A formal proof of Hensel's lemma over the p-adic integers

Robert Y. Lewis
Vrije Universiteit Amsterdam
r.y.lewis@vu.nl

Abstract

The field of p-adic numbers $\mathbb{Q}_p$ and the ring of p-adic integers $\mathbb{Z}_p$ are essential constructions of modern number theory. Hensel's lemma, described by Gouvêa as the "most important algebraic property of the p-adic numbers," shows the existence of roots of polynomials over $\mathbb{Z}_p$ provided an initial seed point. The theorem can be proved for the p-adics with significantly weaker hypotheses than for general rings. We construct $\mathbb{Q}_p$ and $\mathbb{Z}_p$ in the Lean proof assistant, with various associated algebraic properties, and formally prove a strong form of Hensel's lemma. The proof lies at the intersection of algebraic and analytic reasoning and demonstrates how the Lean mathematical library handles such a heterogeneous topic.

Keywords Lean, p-adic, Hensel's lemma, formal proof

1 Introduction

It has long been a goal of the formalized mathematics community to verify the typical undergraduate mathematics curriculum in a unified way using a single proof assistant. Various systems have achieved this goal to varying degrees [3, 23, 31, 35, 36], and it now seems reasonable to say that most components of this curriculum have been formalized in one system or another. Undeniably, though, some fields have received much heavier attention than others; in particular, it appears that formal developments in number theory and geometry have lagged behind those in other domains. This imbalance becomes even greater when one looks beyond undergraduate mathematics. While a few landmark projects have verified deep mathematical results [18, 19, 22], the associated theory developments have been thin and specific to the target theorems. Research-level *theorems* have been formalized, but research-level *theories* remain largely untouched.

A recent project begun at the Vrije Universiteit Amsterdam aims to address this imbalance by bringing together traditional mathematicians, formalizers, and tool developers to work toward modern results in number theory.[1] With researchers working at all levels of the theorem proving pipeline, the project will search for technology and design decisions that make it plausible to formalize deep mathematics that spans across fields. This paper breaks some initial ground to build toward this goal by constructing the p-adic numbers $\mathbb{Q}_p$ and the p-adic integers $\mathbb{Z}_p$ in the Lean proof assistant and verifying Hensel's lemma, a foundational result about these numbers.

The p-adics are a fundamental object of study in number theory with both theoretic and numeric applications. Their construction involves a mix of analytic and algebraic methods. For this reason, they make an excellent (or even necessary) point from which to embark on a project to formalize number theory. Hensel's lemma, an analogue of Newton's method for approximating roots, holds a very prominent place in the study of the p-adics. Its computational applications make it of interest to number theorists and computer scientists alike.

In Section 2, we give an informal overview of the p-adic numbers and Hensel's lemma, outlining the construction and proof followed in our formalization. Section 3 briefly describes the Lean theorem prover and the mathematical library on which this project depends. Sections 4 and 5 explain the formal construction of $\mathbb{Q}_p$ and $\mathbb{Z}_p$ and the formal proof of Hensel's lemma, respectively, focusing on design decisions made during the formalization process. In Sections 6 and 7, we consider related work and reflect on the project.

The formalization described in this paper is incorporated into the Lean mathematical library, available on GitHub.[2] Since this library is regularly changing, we preserve a snapshot of its status at the time this paper was submitted. This snapshot, and a map between this paper and the formalization, can be found on the author's website.[3] The code blocks presented in this paper should be read as schematic, not literal: we sometimes change names, omit universe levels, and swap implicit and explicit arguments for the sake of presentation.

2 The p-adic numbers and Hensel's lemma

Readers who have seen the construction of the real numbers $\mathbb{R}$ via Cauchy sequences of rationals will find the construction of $\mathbb{Q}_p$ familiar. To motivate stepping from $\mathbb{Q}$ to $\mathbb{R}$, we traditionally point to the fact that $\mathbb{Q}$ is *incomplete*: there are sequences of rational numbers that seem to approach a value, and yet no value exists at the limit. As an example, consider

[1] https://lean-forward.github.io/

[2] https://github.com/leanprover/mathlib/

[3] https://robertylewis.com/padics/

CPP'19, January 14–15, 2019, Lisbon, PT
2019.

```
  1111111          1212121          1111111
...44444444      ...31313132      ...31313132
+         1    ×         3    +...44444444
----------       ----------       ----------
          0                1      ...31313131
```

Figure 1. If we represent $\mathbb{Q}_p$ as left-infinite streams of digits, we can perform addition and multiplication in base p by carrying remainders to the left. 5-adically, $\ldots 444444 + 1 = 0$ and $\ldots 313132 \times 3 = 1$. The former means that adding $\ldots 444444$ to any number is the same as subtracting 1. All of these numbers (besides 0) have 5-adic norm equal to 1.

the sequence $r = (1, 1.4, 1.41, 1.414, 1.4142, \ldots)$ whose nth entry is the greatest $(n+1)$-digit rational number whose square is less than 2. This sequence has no limit in $\mathbb{Q}$, since $\sqrt{2}$ is irrational. By creating points to represent the limit of r and similarly "convergent" sequences, we obtain $\mathbb{R}$.

More precisely, we say that a sequence $s : \mathbb{N} \to \mathbb{Q}$ is *Cauchy* if for every positive $\epsilon \in \mathbb{Q}$, there exists a number N such that for all $k \geq N$, $|s_N - s_k| < \epsilon$. Two sequences s and t are *equivalent*, written $s \sim t$, if for every positive $\epsilon \in \mathbb{Q}$, there exists an N such that for all $k \geq N$, $|s_k - t_k| < \epsilon$. Morally, a Cauchy sequence is one whose terms eventually get arbitrarily close to each other, and "should" converge to a (possibly irrational) value; two Cauchy sequences are equivalent if they "should" converge to the same value. $\mathbb{R}$ is defined to be the quotient of the set of Cauchy sequences with respect to $\sim$, which is to say that a real number is a set of equivalent Cauchy sequences. It can be shown that Cauchy sequences inherit the field operations from $\mathbb{Q}$, and that these operations respect $\sim$, so they can be lifted to the quotient.

We often call $\mathbb{R}$ the *completion* of $\mathbb{Q}$ because it is the smallest extension of $\mathbb{Q}$ that fills in the missing values. But it is more accurate to say that $\mathbb{R}$ is *a* completion of $\mathbb{Q}$, since the notion of a Cauchy sequence is parametrized by a notion of closeness. A function f on $\mathbb{Q}$ is a (generic) absolute value if it is positive-definite ($f(0) = 0$ and $f(k) > 0$ otherwise), subadditive ($f(x + y) \leq f(x) + f(y)$), and multiplicative ($f(x \cdot y) = f(x) \cdot f(y)$). We can replay the construction above, replacing the standard absolute value on $\mathbb{Q}$ with any generic absolute value. If we use the trivial absolute value $a(x) = (0 \texttt{ if } x = 0 \texttt{ else } 1)$, then a sequence of rationals is Cauchy if and only if it is eventually constant, and so the completion process adds no new points to $\mathbb{Q}$.

There are infinitely many absolute values on $\mathbb{Q}$, even identifying scalar multiples: every prime number induces a unique absolute value. Fix $p \in \mathbb{N}$ with $p > 1$, and for $z \in \mathbb{Z}$ with $z \neq 0$, define the *p-adic valuation* $v_p(z)$ to be the largest n such that $p^n | z$. This valuation extends to $\mathbb{Q}$ by setting $v_p(q/r) = v_p(q) - v_p(r)$, where q and r are coprime. The *p-adic norm* on $\mathbb{Q}$ is defined by $|x|_p = p^{-v_p(x)}$ with $|0|_p = 0$. If p is prime, this function is an absolute value. Surprisingly, this exhausts the list of possibilities. Ostrowski's theorem [37] proves that any absolute value on $\mathbb{Q}$ is (a positive scalar power of) either the standard absolute value, the trivial absolute value, or the p-adic norm for some prime p. The *p-adic numbers* $\mathbb{Q}_p$ are the completion of $\mathbb{Q}$ with respect to the p-adic norm. Since a positive scalar power of an absolute value induces an isomorphic completion, we can focus our attention on $\mathbb{R}$ and the family $\{\mathbb{Q}_p \mid p \text{ prime}\}$.

