Formalisation of p-adic L-functions in Lean 3

Ashvni Narayanan
London School of Geometry and Number Theory, Imperial College London

Abstract

This paper decribes formalisation of p-adic L-functions in an interactive theorem prover Lean 3. Kubota-Leopoldt p-adic L-functions are fundamental number theoretic objects. These special functions emerge from the special values they take at negative integers in terms of generalized Bernoulli polynomials. We formalise their definition in terms of a p-adic integral with respect to the Bernoulli measure, and prove that they take the required values at negative integers.

2012 ACM Subject Classification Mathematics of computing → Mathematical software; Theory of computation → Formal languages and automata theory

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

Digital Object Identifier 10.4230/LIPIcs.CVIT.2016.23

Supplementary Material Copies of the source files relevant to this paper are available in a separate repository.
Software: https://github.com/laughinggas/p-adic-L-functions

Funding *Ashvni Narayanan*: EPSRC Grant EP/S021590/1 (UK)

Acknowledgements The author is supported by EPSRC. The author would like to thank her PhD supervisor Prof Kevin Buzzard for several helpful insights. She would also like to thank Dr Filippo A. E. Nuccio for helpful conversations that helped give shape to the project. She is very grateful to the entire Lean community for their timely help and consistent support.

1 Introduction

We are working on formalising mathematics in an interactive theorem prover called Lean. Formal verification involves the use of logical and computational methods to establish claims that are expressed in precise mathematical terms [1]. Lean is a powerful tool that facilitates formalisation of a system of mathematics supported by a basic set of axioms. There is a large mathematical library of theorems verified by Lean called `mathlib`, maintained by a community of computer scientists and mathematicians. One can then formally verify proofs of new theorems dependent on preexisting theorems in `mathlib`. `mathlib` currently contains 100579 theorems(as of early October 2022). It would be impossible to construct such a vast library without a highly collaborative spirit and a communal decentralized effort, one of Lean's best features.

p-adic L-functions are a well studied number theoretic object. They were initially constructed by Kubota and Leopoldt in [3]. Their motivation was to construct a meromorphic function that helps study the Kummer congruence for Bernoulli numbers, and gives information regarding p-adic class numbers. As a result, these functions take twisted values of the Dirichlet L-function at negative integers, and are also related to the generalized Bernoulli numbers and the p-adic zeta function. There are several different ways of constructing p-adic L-functions, we refer to the constructions given in Chapter 12 of [5]. This has never been done before in any theorem prover. As a result, one needs to build a lot of background (in the maximum possible generality) before embarking on the main goal.

We first give a mathematical overview in this section. The following tools are needed to construct p-adic L-functions : Dirichlet characters, Bernoulli numbers and polynomials, locally compact Hausdorff totally disconnected spaces, and the p-adic integers and its topological properties. We introduce these in Section 2, define the p-adic L-function in Section 3, and evaluate it at negative integers in Section 4, finishing with a summary in Section 5.

1.1 Mathematical overview

We give a brief overview of the mathematics formalised in this project. L-functions are a fundamental object, appearing almost everywhere in modern number theory. The Dirichlet L-function associated to a Dirichlet character χ is given by

$$L(s,\chi) = \sum_{n=1}^{\infty} \frac{\chi(n)}{n^s} = \prod_{p \text{ prime}} \frac{1}{1-\chi(p)p^{-s}}$$

where s is a complex variable with $Re(s) > 1$. This can be analytically extended to the entire complex plane, with a simple pole at $s = 1$ when $\chi = 1$. Note also that $L(s,1)$ is the same as the Reimann zeta function. Moreover, it is known that $L(1-n,\chi) = -\frac{B_{n,\chi}}{n}$, where $B_{n,\chi}$ are the generalized Bernoulli numbers.

In this paper, we construct, for an integer prime p, a p-adic analogue of $L(s,\chi)$, called the Kubota-Leopoldt p-adic L-function, denoted $L_p(s,\chi)$. This is generally done by continuously extending the function $L_p(1-n,\chi) := (1-\chi(p)p^{n-1})L(1-n,\chi)$ to the complete p-adic space $\mathbb{C}_p$. In fact, $L_p(s,1)$ is analytic except for a pole at $s = 1$ with residue $1-\frac{1}{p}$ (Theorem 5.11, [5]).

Formalisation of the p-adic L-functions via analytic continuation was hard, since $\mathbb{C}_p$ did not exist in `mathlib` at the time. Following [5], we define it in terms of a "p-adic integral" with respect to the Bernoulli measure. We explain these terms below.

A profinite space is a compact, Hausdorff and totally disconnected space. The p-adic integers $\mathbb{Z}_p$, which are the completion of the integers $\mathbb{Z}$ with respect to the valuation $\nu_p(p^\alpha \prod_{p_i \neq p} p_i^{\alpha_i}) = \alpha$ are a profinite space. One may also think of them as the inverse limit of the discrete topological spaces $\mathbb{Z}/p^n\mathbb{Z}$, that is, $\mathbb{Z}_p = \operatorname{proj\,lim}_n \mathbb{Z}/p^n\mathbb{Z}$. A corollary of this is that $\mathbb{Z}_p$ has a topological basis of clopen sets (both open and closed), $(\{x \in \mathbb{Z}_p | x \equiv a (\text{mod } p^n)\})_{n,a}$, for $n \in \mathbb{N}$ and $a \in \mathbb{Z}/p^n\mathbb{Z}$.

