t)$ in the function field case, the class group is finite:

```
noncomputable instance : fintype
  (class_group (number_field.ring_of_integers.fraction_map K)) :=
class_group.finite_of_admissible K int.fraction_map int.admissible_abs

noncomputable instance [is_separable f.codomain K] : fintype
  (class_group (function_field.ring_of_integers.fraction_map f K)) :=
class_group.finite_of_admissible F f polynomial.admissible_card_pow_degree
```

Finally, we defined `number_field.class_number` and `function_field.class_number` as the cardinality of the respective class groups.

8 Discussion

8.1 Related work

Broadly speaking, one could see the formalization work as part of number theory. There are several formalization results in this direction. Most notably, Eberl formalized a substantial part of analytic number theory in Isabelle/HOL [9]. Narrowing somewhat to a more algebraic setting, we are not aware of any other formal developments of fractional ideals, Dedekind domains or class groups of global fields.

There are many libraries formalizing basic notions of commutative algebra such as field extensions and ideals, including the Mathematical Components library in Coq [15], the algebraic library for Isabelle/HOL [2], the `set.mm` database for MetaMath [16] 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, Blot [3] formalized algebraic numbers in Coq, Thiemann, Yamada and Joosten [22] in Isabelle/HOL, Carneiro [4] in MetaMath, and Watase [23] in Mizar. To our knowledge, the Coq Mathematical Components library is the only formal development beside ours specifically dealing with number fields [15, `field/algnum.v`].

Apart from the general theory of algebraic numbers, there are formalizations of specific rings of integers. For instance, the Gaussian integers $\mathbb{Z}[i]$ have been formalized in Isabelle/HOL by Eberl [10], in MetaMath by Carneiro [5] and in Mizar by Futa, Mizushima, and Okazaki [12].

Eberl's Isabelle/HOL formalization deserves special mention in this context 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 various basic results of algebraic number theory, there are several natural directions for future work, including formalizing some of the following results.

- Finiteness of the class group for rings of integers in all global fields. This would entail, apart from some details at the end of the proof, dropping the separability condition in the result mentioned in the first paragraph of Section 6.
- The group of units of the ring of integers in a number field is finitely generated, or slightly stronger, Dirichlet's unit theorem [17, Theorem 7.4] (and the function field analogue).
- Other finiteness results in algebraic number theory, most notably Hermite's theorem about the existence of finitely many number fields, up to isomorphism, with bounded discriminant [17, Theorem 2.16] (and the function field analogue).
- Class number computations, say of quadratic number fields. This could be part of verifying correctness of number theoretic software, such as KASH/KANT [18] and PARI/GP [21].
- Applications of algebraic number theory to solving Diophantine equations, such as determining all pairs of integers (x, y) such that $y^2 = x^3 + D$ for given nonzero $D \in \mathbb{Z}$.

8.3 Conclusion

In this project, we confirmed the rule that the hardest part of formalization is to get the definitions right. Once this is accomplished, the paper proof (sometimes first adapted with formalization in mind) almost always translates into a formal proof without too much effort. In particular, we regularly had to invent abstractions to treat instances of the "same" situation uniformly. Instead of fixing a canonical representation, be it $K \subseteq L \subseteq F$ as subfields or the field of fractions $\operatorname{Frac} R$, or the monogenic $K(\alpha)$, we found that making the essence of the situation an explicit parameter, as in `is_scalar_tower`, `fraction_map` or `power_basis`, allows to treat 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 that 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 E. Artin and G. Whaples. Axiomatic characterization of fields by the product formula for valuations. *Bull. Amer. Math. Soc.*, 51(7):469–492, 07 1945. URL: `https://projecteuclid.org:443/euclid.bams/1183507128`.

2 C. Ballarin editor, J. Aransay, M. Baillon, P. E. de Vilhena, S. Hohe, F. Kammüller, and L. C. Paulson. The Isabelle/HOL algebra library. `http://isabelle.in.tum.de/dist/library/HOL/HOL-Algebra/index.html`.

3 Valentin Blot. Basics for algebraic numbers and a proof of Liouville's theorem in C-CoRN, 2009. MSc internship report.

4 M. Carneiro. Definition `df-aa`. `http://us.metamath.org/mpeuni/df-aa.html`.

5 M. Carneiro. Definition `df-gz`. `http://us.metamath.org/mpeuni/df-gz.html`.

6 L. de Moura, S. Kong, J. Avigad, F. van Doorn, and J. 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`.

7 D. S. Dummit and R. M. Foote. *Abstract algebra*. John Wiley & Sons, Inc., Hoboken, NJ, third edition, 2004.

8 M. Eberl. Minkowski's theorem. *Archive of Formal Proofs*, July 2017. `https://isa-afp.org/entries/Minkowskis_Theorem.html`, Formal proof development.

9 M. Eberl. Nine chapters of analytic number theory in Isabelle/HOL. In J. Harrison, J. O'Leary, and A. Tolmach, editors, *ITP 2019*, volume 141 of *LIPIcs*, pages 16:1–16:19. Schloss Dagstuhl, Leibniz-Zentrum fuer Informatik, 2019. `doi:10.4230/LIPIcs.ITP.2019.16`.

10 M. Eberl. Gaussian integers. *Archive of Formal Proofs*, April 2020. `https://isa-afp.org/entries/Gaussian_Integers.html`, Formal proof development.

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 A. Grabowski, A. Kornilowicz, and C. 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 K. Ireland and M. Roosen. *A Classical Introduction to Modern Number Theory*. Springer-Verlag New York, second edition, 1990.

15 A. Mahboubi and E. Tassi. *The Mathematical Components Libraries*. `https://math-comp.github.io/mcb/`, 2017.

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

17 J. Neukirch. *Algebraic number theory*, volume 322 of *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`.

18 M. E. et al Pohst. The computer algebra system KASH/KANT. `http://www.math.tu-berlin.de/~kant`.

19 A. Stasinski. A uniform proof of the finiteness of the class group of a global field. *to appear in Amer. Math. Monthly*. URL: `https://arxiv.org/abs/1909.07121`.

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

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

22 R. Thiemann, A. Yamada, and S. Joosten. Algebraic numbers in Isabelle/HOL. *Archive of Formal Proofs*, December 2015. `https://isa-afp.org/entries/Algebraic_Numbers.html`, Formal proof development.

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