To get an intuition for how the p-adic numbers behave, consider what it means for two rationals to be close under the p-adic norm. A rational x has small p-adic norm if $p^{-v_p(x)}$ is small, that is, if $v_p(x)$ is large, which means that x is divisible by a large power of p. Thus $|x - y|_p$ is small if x and y are separated by a multiple of p^k for some large k. The elements of a Cauchy sequence, then, are separated by multiples of larger and larger powers of p.

The traditional decimal expansion allows us to write any nonzero real number in the form $\pm\Sigma_{i=-\infty}^{k} a_i \cdot 10^i$, where k is a (finite, possibly negative) integer and each $a_i \in \{0, 1, \ldots, 9\}$ with $a_k \neq 0$. We can analogously define a p-adic expansion allowing us to write any nonzero element of $\mathbb{Q}_p$ in the form $\Sigma_{i=k}^{\infty} a_i \cdot p^i$ (Figure 1). Such a series does not necessarily converge to a real number, but it does converge under the p-adic norm. For $m \geq n$, the difference between the mth and nth partial sums is divisible by p^n, so its norm is bounded by p^{-n} which tends to 0 as n grows.

The p-adic norm on $\mathbb{Q}$ extends to a norm on $\mathbb{Q}_p$, such that $|\Sigma_{i=k}^{\infty} a_i \cdot p^i|_p = p^{-k}$. These norms share a useful (if counterintuitive) property. The familiar triangle inequality states that $|x + y| \leq |x| + |y|$ for any x and y. But we can show something stronger for the p-adic norm, namely the *nonarchimedean* property, which states that

$$|x + y|_p \leq \max(|x|_p, |y|_p).$$

In fact, if $|x|_p \neq |y|_p$, it holds that $|x + y|_p = \max(|x|_p, |y|_p)$. This is not hard to check over $\mathbb{Q}$ by unfolding definitions.

Under the standard absolute value, any Cauchy sequence of integers is eventually constant, but this is not the case under the p-adic norm. It can be shown that such a sequence converges to a p-adic number z with $|z|_p \leq 1$, and so we define the *p-adic integers* $\mathbb{Z}_p = \{z \in \mathbb{Q}_p \mid |z|_p \leq 1\}$ (Figure 2). Equivalently, a p-adic integer is a p-adic number whose p-adic expansion has no nonzero values to the right of the decimal point. Because of the nonarchimedean property, the sum of two p-adic integers is still an integer, and thus $\mathbb{Z}_p$ forms a ring.

As a complete structure with a nonarchimedean norm, $\mathbb{Q}_p$ is a natural setting to develop a theory of analysis. Many of the familiar notions of calculus over $\mathbb{R}$ are simplified in this setting. For instance, deciding whether an infinite series $\sum_{i=0}^{\infty} r_i$ over $\mathbb{R}$ converges can be a subtle problem, and calculus students traditionally learn a list of "convergence tests" to answer it. Over $\mathbb{Q}_p$, the nonarchimedean property guarantees that such a series converges if and only if $\lim_{i \to \infty} |r_i|_p = 0$.

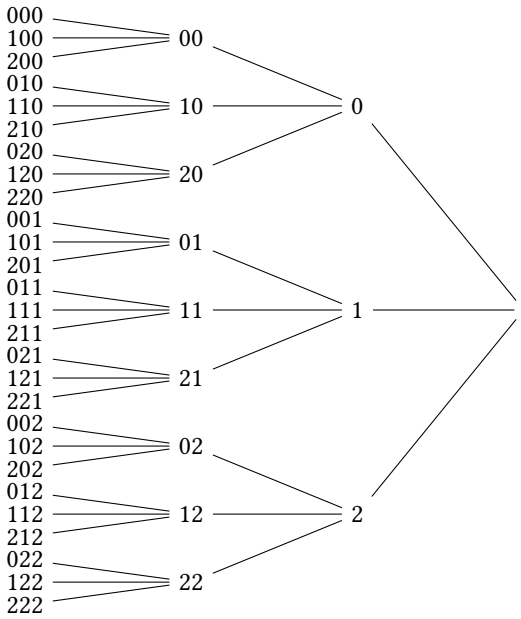

Figure 2. This diagram displays initial segments of elements of $\mathbb{Z}_3$. The 3-adic integers are the infinite continuation of this tree to the left. The distance between two 3-adic integers is determined by the depth of their first common ancestor, where deeper means closer. Thus, the open sets of the topology on $\mathbb{Z}_3$ are generated by the down(left)-sets of this infinite tree.

Mahler's theorem [32] gives a remarkably straightforward characterization of continuous functions on $\mathbb{Z}_p$.

Applications of p-adic analysis arise in many areas of number theory, including in the studies of Diophantine equations and arithmetic progressions [29]. In computer science, the p-adics can be used to implement efficient rational arithmetic [25]. The p-adic integers $\mathbb{Z}_p$ are particularly useful for establishing facts about divisibility and modularity. Just as analysis over $\mathbb{Q}_p$ is in some sense simpler than analysis over $\mathbb{R}$, the algebraic structure on $\mathbb{Z}_p$ makes these results comparatively easy to obtain. Another application is in the method of Chabauty-Coleman [34], which can often be used to determine rational points on algebraic varieties. This method is used in the resolution of certain generalized Fermat equations [13], closely related to the mathematics that the Lean Forward project will address.

Gouvêa [20] cites Hensel's lemma [26] as the "most important algebraic property of the p-adic numbers." This result, which establishes a connection between the number-theoretic properties and the analysis of polynomial functions over $\mathbb{Z}_p$, is the backbone of the study of the p-adics. It is often applied, in various contexts, to prove the (non)existence of solutions to polynomial equations over various rings; in computer science, it appears in floating point rounding algorithms. Hensel's lemma is stated in the literature in many forms. The central idea is that for any univariate polynomial f over $\mathbb{Z}_p$, if one can find a point a such that $f(a)$ and $f'(a)$ satisfy certain requirements, then f has a unique root within a neighborhood of a. (We state the hypotheses explicitly in Section 5.)

Hensel's lemma can be used to reduce the problem of finding roots of a polynomial over $\mathbb{Z}_p$ to the (finite) problem of finding roots over $\mathbb{Z}/p\mathbb{Z}$. The *local-global principle*, also known as the Hasse or Hasse-Minkowski principle, is one of the central principles of Diophantine geometry [39]. It describes a general system for translating questions about roots over $\mathbb{Q}$ to questions about roots over $\mathbb{Q}_p$, which are often easier to answer. A striking application of this principle shows that a quadratic form over $\mathbb{Q}$ has nontrivial roots in $\mathbb{Q}$ if and only if it has nontrivial roots in $\mathbb{R}$ and $\mathbb{Q}_p$ for all prime p. The scope and applications of the local-global principle are actively explored in number theory today; Browning [8] gives a survey of recent results.

3 The Lean mathematical library

The Lean proof assistant, developed principally by Leonardo de Moura, was first released in 2014 [15]. Lean implements a version of the calculus of inductive constructions (CIC) [12] with support for quotient types and classical reasoning. Since the release of the most recent version in 2017 [14], there has been a concerted effort to develop a comprehensive library for use in mathematics and computer science [10].