Locally constant functions are those for which the preimage of any set is open. Given a profinite space X and a normed ring R, one can show that the locally constant functions from X to R (denoted $LC(X,R)$) are dense in the space of continuous functions from X to R (denoted $C(X,R)$).

Given an abelian group A, a distribution is defined to be an A-linear map from $LC(X,A)$ to A. A p-adic measure ϕ is defined to be a bounded distribution, that is, $\forall f \in LC(X,R)$, $\exists K > 0$ such that $||\phi(f)|| \leq K||f||$, where $||f|| = \sup_{x \in X} ||f(x)||$. An example of a p-adic measure is a Bernoulli measure. Given a natural number d coprime to p and a clopen set $U_{n,a}$ of $\mathbb{Z}/d\mathbb{Z} \times \mathbb{Z}_p$, the characteristic function $\chi_{n,a}$ (defined to be 1 on $U_{n,a}$ and 0 otherwise) is a locally constant function. Given a natural number c that is coprime to d and p, we then define the Bernoulli measure E_c by :

$$E_c(U_{n,a}) := \left\{\frac{a}{dp^{n+1}}\right\} - c\left\{\frac{c^{-1}a}{dp^{n+1}}\right\} + \frac{c-1}{2}$$

Given a p-adic measure μ, we define the p-adic integral with respect to μ to be $\int f d\mu := \mu(f)$ for any locally constant function f, and extending this definition to $C(X,R)$. In fact, this is an R-linear map.

Finally, the p-adic L-function is defined to be a p-adic integral with respect to the Bernoulli measure. The characterizing property of the p-adic L-function is its evaluation at negative integers :

$$L_p(1-n,\chi) = -(1-\chi\omega^{-n}(p)p^{n-1})\frac{B_{n,\chi\omega^{-n}}}{n}$$

for $n \geq 1$. If defined as an analytic continuation, this would follow directly. However, when defined as a p-adic integral, additional work is needed to prove this. The equivalence of these two definitions is proved in Theorem 12.2 of [5]. The same theorem also proves the independence of the definition from the additional variable c.

Our contributions to this theory include a formalised definition of the p-adic L-function in generality, taking values in a normed complete non-Archimedean $\mathbb{Q}_p$-algebra, instead of just $\mathbb{C}_p$. Further, it takes as input continuous monoid homomorphisms, also known as elements of the weight space. We have also developed an extensive theory for Dirichlet characters, Bernoulli numbers and polynomials, generalized Bernoulli numbers, properties of p-adic integers and modular arithmetic.

1.2 Lean and `mathlib`

Lean 3 is a functional programming language and interactive theorem prover based on dependent type theory. This project is based on Lean's mathematical library `mathlib 3`, which is characterized by its decentralized nature with over 300 contributors. Thus, it is impossible to cite every author who contributed a piece of code that we used.

We assume the reader is familiar with structures such as `def`, `abbreviation`, `lemma`, `theorem`, which are used constantly. An important property of Lean is its typeclass inference system - Lean "remembers" properties given to a `structure` or `class` embedded in an `instance` structure. This is explained in detail in [4]. We shall also use several tactics in proofs, such as `rw`, `apply`, `conv` and `refine` [1].

2 Preliminaries

2.1 Filters and convergence

None of the mathematical proofs require filters on paper, however, we find that working with them makes formalising these proofs significantly less cumbersome. We shall not delve into the details of what a filter is, but instead explain how we use them to formalise convergence and limits. Often, we have expressions of the form $\lim_{n\to\infty} f_n(x) = a$ for a sequence of functions $(f_n)_n$. This is represented in Lean as :

```
filter.tendsto (λ n : ℕ, f_n) filter.at_top (nhds a)
```

Here, `filter.at_top` (for the naturals) is a filter on $\mathbb{N}$ generated by the collection of sets $\{b | a \leq b\}$ for all $a \in \mathbb{N}$. There is a large library of lemmas regarding `tendsto` in `mathlib`. Some important properties that we use frequently include :

```
/-- If lim_n f(n) = a and lim_n g(n) = b, then lim_n f(n) + g(n) = a + b -/
lemma tendsto.add {α : Type} {M : Type u_1} [topological_space M] [has_add M]
  [has_continuous_add M] {f g : α → M} {x : filter α} {a b : M}
  (hf : tendsto f x (nhds a)) (hg : tendsto g x (nhds b)) :
  tendsto (λ (x : α), f x + g x) x (nhds (a + b))
/-- If f = g, then lim f = lim g. -/
lemma filter.tendsto_congr {α : Type} {β : Type u_1} {f₁ f₂ : α → β} {l₁ : filter α}
  {l₂ : filter β} (h : ∀ (x : α), f₁ x = f₂ x) : tendsto f₁ l₁ l₂ ↔ tendsto f₂ l₁ l₂
/-- If lim (c · f) = 0 for some nonzero constant c, then lim f = 0. -/
lemma tendsto_zero_of_tendsto_const_smul_zero [algebra ℚ_[p] R] {f : ℕ → R} {l : filter ℕ}
  {c : ℚ_[p]} (hc : c ≠ 0) (hf : tendsto (λ y, c · f y) l (nhds 0)) : tendsto f l (nhds 0)
/-- If f₁ and f₂ are equal almost everywhere, then f₁ converges if and only if f₂ converges. -/
lemma filter.tendsto_congr' {α : Type} {β : Type u_1} {f₁ f₂ : α → β} {l₁ : filter α}
  {l₂ : filter β} (h : f₁ =ᶠ[l₁] f₂) : tendsto f₁ l₁ l₂ ↔ tendsto f₂ l₁ l₂
```

We aim to use these lemmas as much as possible in order to avoid messy calculations with inequalities on norms. The last lemma is particularly useful. it shows that sequences that are the same after finitely many elements have the same limit. Given `f₁ f₂ :ℕ → R`, `f₁ =ᶠ[at_top] f₂ ⟷ ∃ (a : ℕ), ∀ (b : ℕ), b ≥ a, f₁ b = f₂ b`.