Lean's mathlib is younger and smaller than similar libraries in other systems, such as Coq's Mathematical Components [31] or Isabelle's Archive of Formal Proofs [36], but it contains developments in many important areas of mathematics. Some noteworthy results that have been formalized include a proof of the law of quadratic reciprocity, a model of ZFC, and the construction of the Lebesgue measure on $\mathbb{R}$.

The datatypes available in mathlib include the concrete types commonly found in mathematics, among them $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{R}$, and $\mathbb{C}$; finite sets and multisets over a base type; and embeddings and isomorphisms between types. The algebraic hierarchy of mathlib is designed using *type classes*, which endow a base type with extra structure in the forms of operations, properties, and notation [40, 43]. Lean's type class resolution mechanism automatically manages inheritance between type classes (Figure 3). If a type class `T'` extends (directly or by transitivity) a type class `T`, any theorem proved over `T` will apply to any type that instantiates `T'`. The algebraic hierarchy begins with semigroups and monoids and extends to rich structures including fields, Noetherian rings, and principal ideal domains. Van Doorn, von Raumer, and Buchholz [41] give a more detailed explanation of how type classes are used to define an algebraic hierarchy in Lean.

Topological structure is also managed using type classes. In particular, topologies on metric spaces, normed spaces,

```
class semigroup (α : Type) extends has_mul α :=
(mul_assoc : ∀ a b c : α, a * b * c = a * (b * c))

class monoid (α : Type) extends semigroup α,
  has_one α :=
(one_mul : ∀ a : α, 1 * a = a)
(mul_one : ∀ a : α, a * 1 = a)

class group (α : Type) extends monoid α,
  has_inv α :=
(mul_left_inv : ∀ a : α, a⁻¹ * a = 1)

lemma one_inv (α : Type) [group α] :
  1⁻¹ = (1 : α) :=
inv_eq_of_mul_eq_one (one_mul 1)
```

Figure 3. A sample of the bottom of the algebraic hierarchy. The lemma `one_inv` can be applied to any `α` for which Lean can infer an instance of `group α`.

and similar structures are inherited from the topology defined on uniform spaces [7], of which all of these structures are instances. Topological notions such as limits and continuity are defined using filters [27], which specialize to more familiar definitions on metric or normed spaces.

In contrast to many other libraries for CIC-based systems, mathlib does not focus on constructive mathematics. Most of the core datatypes are defined computably, making them able to be reduced in the kernel or virtual machine. But the more abstract mathematical theories freely use classical logic; these theories are mostly noncomputable. Since the system can easily track the computability of a declaration, terms that do not depend on additional axioms will still compute.

Lean features a powerful *metaprogramming* framework that allows users to write custom tactics in the language of Lean itself [17]. There are a number of such tactics included in mathlib. Relevant to this project are `linarith`, which proves linear inequality goals using certified Fourier-Motzkin elimination; `ring`, a tactic based on Gregoire and Mahboubi's work in Coq [21] which normalizes expressions in the language of (semi)rings; and `wlog`, which reduces symmetric goals to a single case.

The development described in this paper uses a large portion of mathlib. In particular, it makes use of the concrete datatypes $\mathbb{Q}$ and $\mathbb{Z}$, along with many lemmas concerning divisibility and modular arithmetic; the topology library, for properties about continuity and limits; the analysis library, for the definitions of normed rings and fields and the topological properties of these structures; the abstract algebra library, to derive additional algebraic structure on $\mathbb{Z}_p$; and the polynomial library, which is needed even to state Hensel's lemma. This project has led to contributions to mathlib in all of these domains.

Readers unused to Lean syntax should note that explicit arguments to declarations are enclosed in parentheses `()`, implicit arguments are enclosed in curly brackets `{}`, and type class arguments are enclosed in square brackets `[]`. Only explicit arguments are given by the user when applying a declaration. For instance, writing a theorem as

```
lemma one_mul {α : Type} [group α] (a : α) : ...
```

specifies that the type `α` is supposed to be inferred automatically (say, from the argument `a`). The group structure on `α`, which is introduced anonymously, should be inferred by type class resolution. In the context `z : ℤ`, Lean will confirm that `one_mul z` is a proof that `1 * z = z`.

Another important feature of Lean syntax is its projection notation. Suppose `S` is a structure (or record) type with a field `val`, and `t : S`. The typical way to access the `val` field of `t` is by `S.val t`; here `S.val` is a compound name, with `val` living in the namespace `S`. Lean also admits the abbreviation `t.val`, using the period to separate a term and a name. This notation is not restricted to projections, although it is most commonly used there. In general, if a term named `T.op` has been defined and `t : T`, then `t.op` abbreviates `T.op` where `t` is inserted as the first argument of type `T`. For a concrete example, consider the type `polynomial α` and the operator

```
polynomial.eval : α → polynomial α → α
```

which evaluates a univariate polynomial at an argument. If we have `F : polynomial α` and `a : α`, we can use the notation `F.eval a` in place of `polynomial.eval a F`. This notation can be nested, e.g. to replace

```
polynomial.eval a (polynomial.derivative F)
```

with `F.derivative.eval a`.

4 Formalizing the p-adic numbers

In this section we describe the formal construction of $\mathbb{Q}_p$ and $\mathbb{Z}_p$ and the proofs of their associated algebraic properties. We approximately follow the presentation from Gouvêa [20], although many of the ideas here are canonical in the mathematical literature. Broadly, our construction goes by the following plan:

1. Define the p-adic valuation on $\mathbb{Z}$, extend it to $\mathbb{Q}$, and use this to define the p-adic norm.
2. Show that the p-adic norm on $\mathbb{Q}$ is a non-archimedean absolute value.
3. Define $\mathbb{Q}_p$ as the completion of $\mathbb{Q}$ with respect to the p-adic norm.
4. Show that $\mathbb{Q}_p$ inherits field operations and a norm from $\mathbb{Q}$.
5. Define $\mathbb{Z}_p$ as a subtype of $\mathbb{Q}_p$, and show that it instantiates various algebraic structures.

Throughout this development, we will fix a natural number p. Some proofs in step 2 assume only that $p > 1$. In the rest of the development, we work under the assumption that

p is prime. We manage this primality assumption using type classes, so such arguments never need to be given explicitly. In the code snippets below, we typically assume that these arguments have been fixed as parameters, and only include them in the signatures of our functions when we wish to highlight them.

The valuation and norm functions defined in step 1 are total: instead of taking proofs (e.g. that p is prime) as arguments, they return the value 0 when their arguments are not in the intended domain. This approach to defining partial functions is common in logics that support only total functions. Proofs of properties of these functions assume that the arguments are in the intended domain; these proofs are often inferred by the type class mechanism and are thus transparent to the user.

4.1 The p-adic valuation and norm on $\mathbb{Q}$

The p-adic valuation $v_p(z)$ of an integer $z \neq 0$ is the largest k such that $p^k | z$. This extends to $\mathbb{Q}$ by setting $v_p(q/r) = v_p(q) - v_p(r)$ when q and r are coprime. We define these functions in Lean using `nat.find_greatest P b`, which returns the greatest `n` $\leq$ `b` satisfying the predicate `P`. Recall that `z.nat_abs`, `q.num`, and `q.denom` are projection notation for `int.nat_abs z`, `rat.num q`, and `rat.denom q` respectively.

```
def padic_val (p : ℕ) (z : ℤ) : ℕ :=
if z = 0 then 0
else if p > 1 then
  nat.find_greatest (λ k, (p ^ k) | z) z.nat_abs
else 0

def padic_val_rat (p : ℕ) (q : ℚ) : ℤ :=
(padic_val p q.num : ℤ)-(padic_val p q.denom : ℤ)

def padic_norm (p : ℕ) (q : ℚ) : ℚ :=
if q = 0 then 0
else (p : ℚ)^(-(padic_val_rat p q))
```

These definitions are computable and can thus be evaluated on closed inputs. Note that `padic_val` and `padic_norm` both require the natural number `p` as an explicit argument. In general, `p` cannot be inferred from context. This makes it difficult to introduce generic notation for these functions, or to use $\mathbb{Q}$ to instantiate type classes that depend on the norm, such as `normed_field`. This complication will be resolved once we define $\mathbb{Q}_p$, since `p` will be an argument to the type of p-adic numbers.

4.2 Properties of the p-adic norm

Proving the essential properties of v_p is similarly straightforward, under the assumption that $p > 1$. The only lemmas that require p to be prime are the multiplicative properties, e.g.:

```
lemma mul {m n : ℤ} (hm : m ≠ 0) (hn : n ≠ 0) :
 padic_val p (m*n) = padic_val p m + padic_val p n
```

For the most part, properties of `padic_norm` follow from analogous properties of `padic_val_rat`, which themselves follow from analogous properties of `padic_val`. Lifting proofs requires some care with casts between $\mathbb{N}$, $\mathbb{Z}$, and $\mathbb{Q}$.

The most involved proof in this section is the core of the later proof that the p-adic norm is nonarchimedean.

```
theorem min_le_padic_val_rat_add {q r : ℚ}
  (hq : q ≠ 0) (hr : r ≠ 0) (hqr : q + r ≠ 0) :
  min (padic_val_rat p q) (padic_val_rat p r)
    ≤ padic_val_rat p (q + r)
```

Proving this fact requires an elementary but subtle computation. Once it is completed, the proof that `padic_norm p` instantiates the `is_absolute_value` type class follows quickly. This instance depends on the primality of p, which is inferred by type class resolution.

4.3 Completing $\mathbb{Q}$

There are many related notions of Cauchy completions in the mathematical literature, varying in the level of abstraction and in the structure on the base space. We considered a number of options for constructing $\mathbb{Q}_p$.

The first and most generic option was to perform the *uniform completion* of $\mathbb{Q}$ with respect to the uniform structure generated by the p-adic norm [28]. A uniform space is an abstraction that falls somewhere in between a metric space, in which every two points are separated by a real-valued distance, and a topological space, which provides a generic but unquantified notion of "separatedness." A uniform structure allows one to consider relative distances between points without assigning concrete values to these distances. Any uniform space α can be completed by considering the space of Cauchy filters over α, where a Cauchy filter is a topological generalization of a Cauchy sequence. When the uniform structure on α is induced by a metric (or norm), this construction reduces to the completion described in Section 2.

The uniform completion process has been formalized in Lean and could, in principle, be immediately specialized to obtain $\mathbb{Q}_p$. However, the generality of this construction is not so amenable to a "concrete" number type like $\mathbb{Q}_p$. It is quite difficult to lift operations that are not uniformly continuous, such as multiplication, from the base type to the completion space. Furthermore, one must reconcile the filter-based notion of completeness with a sequential notion in order to prove Hensel's lemma, which depends on finding a limit of a sequence of p-adic integers. These complications created by the generality of the uniform completion came with few upsides; it seemed more prudent to take a different approach. Regardless of the initial construction, $\mathbb{Q}_p$ is easily instantiated as a complete uniform space after the fact.

A second option was to specialize the uniform completion to the completion of a normed structure. (We rejected the idea of using the metric completion, which falls in between,

```
class is_absolute_value {α} [ordered_field α]
  {β} [ring β] (f : β → α) : Prop :=
(abv_nonneg : ∀ x, 0 ≤ f x)
(abv_eq_zero : ∀ {x}, f x = 0 ↔ x = 0)
(abv_add : ∀ x y, f (x + y) ≤ f x + f y)
(abv_mul : ∀ x y, f (x * y) = f x * f y)

parameters {α : Type} [comm_ring α]
parameters (β : Type) [ordered_field β]
parameters (abv : α → β) [is_absolute_value abv]

def cau_seq : Type :=
{f : ℕ → α // is_cauchy abv f}

def equiv (f g : cau_seq) : Prop :=
∀ ε > 0, ∃ i, ∀ j ≥ i, abv (f j - g j) < ε

def completion : Type := quotient cau_seq equiv

instance : comm_ring completion := ...
```

Figure 4. A schematic depiction of the general Cauchy sequence completion. In Lean, *parameters* are automatically included as arguments to declarations throughout the duration of a section.

since it comes with all the disadvantages of the norm completion and more.) Under this specialization, the interface for lifting operations looks more familiar. Norm completions are not uncommon in mathematics, so although implementing an interface for this would take some initial effort, it could be reused in the future.

Two downsides discouraged us from this approach. First, the norm referred to in an arbitrary normed space is typically real-valued. Defining $\mathbb{Q}_p$ would thus depend on $\mathbb{R}$, which is already a completion of $\mathbb{Q}$ (using a different completion process). While logically sound, this approach would be pedagogically dubious, since it removes the direct analogy between $\mathbb{Q}_p$ and $\mathbb{R}$; it would also obscure the fact that the p-adic norm only takes rational values. A second, more practical, concern with this approach was related to the necessary type class inference. The norm on a space is generally inferred automatically, and the default instance for $\mathbb{Q}$ is the traditional absolute value. To use the p-adic norm instead, we would have to locally override this instance. Doing this would be possible but could lead to complications in future developments that use the p-adic norm and traditional absolute value on $\mathbb{Q}$ simultaneously.

The option we elected to follow is to directly define the Cauchy sequence completion of a type α, with respect to an absolute value on α taking values in an arbitrary ordered field (Figure 4). If α has a ring or field structure, this structure lifts immediately to the completion. This approach generalizes the former definition of $\mathbb{R}$ in mathlib; we implement both $\mathbb{R}$ and $\mathbb{Q}_p$ as instances of this general construction. It has the added benefit that the ring operations on $\mathbb{Q}_p$ are computable, although this property is not used in the current development. The downside to this approach is that some work must be done to connect the concrete ϵ-definition of convergence to more general topological notions. This extra work can be minimized by instantiating $\mathbb{Q}_p$ as a normed field, which we do in step 4.

```
def padic (p : ℕ) [prime p] : Type :=
completion ℚ (padic_norm p)

notation ℚ_[p] := padic p
```

The `completion` function takes two explicit arguments, a type and a field-valued function on that type. An implicit argument, inferred by type class resolution, shows that the field-valued function is an absolute value. Since the p-adic norm is only an absolute value if p is prime, it is essential to assume primality in the definition of $\mathbb{Q}_p$. This hypothesis is also an implicit type class argument and normally will be inferred automatically.

4.4 Operations on $\mathbb{Q}_p$

The field operations on $\mathbb{Q}_p$ are obtained for free through the Cauchy sequence construction. It also follows directly that $\mathbb{Q}$ is embedded in $\mathbb{Q}_p$. (Any constant sequence of rationals is Cauchy and is not equivalent to any other constant sequence of rationals, so each rational q induces a unique p-adic number $\bar{q}$.)

It takes more effort to lift the p-adic norm on $\mathbb{Q}$ to a norm on $\mathbb{Q}_p$. Intuitively, one might be tempted to claim that "the norm of the limit is the limit of the norms," that is, that we should define the norm of a Cauchy sequence to be the limit of the norms of each entry. But since the p-adic norm is rational-valued, and $\mathbb{Q}$ is not complete, this would unhelpfully produce a $\mathbb{Q}_p$-valued norm. Note the contrast with the real numbers: because the linear order on $\mathbb{Q}$ lifts to $\mathbb{R}$, a real-valued norm makes sense. Instead, we exploit an important property of the p-adic norm, derived from the fact that its values lie in the set $\{0\} \cup \{p^k \mid k \in \mathbb{Z}\}$. Because these values are separated (except at 0), the norms of the entries of any (nonzero) Cauchy sequence are eventually constant.

```
lemma stationary {f : cau_seq ℚ (padic_norm p)}
  (hf : ¬ f ≈ 0) : ∃ N, ∀ m n, m ≥ N → n ≥ N →
  padic_norm p (f n) = padic_norm p (f m)

def norm (f : cau_seq ℚ (padic_norm p)) : ℚ :=
if hf : f ≈ 0 then 0
else padic_norm p (f (stationary_point hf))

def padic_norm_e : ℚ_[p] → ℚ :=
quotient.lift norm norm_respects_equiv
```

Thus, the limit is indeed rational valued, and we can define a rational-valued norm on the type of Cauchy sequences.