2.2 Modular arithmetic and $\mathbb{Z}_p$

Some fundamental objects with which we shall work throughout are the finite spaces $\mathbb{Z}/n\mathbb{Z}$. Note that proving properties for `zmod n` is equivalent to proving them for any finite abelian group. Given $n \in \mathbb{N}$, $\mathbb{Z}/n\mathbb{Z}$ is the same as `fin n`, the set of natural numbers upto n. This has a natural group structure, and is given the discrete topology, making it a topological group. Some of the maps used constantly include `val:zmod n → ℕ`, which takes any element to its smallest nonnegative

[1] https://leanprover-community.github.io/mathlib_docs/tactics.html has a full list of tactics in Lean

reprentative less than `n`; and `cast_hom:zmod n → R`, a coercion to a ring, obtained by composing the canonical coercion with `val`. If `R` has characteristic dividing `n`, the map is a ring homomorphism. Given coprime natural numbers m and n, an important equivalence is : `chinese_remainder:zmod (m * n) ≃+* zmod m × zmod n`. About 30 additional lemmas were required, which have been put in a separate file, `zmod_properties.lean`. We assume p is a prime and d is a positive natural with $gcd(d, p) = 1$. The ring of p-adic integers, denoted `ℤ_[p]`, is the completion of $\mathbb{Z}$ by the p-adic valuation. It has the following properties :

- As a profinite limit, $\mathbb{Z}_p = \varprojlim \mathbb{Z}/p^n\mathbb{Z}$. As a result, one can find (compatible) surjective ring homomorphisms `to_zmod_pow : ℤ_[p] →+* zmod (p^n)` for all `n`.
- $\mathbb{Z}_p$ has the profinite topology, induced by the discrete topology on $\mathbb{Z}/p^n\mathbb{Z}$. Hence, $\mathbb{Z}_p$ is a compact, Hausdorff totally disconnected space with a clopen basis of the form $(\{a + p^n\mathbb{Z}_p\})_{n,a}$ for $a \in \mathbb{Z}/p^n\mathbb{Z}$.
- For $p > 2$, $\mathbb{Z}_p^\times \simeq (\mathbb{Z}/p\mathbb{Z})^\times \times (1 + p\mathbb{Z}_p)$[2]. Also, $(\mathbb{Z}/d\mathbb{Z})^\times \times \mathbb{Z}_p^\times \simeq (\mathbb{Z}/dp\mathbb{Z})^\times \times (1 + p\mathbb{Z}_p)$

These properties characterize the p-adic integers, and are integral to this work. While some preexisted in `mathlib`, about 40 additional lemmas have been proved in `padic_int.properties.lean`. Moreover, lots of facts regarding the clopen basis given above have been utilized and proved in `padic_int.clopen_properties.lean`.

2.3 Dirichlet characters and the Teichmüller character

An important task was to formalise Dirichlet characters, an integral part of the definition of the p-adic L-function. Dirichlet characters are often not found to be defined in this technical manner. Another addition is the definition of Dirichlet characters of level and conductor 0. The words character and Dirichlet character are used interchangeably.

The Dirichlet characters are usually defined as group homomorphisms from $\mathbb{Z}/n\mathbb{Z}^\times$ to $\mathbb{C}^\times$ for some natural number n. We can then assign levels to Dirichlet characters and relate Dirichlet characters of different levels, constructing a "compatible" system of characters for certain values of n. We generalize the definition from group homomorphisms to $\mathbb{C}$ to monoid homomorphisms on any `monoid` :

```
abbreviation dirichlet_character (R : Type*) [monoid R] (n : ℕ) := units (zmod n) →* units R
abbreviation lev {R : Type*} [monoid R] {n : ℕ} (χ : dirichlet_character R n) : ℕ := n
```

Note that the linter returns an extra unused argument warning for the latter definition.

Given a Dirichlet character χ, `asso_dirichlet_character` χ returns a monoid homomorphism from $\mathbb{Z}/n\mathbb{Z}$ to R, which is χ on the units and 0 otherwise. Most of our work is on $\mathbb{Z}/n\mathbb{Z}$ instead of its units, hence this is a vital definition.

```
noncomputable abbreviation asso_dirichlet_character {R : Type*} [monoid_with_zero R] {n : ℕ}
(χ : dirichlet_character R n) : zmod n →* R :=
{ to_fun := function.extend (units.coe_hom (zmod n)) ((units.coe_hom R) ∘ χ) 0, .. }
```

One would like to shift between compatible Dirichlet characters of different levels. For this, we have the following tools :

```
/-- Extends the Dirichlet character χ of level n to level m, where n | m. -/
def change_level {m : ℕ} (hm : n | m) : dirichlet_character R n →* dirichlet_character R m :=
{ to_fun := λ ψ, ψ.comp (units.map (zmod.cast_hom hm (zmod n))), .. }
/-- χ₀ of level d factors through χ of level n if d | n and χ₀ = χ ∘ (zmod n → zmod d). -/
structure factors_through (d : ℕ) : Prop :=
(dvd : d | n) (ind_char : ∃ χ₀ : dirichlet_character R d, χ = χ₀.change_level dvd)
```

With the assistance of a few lemmas, it is easy to translate between `change_level` and `factors_through`. The notions of primitivity and conductor of a Dirichlet character follow easily :

[2] $\mathbb{Z}_2^\times \cong \{\pm 1\} \times (1 + 4\mathbb{Z}_2)$

```
/-- The set of natural numbers for which a Dirichlet character is periodic. -/
def conductor_set : set ℕ := {x : ℕ | χ.factors_through x}
/-- The minimum natural number n for which a Dirichlet character is periodic. -/
noncomputable def conductor : ℕ := Inf (conductor_set χ)
/-- A character is primitive if its level is equal to its conductor. -/
def is_primitive : Prop := χ.conductor = n
/-- The primitive character associated to a Dirichlet character. -/
noncomputable def asso_primitive_character : dirichlet_character R χ.conductor :=
classical.some (χ.factors_through_conductor).ind_char
```

Here, `classical.some` makes an arbitrary choice of an element from a nonempty space, and `classical.some_spec` lists down the properties of this random element coming from the space. Our definition of Dirichlet characters holds also for $n = 0$, ie, on $\mathbb{Z}$. This is easily separated from the rest by the lemma :

```
lemma conductor_eq_zero_iff_level_eq_zero : χ.conductor = 0 ↔ n = 0
```

An issue with the definition `is_primitive` is that it equates levels of two Dirichlet characters, however, this does not imply the types are equal. As an example, consider :

```
lemma dirichlet_character_eq_of_eq {a b : ℕ} (h : a = b) :
dirichlet_character R a = dirichlet_character R b
```

It is best avoided to make lemmas about equality of types, however, the tactic `subst` fails here. We needed it to deal with further complex statements, such as, χ and χ`.asso_primitive_dirichlet_character` being the "same" when χ is primitive. The tactic `congr'` and the lemma `cast_heq` were useful in dealing with such issues. To this effect, making `change_level` a `monoid_hom` helped with constructing the following useful lemma, reiterating the importance of choosing an "appropriate" definition :

```
lemma change_level_heq {a b : ℕ} {S : Type*} [comm_monoid_with_zero S]
  (χ : dirichlet_character S a) (h : a = b) : change_level (show a | b, from by {rw h}) χ == χ
```

Traditionally only for primitive characters, our definition of multiplication of characters extends to any two characters:

```
noncomputable def mul {m n : ℕ} (χ₁ : dirichlet_character R n) (χ₂ : dirichlet_character R m) :=
asso_primitive_character(change_level χ₁ (dvd_lcm_left n m) * change_level χ₂ (dvd_lcm_right n m))
```

This takes as input characters χ_1 and χ_2 of levels n and m respectively, and returns the primitive character associated to $\chi_1'\chi_2'$, where χ_1' and χ_2' are obtained by changing the levels of χ_1 and χ_2 to nm.
We need the notion of odd and even characters. A character χ is odd if $\chi(-1) = -1$, and even if $\chi(-1) = 1$. If the target is a commutative ring, then any character is either odd or even :

```
lemma is_odd_or_is_even {S : Type*} [comm_ring S] [no_zero_divisors S] {m : ℕ}
  (ψ : dirichlet_character S m) : ψ.is_odd ∨ ψ.is_even
```

Other important properties include :

```
/-- Dirichlet characters are continuous. -/
lemma continuous {R : Type*} [monoid R] [topological_space R] {n : ℕ}
  (χ : dirichlet_character R n) : continuous χ
/-- Associated Dirichlet characters are continuous. -/
lemma asso_dirichlet_character_continuous {R : Type*} [monoid_with_zero R] [topological_space R]
  {n : ℕ} (χ : dirichlet_character R n) : continuous (asso_dirichlet_character χ)
/-- Associated Dirichlet characters are bounded. -/
```

```
lemma asso_dirichlet_character_bounded {R : Type*} [monoid_with_zero R] [normed_group R]
  {n : ℕ} [fact (0 < n)] (χ : dirichlet_character R n) : ∃ M : ℝ,
  ‖(⟨asso_dirichlet_character χ, χ.asso_dirichlet_character_continuous⟩ : C(zmod n, R))‖ < M
```

In the last lemma, the norm is the sup norm on $C(\mathbb{Z}/n\mathbb{Z}, R)$. These are all the ingredients we needed. Let us now define a special Dirichlet character, the Teichmüller character.

2.3.1 Teichmüller character

The initial effort was to formalise the definition of the Teichmüller character directly. However, it was discovered that Witt vectors, and in particular Teichmüller lifts had previously been added to `mathlib`. This was used and was very helpful. It reiterates the importance of the collaborative spirit of Lean, and of making definitions in the correct generality.

It is beyond the scope of this text to define Witt vectors and do it justice. For a commutative ring R and a prime number p, one can obtain a ring of Witt vectors $\mathbb{W}(R)$. When we take $R = \mathbb{Z}/p\mathbb{Z}$, we get

```
def equiv : 𝕎 (zmod p) ≃+* ℤ_[p]
```

One also obtains the Teichmüller lift $R \to \mathbb{W}(R)$. Given $r \in R$, the 0-th coefficient is r, and the other coefficients are 0. This map is a multiplicative monoid homomorphism and is denoted `teichmuller`.