This norm respects equivalence, so it can be lifted to the quotient $\mathbb{Q}_p$. With some work, the norm can be shown to preserve the essential properties of the norm on $\mathbb{Q}$, including the nonarchimedean property. We can also check that the norm is indeed an extension of the norm on $\mathbb{Q}$, meaning that for any $q \in \mathbb{Q}$, $|\bar{q}|_p = |q|_p$.

Since we have defined $\mathbb{Q}_p$ as the completion of $\mathbb{Q}$ with respect to the p-adic norm, and the p-adic norm extends to $\mathbb{Q}_p$, it is important to check that $\mathbb{Q}_p$ is in fact complete with respect to its norm. We again highlight the difference here between $\mathbb{Q}_p$ and $\mathbb{R}$. Since the absolute value on $\mathbb{R}$ is real-valued, as opposed to rational-valued, the arguments that $\mathbb{Q}_p$ and $\mathbb{R}$ are complete differ in significant ways. We do not use a general proof to cover both cases, since despite some structural similarity, the generalization is rather convoluted. We also prove that $\mathbb{Q}$ is dense in $\mathbb{Q}_p$—meaning that every $q \in \mathbb{Q}_p$ is arbitrarily close to $\bar{r}$ for some $r \in \mathbb{Q}$—similarly to how we prove the analogous statement in $\mathbb{R}$, with a separate implementation to account for the different absolute value.

We have thus established that $\mathbb{Q}_p$ is a complete field, densely embedding $\mathbb{Q}$, with a nonarchimedean norm that extends the p-adic norm on $\mathbb{Q}$. These properties uniquely characterize $\mathbb{Q}_p$: any structure with these properties is isomorphic to $\mathbb{Q}_p$. (We have not yet formalized this statement.)

Finally, we instantiate $\mathbb{Q}_p$ as a normed field. From this instance, $\mathbb{Q}_p$ inherits a topology and uniform structure. The only complication, mentioned above, is that the generic norm of a normed ring is real-valued instead of rational-valued. But since the essential properties of the p-adic norm are already established, casting to $\mathbb{R}$ is less troublesome here; similarly, the pedagogical concerns about using $\mathbb{R}$ in the construction of $\mathbb{Q}_p$ are no longer relevant.

From the normed field instance, we inherit the generic notation ‖x‖ for the norm of a p-adic number x. Unlike for the p-adic norm on $\mathbb{Q}$, there is no ambiguity here about the parameter p, since it can be inferred from the type of x.

4.5 Defining $\mathbb{Z}_p$

The p-adic integers $\mathbb{Z}_p$ are traditionally defined as the subset $\{z \in \mathbb{Q}_p \mid |z|_p \leq 1\}$. This is equivalent to the completion of $\mathbb{Z}$ using the p-adic norm, but for formalization purposes, the former definition is much simpler.

```
def padic_int (p : ℕ) [prime p] : Type :=
{z : ℚ_[p] // ‖z‖ ≤ 1}

notation ℤ_[p] := padic_int p
```

The notation here is for Lean's subtype data structure, meaning that a term z : ℤ_[p] is a dependent pair of a term x : ℚ_[p] with a proof that ‖x‖ ≤ 1. Note that this is not a "strict" subtype, in the sense that the term z does not have type ℚ_[p]; rather z.val, the first projection of z, has this type. We can move between the two types with little friction by defining a coercion from ℤ_[p] to ℚ_[p]. However, it is still convenient to minimize this kind of context shift, as we will discuss in Section 5.

From the properties of the p-adic norm, we obtain that $\mathbb{Z}_p$ is closed under sums and products and show that it forms a subring of $\mathbb{Q}_p$. This subring has algebraic structure that make it a fruitful object of study. Most fundamentally, we instantiate $\mathbb{Z}_p$ as a normed commutative local ring with maximal ideal $\{x \in \mathbb{Z}_p \mid |x|_p < 1\}$. We also show that it complete—meaning that any Cauchy sequence of p-adic integers converges to a p-adic integer—and that it densely embeds $\mathbb{Z}$.

As with $\mathbb{Q}_p$, the topology on $\mathbb{Z}_p$ is inherited from its norm. The open sets are generated by the family of balls $\{x \in \mathbb{Z}_p \mid |z - x|_p < \epsilon\}$, ranging over $\epsilon \in \mathbb{R}_{>0}$ and $z \in \mathbb{Z}_p$.

5 Formalizing Hensel's lemma

Hensel's lemma establishes another fundamental algebraic property of $\mathbb{Z}_p$. This result provides simple criteria for locating p-adic integer roots of a polynomial; it is widely applied in p-adic analysis, and is also used in approximation algorithms in computer science [33]. The general notion of a Henselian local ring, defined to be a local ring for which Hensel's lemma holds, appears in algebraic geometry. Weaker analogues of Hensel's lemma hold over other structures, including the standard integers $\mathbb{Z}$, but the hypotheses of these analogues are harder to satisfy than those for $\mathbb{Z}_p$.

The formal proof of Hensel's lemma follows a writeup by Conrad [11]. Conrad's description is more concrete than Gouvêa's [20] and avoids unnecessary detours into the group $\mathbb{Z}/p\mathbb{Z}$, although the approaches are schematically identical. We slightly modify Conrad's proof to perform as much computation as possible inside $\mathbb{Z}_p$, without stepping into $\mathbb{Q}_p$.

Conrad [11, Theorem 4.1] states Hensel's lemma as follows. Here, $\mathbb{Z}_p[X]$ is the ring of univariate polynomials over a variable X with coefficients in $\mathbb{Z}_p$. The derivative f' is the formal polynomial derivative, which does not rely on the notion of a limit.

Theorem. *Suppose that $f(X) \in \mathbb{Z}_p[X]$ and $a \in \mathbb{Z}_p$ satisfy $|f(a)|_p < |f'(a)|_p^2$. There exists a unique $z \in \mathbb{Z}_p$ such that $f(z) = 0$ and $|z - a|_p < |f'(a)|_p$. Furthermore, it holds that $|z - a|_p = |f(a)|_p / |f'(a)|_p$ and $|f'(z)|_p = |f'(a)|_p$.*

Hensel's lemma is sometimes stated with the requirements $f(a) \equiv 0 \mod p$ and $f'(a) \not\equiv 0 \mod p$. This is a weaker corollary of what we state here. The statement we have proven in Lean is a direct translation of the stronger version.

```
theorem hensels_lemma {p : ℕ} [hp : prime p]
  {F : polynomial ℤ_[p]} {a : ℤ_[p]} :
  ‖F.eval a‖ < ‖F.derivative.eval a‖^2 →
  ∃ z : ℤ_[p], F.eval z = 0 ∧
    ‖z - a‖ < ‖F.derivative.eval a‖ ∧
    ‖z - a‖ = ‖F.eval a‖ / ‖F.derivative.eval a‖ ∧
    ‖F.derivative.eval z‖ = ‖F.derivative.eval a‖ ∧
    ∀ z' : ℤ_[p], F.eval z' = 0 →
      ‖z' - a‖ < ‖F.derivative.eval a‖ → z' = z
```

While Hensel's lemma can be proved in various different settings at different levels of generality, nearly all proofs follow the same approach: starting with the "seed point" a, they recursively define a sequence of approximations to the desired root, and argue that this sequence converges. This argument is typically seen as an analogy to Newton's method for finding roots of real functions.