Note that given a multiplicative homomorphism `R →* S` for monoids `R` and `S`, one can obtain a multiplicative homomorphism `units R →* units S` (by `units.map`). Combining this with the previous two definitions, we obtain our definition of the Teichmüller character :

```
noncomputable abbreviation teichmuller_character_mod_p (p : ℕ) [fact (nat.prime p)] :
    dirichlet_character ℤ_[p] p :=
  units.map (((witt_vector.equiv p).to_monoid_hom).comp (witt_vector.teichmuller p))
```

Often we view this as taking values in a $\mathbb{Q}_p$-algebra R, by composing it with the algebra map `algebra_map ℚ_[p] R`, which identifies elements of $\mathbb{Q}_p$ in R.
For odd primes p, the Teichmüller character is primitive, and `1` otherwise :

```
lemma is_primitive_teichmuller_character_zmod_p (p : ℕ) [fact (nat.prime p)] (hp : 2 < p) :
  (teichmuller_character_mod_p p).is_primitive
lemma teichmuller_character_mod_p_two : teichmuller_character_mod_p 2 = 1
```

2.4 Bernoulli polynomials and the generalized Bernoulli number

The Bernoulli numbers B_n are generating functions given by $\sum B_n \frac{t^n}{n!} = \frac{t}{e^t - 1}$. Note that several authors think of Bernoulli numbers B'_n to be defined as $\sum B'_n \frac{t^n}{n!} = \frac{t}{1 - e^{-t}}$. The difference between these two is : $B'_n = (-1)^n B_n$, with $B_1 = -\frac{1}{2}$. A reformulation gives :

$$B_n = 1 - \sum_{k=0}^{n-1} \binom{n}{k} \frac{B_k}{n-k+1}$$

In `mathlib`, B_n was already defined (by Johan Commelin) as above. However, we needed B'_n, which was then defined as :

```
  def bernoulli (n : ℕ) : ℚ := (-1)^n * bernoulli' n
```

The Bernoulli polynomials, denoted $B_n(X)$, a generalization of the Bernoulli numbers, are generating functions $\sum_{n=0}^{\infty} B_n(X) \frac{t^n}{n!} = \frac{te^{tX}}{e^t - 1}$. This gives :

$$B_n(X) = \sum_{i=0}^{n} \binom{n}{i} B_i X^{n-i}$$

We now define the Bernoulli polynomials as :

```
def polynomial.bernoulli (n : ℕ) : polynomial ℚ :=
  Σ i in range (n + 1), polynomial.monomial (n - i) ((bernoulli i) * (choose n i))
```

Here, `polynomial.monomial n a` translates to aX^n. A small aspect of this naming convention is that if the namespaces for Bernoulli numbers and polynomials are both open (which is often the case), in order to use the Bernoulli numbers, one needs to use `_root_.bernoulli`. We shall use them interchangeably here, when the context is clear.

The following properties of Bernoulli polynomials were proved :

```
/-- B_0(X) = 1 -/
lemma polynomial.bernoulli_zero : bernoulli 0 = 1
/-- B_n(0) = B_n -/
lemma polynomial.bernoulli_eval_zero (n : ℕ) : (bernoulli n).eval 0 = bernoulli n
/-- B_n(1) = B_n' -/
lemma polynomial.bernoulli_eval_one (n : ℕ) : (bernoulli n).eval 1 = bernoulli' n
```

For a commutative $\mathbb{Q}$-algebra A and $t \in A$,

$$\left(\sum_{n=0}^{\infty} B_n(t)\frac{X^n}{n!}\right)(e^X - 1) = Xe^{tX} :$$

```
theorem polynomial.bernoulli_generating_function (t : A) :
  mk (λ n, aeval t ((1 / n! : ℚ) · bernoulli n)) * (exp A - 1) = X * rescale t (exp A)
```

The theorem `power_series.mk` (abbreviated as `mk`) defines a formal power series in terms of its coefficients, that is, $\sum_{n=0}^{\infty} a_n X^n$ translates to `mk (λ n, a_n)`. The symbol $\cdot$ represents scalar multiplication of $\mathbb{Q}$ on $\mathbb{Q}$-polynomials. The exponential function e^x, which is defined as a Taylor series expansion, takes as input A, but not x. In order to define e^{tx}, we need to use (a ring homomorphism) `rescale y` : for an element y of a commutative semiring B, it takes a formal power series $f(X)$ over B to $f(yX)$.

The proof of the last theorem involves equating the n^{th} coefficients of the RHS and the LHS. After differentiating between n zero and nonzero, one requires the following lemma to complete the nonzero case :

```
theorem polynomial.sum_bernoulli (n : ℕ) :
  Σ k in range (n + 1), ((n + 1).choose k : ℚ) · bernoulli k = polynomial.monomial n (n + 1 : ℚ)
```

The proof of this theorem follows from the following property of Bernoulli numbers :

```
  theorem sum_bernoulli (n : ℕ):
  Σ k in range n, (n.choose k : ℚ) * bernoulli k = if n = 1 then 1 else 0
```

In order to prove properties of generalized Bernoulli numbers, we needed the following theorem :

```
/-- Bernoulli polynomials multiplication theorem :
  For k ≥ 1, B_m(k*x) = Σ i in range k, B_m (x + i / k).  -/
theorem bernoulli_eval_mul' (m : ℕ) {k : ℕ} (hk : k ≠ 0) (x : ℚ) :
  (bernoulli m).eval ((k : ℚ) * x) = k^(m - 1 : ℤ) * Σ i in range k, (bernoulli m).eval (x + i/k)
```

The proof uses the generating function equality, which makes the proof calculation heavy. It suffices to show

$$\sum_n \sum_{i=0}^{k-1} \frac{k^{n-1}}{n!} B_n\left(x + \frac{i}{k}\right) = (e^{kx} - 1)\sum_n \frac{B_n(kx)}{n!}$$

2.4.1 Generalized Bernoulli numbers

Given a primitive Dirichlet character χ of conductor f, the generalized Bernoulli numbers are defined as (section 4.1, [5]) $\sum_{n=0}^{\infty} B_{n,\chi} \frac{t^n}{n!} = \sum_{a=1}^{f} \frac{\chi(a)te^{at}}{e^{ft}-1}$. For any multiple F of f, Proposition 4.1 of [5] gives us :

$$B_{n,\chi} = F^{n-1} \sum_{a=1}^{F} \chi(a) B_n\left(\frac{a}{F}\right)$$

This is much easier to work with, so we use this as our definition, taking $F = f$:

```
def general_bernoulli_number {S : Type*} [comm_semiring S] [algebra ℚ S] {n : ℕ}
  (ψ : dirichlet_character S n) (m : ℕ) : S :=
  (algebra_map ℚ S ((ψ.conductor)^(m - 1 : ℤ)))*(Σ a in finset.range ψ.conductor,
  asso_dirichlet_character (asso_primitive_character ψ) a.succ *
  algebra_map ℚ S ((polynomial.bernoulli m).eval (a.succ / ψ.conductor : ℚ)))
```

Contrary to the traditional definition, this is for all characters. The Dirichlet character ψ takes values in any commutative $\mathbb{Q}$-algebra, instead of $\mathbb{C}$. One had to also explicitly mention that `m - 1` must be taken to have type $\mathbb{Z}$, since Lean would otherwise infer it to have type $\mathbb{N}$, which might have caused errors (subtraction on $\mathbb{N}$ and $\mathbb{Z}$ are different).

The following are some important properties of generalized Bernoulli numbers :

```
/-- B_{n,1} = B_n, where 1 is the trivial Dirichlet character of level 1. -/
lemma one_eval {n : ℕ} :
  general_bernoulli_number (1 : dirichlet_character S 1) n = algebra_map ℚ S (bernoulli' n)
/-- Showing that the definition of general_bernoulli_number is independent of F,
  where F is a multiple of the conductor. -/
lemma eq_sum_bernoulli_of_conductor_dvd {F : ℕ} [hF : fact (0 < F)] (m : ℕ) (h : ψ.conductor|F) :
  general_bernoulli_number ψ m = (algebra_map ℚ S) (F^(m - 1 : ℤ)) *
  (Σ a in finset.range F, asso_dirichlet_character ψ.asso_primitive_character a.succ *
  algebra_map ℚ S ((polynomial.bernoulli m).eval (a.succ / F : ℚ)))
```

The latter lemma proves Proposition 4.1 of [5]. It is false for $F = 0$, since there is no dependency of the LHS on F.

2.4.2 A special property of generalized Bernoulli numbers

An important property of these numbers is (from Proposition 7.2 of [5]), for $n > 1$, :

▶ **Theorem 1.**

$$\lim_{n\to\infty} \frac{1}{dp^n} \sum_{0<a<dp^n;(a,dp)=1} \chi\omega^{-m}(a)a^m = (1 - \chi\omega^{-m}(p)p^{m-1})B_{m,\chi\omega^{-m}}$$

This is formulated in Lean as :

```
theorem lim_even_character' :
  tendsto (λ (n : ℕ), (1 / ↑(d * p ^ n)) · Σ (i : ℕ) in finset.range (d * p ^ n),
  (asso_dirichlet_character (χ.mul (teichmuller_character_mod_p' p R ^ k))) ↑i * ↑i ^ k) at_top
  (nhds (general_bernoulli_number (χ.mul (teichmuller_character_mod_p' p R ^ k)) k))
```

There were two options for the `finset` used in `finset.sum` : `finset.range` or `zmod (d * p^j)`. While both are mathematically equivalent, they create different kinds of coercions. We worked with both, as explained in Section 3.

The proof of this theorem follows from the proof in Lemma 7.11 of [5]. Majorly, it equates the two sides modulo p^n for a sufficiently large n, and uses the fact that $\lim_{n\to\infty} \frac{1}{dp^n} \sum_{0<a<dp^n;(a,dp)=1} \chi\omega^{-m}(a)a^m = 0$:

```
lemma sum_even_character_tendsto_zero_of_units
  (na' : ∀ (n : ℕ) (f : (zmod n)ˣ → R), ‖Σ i : (zmod n)ˣ, f ‖i ≤ ⨆ (i : (zmod n)ˣ), ‖f ‖i)
  (na : ∀ (n : ℕ) (f : ℕ → R), ‖ Σ (i : ℕ) in finset.range n, f ‖i ≤ ⨆ (i : zmod n), ‖f i.‖val)
  [fact (0 < m)] {k : ℕ} (hk : 1 < k) (hχ : χ.is_even) (hp : 2 < p) :
  tendsto (λ n, Σ (i : (zmod (d * p^n))ˣ), ((asso_dirichlet_character
  (dirichlet_character.mul χ (teichmuller_character_mod_p' p R^k))) i * i^(k - 1)))
  at_top (nhds 0)
```

Notice that this sum is over the `fintype (zmod (d * p^n))`$^\times$. The conditions `na` and `na'` arise from `R` being a non-Archimedean ring. The formalisation is very calculation intensive, and is a good example of a small proof on paper being magnified in Lean, because there are multiple coercions to be dealt with. Also, terms needs to be moved around to be multiplied, deleted and cancelled. Unfortunately, there is no tactic (including `ring` and `simp`) that help with these elementary but lengthy calculations.