Our proof goes by the following sketch:

1. Establish two generic polynomial identities that will be used at multiple points of the proof.
2. Define a constant, depending only on a, that will be used to bound various quantities.
3. Define a recursive sequence $\mathbb{N} \to \mathbb{Z}_p$, simultaneously proving bounds on the values of f along this sequence.
4. Show that this sequence is Cauchy.
5. Show that the limit of this Cauchy sequence has the desired properties, in particular that it is a root of f.
6. Show that this root is unique within a neighborhood of a.

This sketch comes directly from Conrad. Our approach diverges slightly in step 3, where we reconfigure the recursion to avoid unnecessary casts between $\mathbb{Z}_p$ and $\mathbb{Q}_p$. We also assume that $f(a) \neq 0$ for much of the proof, and handle this later as a (simple) degenerate case. Conrad does not make this special case explicit, but the argument fails at a crucial point if $f(a) = 0$.

5.1 Polynomial identities

Two polynomial identities are used in the proof to rewrite expressions into forms that we can more easily bound. These identities are not specific to the p-adic numbers.

The first identity allows us to separate components of $f(x)$ and $f'(x)$ from the expansion of $f(x + y)$.

```
lemma binom_exp (f : polynomial α) (x y : α) :
  ∃ k : α, f.eval (x + y) = f.eval x +
            (f.derivative.eval x) * y + k * y^2
```

This identity follows from a similar statement on commutative semirings.

```
def binom_exp' {α} [comm_semiring α] (x y : α) :
  ∀ n : ℕ, ∃ k : α,
  (x + y)^n = x^n + n*x^(n-1)*y + k * y^2
```

After inducting on n, this proof follows nearly automatically using Lean's `ring` tactic. The only manual input is the value to instantiate `k`. This value was computed using computer algebra software, and using a link to such software (e.g. [30]), even this step could potentially be automated.

The second identity shows that $x - y$ divides $f(x) - f(y)$.

```
lemma eval_sub (f : polynomial α) (x y : α) :
  ∃ z : α, f.eval x - f.eval y = z*(x - y)
```

This also follows from a similar algebraic statement, which is proved by induction and ring evaluation.

5.2 A bounding value

In the subsequent steps we will fix `F : polynomial ℤ_[p]` and `a : ℤ_[p]`, and assume that the inequality

```
‖F.eval a‖ < ‖F.derivative.eval a‖^2
```

holds. (These assumptions are taken as *parameters* in Lean, which are automatically inserted into declarations throughout the duration of a section.) To establish bounds on the terms of the sequence we will define in step 3, we define an auxiliary constant `T`.

```
def T : ℝ :=
‖(F.eval a).val / ((F.derivative.eval a).val)^2‖
```

The division must take place in $\mathbb{Q}_p$, since $\mathbb{Z}_p$ is not a field. However, our hypothesis guarantees that `T < 1`, so the quotient is in fact an integer. It is trivial to prove the following alternate characterization of `T` (which uses the norm on $\mathbb{Z}_p$), along with various simple facts about `T` that will be useful for establishing bounds.

```
lemma T_def :
  T = ‖F.eval a‖ / ‖F.derivative.eval a‖^2
```

5.3 Defining the Newton sequence

The core step of the proof of Hensel's lemma is to define a sequence $\{a_n\}$ of values that converge to the desired solution. The recursion is typically given by

$$a_0 = a$$

$$a_{n+1} = a_n - \frac{f(a_n)}{f'(a_n)}$$

in informal texts. But without further information, this sequence lives in $\mathbb{Q}_p$ instead of $\mathbb{Z}_p$; we must establish properties about $f(a_n)$ and $f'(a_n)$ before concluding that a_{n+1} is an integer. In an informal presentation, it is not a problem to first define the sequence in $\mathbb{Q}_p$ and show integrality afterward. But doing so in our formal development would introduce another layer of casts, one which we would prefer to avoid. We pay a cost to avoid it: the recursion to build $\{a_n\}$ must incorporate the properties needed to prove integrality, making it slightly clumsier.

We define our induction hypothesis as follows:

```
def ih (n : ℕ) (z : ℤ_[p]) : Prop :=
‖F.derivative.eval z‖ = ‖F.derivative.eval a‖ ∧
‖F.eval z‖ ≤ ‖F.derivative.eval a‖^2 * T ^ (2^n)
```

To construct our Newton sequence, we must (1) provide a value satisfying `ih 0`, and (2) assuming we have a value `z : ℤ_[p]` satisfying `ih n z`, produce a value `z' : ℤ_[p]` satisfying `ih (n+1) z'`. The informal recursion indicates that our base value should be `a`, and it is no trouble to prove `ih 0 a`. Under the assumption `ih n z`, we can check that

```
‖F.eval z / F.derivative.eval z‖ ≤ 1
```

and so the recursive value

```
z' := z - F.eval z / F.derivative.eval z
```

is indeed an integer.

The more difficult part of this induction is to show that `ih (n+1) z'` holds. While there is no deep theory needed to do this, we must calculate some chains of inequalities that, while relatively straightforward, are long and nonlinear. These computations invoke the inductive hypothesis on `z`, the nonarchimedean property of the p-adic norm, and both polynomial identities described in step 1. It takes roughly 70 lines of Lean code to perform these computations, compared to roughly 10 in the informal presentation. Many of these computations fall under the scope of the tool Polya described in [2], and in the future, such a tool could be used to significantly condense this portion of our proof.

These computations are sufficient to define the following:

```
def newton_seq (n : ℕ) : {z : ℤ_[p] // ih n z}
```

Projecting the first components, we obtain a sequence of p-adic integers satisfying the induction hypothesis.

5.4 The Newton sequence is Cauchy

The sequence we have defined should lead us to the root of `F` promised by Hensel's lemma. To reach it, we must show that `newton_seq` is Cauchy, so that the completeness of $\mathbb{Z}_p$ guarantees that a limit exists. We first establish the following lemma, which follows from another inequality computation:

```
lemma newton_seq_dist {n k : ℕ} (hnk : n ≤ k) :
  ‖newton_seq k - newton_seq n‖ ≤
    ‖F.derivative.eval a‖ * T^(2^n)
```

(It is here that the special case $f(a) = 0$ diverges from the general argument.) Since `0 ≤ T < 1`, Lean's analysis library makes it easy to show that the right hand side tends to 0, from which we can deduce (from general properties of sequences) that `newton_seq` is Cauchy. We can thus define `soln : ℤ_[p]` to be the limit of `newton_seq`.

5.5 Properties of the limit

From our induction hypothesis, we see that the values of `‖F.eval (newton_seq n)‖` tend to 0 as `n` grows. It follows from the continuity of the norm (proved generally over normed spaces) that `‖F.eval soln‖ = 0`, and correspondingly `F.eval soln = 0`, so we have indeed found a root. The equation

```
‖F.derivative.eval soln‖ = ‖F.derivative.eval a‖
```

similarly follows from the induction hypothesis and the continuity of the norm and polynomial evaluation. A third limit argument shows that

```
‖soln - a‖ = ‖F.eval a‖ / ‖F.derivative.eval a‖
```

which implies the (less precise but sometimes more useful) bound `‖soln - a‖ < ‖F.derivative.eval a‖`, the last property we sought.

The limit arguments in this section, when unfolded into the language of metric spaces, appear as frustrating manipulations of small numbers ϵ and large numbers N. We work to avoid as much frustration as possible by making these arguments topologically. Establishing general results about Cauchy sequences on topological spaces lets us keep the ϵ-N manipulations largely isolated.

5.6 Uniqueness of the solution

Hensel's lemma does more than just locate a root of the polynomial f: it guarantees that the root is the only one within a neighborhood of the seed point a. The uniqueness proof follows from a short computation using the first polynomial identity from step 1.

```
lemma soln_unique (z : ℤ_[p]) (he : F.eval z = 0)
  (hlt : ‖z - a‖ < ‖F.derivative.eval a‖) :
  z = soln
```

When a is already a root of f, uniqueness follows even more directly. We can thus show Hensel's lemma by a case distinction on whether $f(a) = 0$, providing a as a witness in the special case and `soln` in the general case.

6 Related work

Although number theory is underrepresented in proof assistant libraries compared to other fields of mathematics, various projects have formalized results in this area. The following incomplete list indicates the depth and flavor of such projects.

The prime number theorem has been a popular target for formalization, verified first in Isabelle by Avigad, Donnelly, Gray, and Raff [1] and subsequently by Harrison in HOL Light [24], Carneiro in Metmath [9], and others. Isabelle's *Archive of Formal Proofs* contains a number of related entries, including Eberl's proof of Dirichlet's theorem [16]. Elliptic curves and their number theoretic consequences have been addressed in multiple formalizations, including by Bartzia and Strub [4]. The transcendence of e and π, a result that is at least adjacent to number theory, was first formalized in Coq by Bernard, Bertot, Rideau, and Strub [5]. Analyses of the solutions to Pell's equations have been formalized in various systems, as has the proof of Fermat's little theorem; these and other classical results appear on Wiedijk's list of formalizations of 100 fundamental theorems [44].

The only formal construction of $\mathbb{Q}_p$ and $\mathbb{Z}_p$ found in the literature is by Pelayo, Voevodsky, and Warren [38], carried out in the Coq UniMath library [42]. Because of the univalent foundations of UniMath, it is difficult to compare their approach with ours. One immediate difference is that Pelayo et al. begin with an algebraic construction of $\mathbb{Z}_p$ rather than an analytic construction of $\mathbb{Q}_p$. This construction defines $\mathbb{Z}_p$ as a quotient on the ring of formal power series of $\mathbb{Z}$, and goes on to define $\mathbb{Q}_p$ as the field of fractions on $\mathbb{Z}_p$. The algebraic approach is perhaps more appropriate in a univalent setting.

A complete theory of the p-adics ultimately requires both analytic and algebraic structure. No matter which is chosen for the initial construction, the properties of the other must eventually be derived. The UniMath development ends soon after the ring and field structures are defined, and does not prove any theorems about $\mathbb{Z}_p$ or $\mathbb{Q}_p$.

An undocumented construction of $\mathbb{Q}_p$ by Harrison is also found in the HOL Light repository [23], where it is written that the development is "meant as an example of using metric space completion." The development defines $\mathbb{Q}_p$ as the metric completion of $\mathbb{Q}$, which we believe is better avoided for pedagogical reasons (Section 4.3). Since metric space completion does not preserve the field operations on $\mathbb{Q}$, much of the construction is dedicated to redefining these operations on $\mathbb{Q}_p$. This development ends once the field structure on $\mathbb{Q}_p$ is established, and does not prove any results about the type. It is interesting to note how the construction in a simply typed logic differs from those in dependent type theory. HOL Light does not allow one to define the type $\mathbb{Q}_p$ depending on p (nor on a proof that p is prime). Instead, Harrison defines a general type `padic` that contains the image of $\mathbb{Q}_p$ for each p. Such an approach is common in HOL-based systems, but is rarely used in systems like Lean, where the dependencies pose no problems.

Martin-Dorel, Hanrot, Mayero, and Théry describe a Coq formalization of Hensel's lemma for the standard integers, and show some of its applications in verifying bounds on rounding errors [33]. Their approach is more explicitly algorithmic than ours, as their applications involve computing solutions to polynomials modulo powers of p (a process known as Hensel lifting). For this purpose, it is reasonable to operate over the standard integers, since there are complications in defining $\mathbb{Q}_p$ as a computable field. The statement of Hensel's lemma is rather less elegant than over $\mathbb{Z}_p$, however, and its import as an algebraic property is hidden.

7 Concluding thoughts

The p-adic numbers and Hensel's lemma are an important tool of modern number theory, including the study of Diophantine equations. A recent project begun at the Vrije Universiteit Amsterdam aims to formalize results in this area [6]. Constructing $\mathbb{Q}_p$ and $\mathbb{Z}_p$ are essential steps toward this goal. We plan to pursue consequences of Hensel's lemma in future work, beginning with applications of the local-global principle. The p-adic numbers can be abstracted in different ways, and Hensel's lemma proved in more general contexts; we plan to explore these possibilities. More concretely, it is often useful to consider the alternative characterization of $\mathbb{Z}_p$ as the inverse limit of the rings $\mathbb{Z}/p^n\mathbb{Z}$. We plan to show the connection between this algebraic approach and our analytic approach.

This development has also served as a case study for using Lean for such a project. The mix of analysis, algebra, and concrete computation make the p-adic numbers an interesting target; we found that Lean and its libraries were up to the task. Two particularly painful parts were managing casts between various number structures and proving long but straightforward nonlinear inequalities. The former is especially likely to occur in further number theoretic developments, since it is often necessary to move between $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{R}$, $\mathbb{Q}_p$, $\mathbb{Z}_p$, and other number structures. We hope to develop tools to assist with this movement using Lean's metaprogramming capabilities.

Acknowledgments

We thank Jeremy Avigad, Jasmin Blanchette, Sander Dahmen, Johannes Hölzl, and the Lean mathlib community for their support, advice, and comments. We acknowledge support from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (grant agreement No. 713999, Matryoshka).

References

[1] Jeremy Avigad, Kevin Donnelly, David Gray, and Paul Raff. 2007. A formally verified proof of the prime number theorem. *ACM Trans. Comput. Logic* 9, 1 (2007), 2. https://doi.org/10.1145/1297658.1297660

[2] Jeremy Avigad, Robert Y. Lewis, and Cody Roux. 2016. A heuristic prover for real inequalities. *Journal of Automated Reasoning* (2016).

[3] Grzegorz Bancerek, Czesław Byliński, Adam Grabowski, Artur Korniłowicz, Roman Matuszewski, Adam Naumowicz, and Karol Pąk. 2018. The Role of the Mizar Mathematical Library for Interactive Proof Development in Mizar. *Journal of Automated Reasoning* 61, 1 (01 Jun 2018), 9–32. https://doi.org/10.1007/s10817-017-9440-6

[4] Evmorfia-Iro Bartzia and Pierre-Yves Strub. 2014. A Formal Library for Elliptic Curves in the Coq Proof Assistant. In *Interactive Theorem Proving*, Gerwin Klein and Ruben Gamboa (Eds.). Springer International Publishing, Cham, 77–92.

[5] Sophie Bernard, Yves Bertot, Laurence Rideau, and Pierre-Yves Strub. 2016. Formal Proofs of Transcendence for e and Pi As an Application of Multivariate and Symmetric Polynomials. In *Proceedings of the 5th ACM SIGPLAN Conference on Certified Programs and Proofs (CPP 2016)*. ACM, New York, NY, USA, 76–87. https://doi.org/10.1145/2854065.2854072

[6] Jasmin Blanchette. 2018. Lean Forward: Usable Computer-Checked Proofs and Computations for Number Theorists. https://lean-forward.github.io/

[7] Nicolas Bourbaki. 1998. *General topology. Chapters 1–4*. Springer, Berlin. vii+437 pages. Translated from the French, Reprint of the 1989 English translation.

[8] T. D. Browning. 2018. How often does the Hasse principle hold? In *Algebraic geometry: Salt Lake City 2015*. Proc. Sympos. Pure Math., Vol. 97. Amer. Math. Soc., Providence, RI, 89–102.

[9] Mario Carneiro. 2016. Formalization of the prime number theorem and Dirichlet's theorem. In *Proceedings of the 9th Conference on Intelligent Computer Mathematics (CICM 2016)*. 10–13.