3 Construction of the p-adic L-function

3.1 Density of locally constant functions

We denote the Banach space of continuous functions from a profinite space X to a group A by $C(X, A)$, and its subset of locally constant functions by $LC(X, A)$. A function is locally constant if the preimage of any set is open. For any compact Hausdorff totally disconnected space X and a commutative normed ring A, we have proved that $LC(X, A)$ is a dense subset of $C(X, A)$. Formalising this took about 500 lines of code, and is based on the fact that locally compact Hausdorff totally disconnected spaces have a clopen basis :

```
lemma loc_compact_Haus_tot_disc_of_zero_dim {H : Type*} [t2_space H] [locally_compact_space H]
[totally_disconnected_space H] : is_topological_basis {s : set H | is_clopen s}
```

This turned out to be hard to formalise. Given a set s of H, Lean gives a subset V of s the type `V:set s`; however, Lean does not recognize V as a subset of H. As a result, to use `compact_space s` $\longleftrightarrow$ `is_compact (s:set H)`, one must construct `V':set H` to be the image of V under the closed embedding `coe:s → H`. This process must be repeated each time a subset of H, which is also a topological subspace, is considered. Finally, it must be shown that all these coercions match up in the big topological space H.

3.2 Clopen sets of the p-adic integers

Since $\mathbb{Z}_p$ is a profinite space, it is the inverse limit of finite discrete topological spaces $\mathbb{Z}/p^n\mathbb{Z}$ for all n, and has a clopen basis of the form $U_{a,n} := proj_n^{-1}(a)$ for $a \in \mathbb{Z}/p^n\mathbb{Z}$, where $proj_n$ is the canonical projection ring homomorphism `to_zmod_pow n:ℤ_[p] →+* zmod (p ^ n)`. We first define the collection of sets $(U_{a,n})_{a,n}$:

```
  def clopen_basis : set (set ℤ_[p]) :=
  {x : set ℤ_[p] | ∃ (n : ℕ) (a : zmod (p^n)), x = set.preimage (padic_int.to_zmod_pow n) {a} }
```

We show that `clopen_basis` forms a topological basis and that every element is clopen :

```
  theorem clopen_basis_clopen :
    topological_space.is_topological_basis (clopen_basis p) ∧ ∀ x ∈ (clopen_basis p), is_clopen x
```

The mathematical proof is to show that for any ϵ-ball, one can find $U_{a,n}$ inside it. This is true because, given $n \in \mathbb{N}$ and $x \in \mathbb{Z}/p^n\mathbb{Z}$, the preimage of x under `to_zmod_pow n` is the same as the ball centered at x (now considered as an element of $\mathbb{Z}_p$) with radius p^{1-n}. The following lemmas prove useful :

```
lemma appr_spec (n : ℕ) (x : ℤ_[p]) : x - appr x n ∈ (ideal.span {p^n} : ideal ℤ_[p])
lemma has_coe_t_eq_coe (x : ℤ_[p]) (n : ℕ) :
  (((appr x n) : zmod (p^n)) : ℤ_[p]) = ((appr x n) : ℤ_[p])
```

In the latter lemma, the LHS is a coercion of `appr x n`, which has type ℕ, to `ℤ_p`. The RHS is a coercion of `appr x n` to `zmod (p ^ n)` to `ℤ_p`. This statement is not true in general, that is, given any natural number n, it is not true that the lift of n to $\mathbb{Z}_p$ is the same as the composition of its lift to $\mathbb{Z}/p^n\mathbb{Z}$ and $\mathbb{Z}_p$. It works here because the coercion from $\mathbb{Z}/p^n\mathbb{Z}$ to $\mathbb{Z}_p$ is not the canonical lift. It is a composition of a coercion from $\mathbb{Z}/p^n\mathbb{Z}$ to $\mathbb{N}$, which takes $a \in \mathbb{Z}/p^n\mathbb{Z}$ to the smallest natural number in its $\mathbb{Z}/p^n\mathbb{Z}$ equivalence class.

One can similarly show that the sets $U_{b,a,n} := proj_1^{-1}(b) \times proj_{2,n}^{-1}(a)$ form a clopen basis for $\mathbb{Z}/d\mathbb{Z} \times \mathbb{Z}_p$, where $proj_1$ is the first canonical projection on $b \in \mathbb{Z}/d\mathbb{Z}$ and $proj_{2,n}$ the composition of the second projection on $a \in \mathbb{Z}_p$ with $proj_n$ descried above. We call this set `clopen_basis' p d`. Its properties are formalised in `padic_int.clopen_properties.lean`.

3.3 p-adic distributions and measures

In this section, $X = \varprojlim_{i \in \mathbb{N}} X_i$ denotes a profinite space with X_i finite and projection maps $\pi_i : X \to X_i$ and surjective maps $\pi_{ij} : X_i \to X_j$ for all $i \geq j$. We use G to denote an abelian group, A for a commutative normed ring, R for a commutative complete normed ring having a coercion from $\mathbb{Z}_p$, and $LC(X, Y)$ for the space of locally constant functions from X to Y. We fix a prime p and an integer d such that $gcd(d, p) = 1$.

We begin by defining distributions on profinite sets. In Section 12.1 of [5], Washington gives 3 equivalent definitions of a distribution :

1. A system of maps $\phi_i : X_i \to G$ such that $\forall i \geq j$,

$$\phi_j(x) = \sum_{\pi_{ij}(y)=x} \phi_i(y)$$

2. A G-linear function $\phi : LC(X, G) \to G$.
3. A finitely additive function from the compact open sets of X to G.

Since switching between definitions is cumbersome, and(at that point of time), `mathlib` had no notion of thinking about profinite sets as inverse limits of finite sets, we chose to work with the second definition. However, this is already a Type, hence there is no need to redefine it.