[10] Mario Carneiro. 2018. The Lean 3 Mathematical Library (presentation). http://robertylewis.com/files/icms/Carneiro_mathlib.pdf

[11] Keith Conrad. [n. d.]. Hensel's Lemma. http://www.math.uconn.edu/~kconrad/blurbs/gradnumthy/hensel.pdf

[12] Thierry Coquand and Christine Paulin. 1990. Inductively defined types. In *COLOG-88 (Tallinn, 1988)*. Lec. Notes in Comp. Sci., Vol. 417. Springer, Berlin, 50–66. https://doi.org/10.1007/3-540-52335-9_47

[13] Sander R. Dahmen and Samir Siksek. 2014. Perfect powers expressible as sums of two fifth or seventh powers. *Acta Arith.* 164, 1 (2014),

65–100. https://doi.org/10.4064/aa164-1-5

[14] Leonardo de Moura, Gabriel Ebner, Jared Roesch, and Sebastian Ullrich. 2017. The Lean Theorem Prover (presentation). http://leanprover.github.io/presentations/20170116_POPL

[15] Leonardo de Moura, Soonho Kong, Jeremy Avigad, Floris van Doorn, and Jakob von Raumer. 2014. The Lean Theorem Prover. *http://leanprover.github.io/files/system.pdf* (2014).

[16] Manuel Eberl. 2017. Dirichlet L-Functions and Dirichlet's Theorem. *Archive of Formal Proofs* (Dec. 2017). http://isa-afp.org/entries/Dirichlet_L.html, Formal proof development.

[17] Gabriel Ebner, Sebastian Ullrich, Jared Roesch, Jeremy Avigad, and Leonardo de Moura. 2017. A metaprogramming framework for formal verification. *Proceedings of the ACM on Programming Languages* 1, ICFP (2017), 34.

[18] Georges Gonthier. 2008. Formal proof–the four-color theorem. *Notices of the AMS* 55, 11 (2008), 1382–1393.

[19] Georges Gonthier, Andrea Asperti, Jeremy Avigad, Yves Bertot, Cyril Cohen, François Garillot, Stéphane Le Roux, Assia Mahboubi, Russell O'Connor, Sidi Ould Biha, Ioana Pasca, Laurence Rideau, Alexey Solovyev, Enrico Tassi, and Laurent Théry. 2013. A Machine-Checked Proof of the Odd Order Theorem. In *Interactive Theorem Proving*, Sandrine Blazy, Christine Paulin-Mohring, and David Pichardie (Eds.). Springer, Berlin, 163–179.

[20] Fernando Q. Gouvêa. 1997. *p-adic Numbers* (second ed.). Springer, Berlin. vi+298 pages. https://doi.org/10.1007/978-3-642-59058-0

[21] Benjamin Gregoire and Assia Mahboubi. 2005. Proving Equalities in a Commutative Ring Done Right in Coq. In *Theorem Proving in Higher Order Logics (TPHOLs 2005), LNCS 3603*. Springer, 98–113.

[22] Thomas Hales, Mark Adams, Gertrud Bauer, Tat Dat Dang, John Harrison, Hoang Le Truong, Cezary Kaliszyk, Victor Magron, Sean McLaughlin, Tat Thang Nguyen, et al. 2017. A formal proof of the Kepler conjecture. In *Forum of Mathematics, Pi*, Vol. 5. Cambridge University Press.

[23] John Harrison. [n. d.]. The HOL Light theorem prover. https://www.cl.cam.ac.uk/~jrh13/hol-light/

[24] John Harrison. 2009. Formalizing an analytic proof of the Prime Number Theorem. *Journal of Automated Reasoning* 43 (2009), 243–261.

[25] Eric C. R. Hehner and RNS Horspool. 1979. A new representation of the rational numbers for fast easy arithmetic. *SIAM J. Comput.* 8, 2 (1979), 124–134.

[26] Kurt Hensel. 1908. *Theorie der algebraischen Zahlen.* Number v. 1 in Cornell University Library historical math monographs. B. G. Teubner. https://books.google.nl/books?id=CHUAAAAAMAAJ

[27] Johannes Hölzl, Fabian Immler, and Brian Huffman. 2013. Type classes and filters for mathematical analysis in Isabelle/HOL. In *International Conference on Interactive Theorem Proving*. Springer, 279–294.

[28] Ioan James. 1999. *Topologies and uniformities.* Springer-Verlag London, Ltd., London. xvi+230 pages. https://doi.org/10.1007/978-1-4471-3994-2 Revised version of ıt Topological and uniform spaces [Springer, New York, 1987; MR0884154 (89b:54001)].

[29] Christer Lech. 1953. A note on recurring series. *Ark. Mat.* 2, 5 (08 1953), 417–421. https://doi.org/10.1007/BF02590997

[30] Robert Y. Lewis. 2017. An extensible ad hoc interface between Lean and Mathematica.. In *Proof eXchange for Theorem Proving*.

[31] Assia Mahboubi and Enrico Tassi. [n. d.]. *Mathematical Components.* https://math-comp.github.io/mcb/

[32] K. Mahler. 1958. An interpolation series for continuous functions of a p-adic variable. *J. Reine Angew. Math.* 199 (1958), 23–34. https://doi.org/10.1515/crll.1958.199.23

[33] Érik Martin-Dorel, Guillaume Hanrot, Micaela Mayero, and Laurent Théry. 2015. Formally Verified Certificate Checkers for Hardest-to-Round Computation. *Journal of Automated Reasoning* 54, 1 (01 Jan 2015), 1–29. https://doi.org/10.1007/s10817-014-9312-2

[34] William McCallum and Bjorn Poonen. 2012. The method of Chabauty and Coleman. In *Explicit methods in number theory*. Panor. Synthèses, Vol. 36. Soc. Math. France, Paris, 99–117.

[35] Norman Megill. 2006. Metamath. *The Seventeen Provers of the World* (2006), 88–95.

[36] Tobias Nipkow. [n. d.]. Archive of Formal Proofs. https://www.isa-afp.org/

[37] Alexander Ostrowski. 1916. Über einige Lösungen der Funktionalgleichung $\psi(x) \cdot \psi(y) = \psi(xy)$. *Acta Math.* 41 (1916), 271–284. https://doi.org/10.1007/BF02422947

[38] Álvaro Pelayo, Vladimir Voevodsky, and Michael A. Warren. 2015. A univalent formalization of the p-adic numbers. *Math. Structures Comput. Sci.* 25, 5 (2015), 1147–1171. https://doi.org/10.1017/S0960129514000541

[39] J.-P. Serre. 1973. *A course in arithmetic.* Springer-Verlag, New York-Heidelberg. viii+115 pages. Translated from the French, Graduate Texts in Mathematics, No. 7.

[40] Bas Spitters and Eelis van der Weegen. 2011. Type classes for mathematics in type theory. *Mathematical Structures in Computer Science* 21, 4 (2011), 795–825.

[41] Floris van Doorn, Jakob von Raumer, and Ulrik Buchholtz. 2017. *Homotopy Type Theory in Lean.* Springer International Publishing, Cham, 479–495. https://doi.org/10.1007/978-3-319-66107-0_30

[42] Vladimir Voevodsky, Benedikt Ahrens, Daniel Grayson, et al. [n. d.]. UniMath — a computer-checked library of univalent mathematics. available at https://github.com/UniMath/UniMath. https://github.com/UniMath/UniMath

[43] P. Wadler and S. Blott. 1989. How to Make Ad-hoc Polymorphism Less Ad Hoc. In *Proceedings of the 16th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL '89)*. ACM, New York, NY, USA, 60–76. https://doi.org/10.1145/75277.75283

[44] Freek Wiedijk. [n. d.]. Formalizing 100 Theorems. http://www.cs.ru.nl/~freek/100/