The topology on $C(X, A)$ comes from its normed group structure induced by the norm on A : $||f - g|| = sup_{x \in X} ||f(x) - g(x)||$. In fact, this topology is the same as the topology defined on bounded functions on X, since X is a compact space. Since the API for bounded continuous functions on compact spaces was developed at around the same time (created by Oliver Nash), we simply used the existing lemmas along with the equivalence `bounded_continuous_function.equiv_bounded_of_compa`

While showing that, $\forall f \in C(X, A), ||f|| = 0 \implies f = 0$, it suffices to show $||f|| \leq 0 \implies \forall x \in X, ||f(x)|| \leq 0$. We then use the following lemma, which requires a nonempty assumption on X :

```
  theorem cSup_le_iff {α : Type*} [conditionally_complete_lattice α]
  {s : set α} {a : α} (hb : bdd_above s) (ne : nonempty s) :
  Sup s ≤ a ↔ (∀b ∈ s, b ≤ a)
```

The case that X is empty is separately (and trivially) solved.

We can now define p-adic measures. Measures are bounded distributions. Note that the p-adic measures are not to be confused with measures arising from measure theory. The key difference lies in the fact that the clopen sets of a profinite space do not, in general, form a σ-algebra.

```
  def measures [nonempty X] :=
  {φ : (locally_constant X A) →ₗ[A] A // ∃ K : ℝ, 0 < K ∧
  ∀ f : (locally_constant X A), ‖φ f‖ ≤ K * ‖inclusion X A f‖ }
```

The boundedness of the distribution is needed to make the measure continuous :

```
lemma integral_cont (φ : measures X A) : continuous ⇑φ
```

The proof is straightforward : for $b \in LC(X, A)$, given $\epsilon > 0$, there exists a $\delta > 0$ such that for all $a \in LC(X, A)$ with $||b - a|| < \delta$, $||\phi(a) - \phi(b)|| < \epsilon$. Since ϕ is a measure, it suffices to prove that $K * ||inclusion(a - b)|| < \epsilon$. Choosing $\delta = \epsilon / K$ gives the desired result.

The Bernoulli measure is an essential p-adic measure. We make a choice of an integer c with $gcd(c, dp) = 1$, and c^{-1} is an integer such that $cc^{-1} \equiv 1 \mod dp^{2n+1}$. For $x_n \in (\mathbb{Z}/dp^{n+1}\mathbb{Z})^\times$, the (first) Bernoulli measure is defined by

$$E_{c,n}(x_n) = B_1\left(\left\{\frac{x_n}{dp^{n+1}}\right\}\right) - cB_1\left(\left\{\frac{c^{-1}x_n}{dp^{n+1}}\right\}\right)$$

The system $(E_{c,n})_{n\in\mathbb{N}}$ forms a distribution according to the first definition given above. We want to get an equivalent reformulation in terms of the second definition. We know that, since X is compact, every locally constant function can be written in terms of a finite sum of a characteristic function of a basis element multiplied by a constant. Since X is profinite, from the previous section, we know that there exists a clopen basis of the form `set.preimage (padic_int.to_zmod_pow n) a` for $a \in \mathbb{Z}/dp^n\mathbb{Z}$. For the sake of simplicity, let us assume $d = 1$. Thus, for a clopen set $U_{a,n}$:= `set.preimage (padic_int.to_zmod_pow n) a`, we define

$$E_c(\chi_{U_{a,n}}) = E_{c,n}(a)$$

In Lean, this translates to (note that `fract x` represents the fractional part of x) :

```
def E_c (hc : gcd d p = 1) :=
  λ (n : ℕ) (a : (zmod (d * (p^n)))), fract ((a : ℤ) / (d*p^(n + 1))
  - c * fract ((a : ℤ) / (c * (d*p^(n + 1)))) + (c - 1)/2
```

3.4 The Bernoulli measure

Throughout this section, we assume that R is a normed commutative ring which is a $\mathbb{Q}_p$-algebra, $\mathbb{Q}$-algebra, and has a nonarchimedean norm, that is, for any finite set of elements $(r_i)_{i=1}^n$ of R, $\sum_{i=1}^n \parallel r_i \parallel \leq \sup \parallel r_i \parallel$.

The original plan was to define a set of the form :

```
def bernoulli_measure (hc : c.gcd p = 1) :=
 {x : locally_constant (zmod d × ℤ_[p]) R →ₗ[R] R | ∀ (n : ℕ)
    (a : zmod (d * (p^n))), x (char_fn R (is_clopen_clopen_from p d n a)) =
    (algebra_map ℚ R) (E_c p d hc n a) }
```

and to show that it is nonempty. However, this is quite a roundabout way, since one then has to use `classical.some` to extract the Bernoulli measure. We use an elegant way to tackle the problem.

First, we define `eventually_constant_seq` to be the type of sequences satisfying :

```
/-- A sequence has the ‘is_eventually_constant‘ predicate if all the elements of the sequence are
    eventually the same. -/
def is_eventually_constant {α : Type*} (a : ℕ → α) : Prop :=
 { n | ∀ m, n ≤ m → a (nat.succ m) = a m }.nonempty
```

Then, given a locally constant function f from $\mathbb{Z}/d\mathbb{Z} \times \mathbb{Z}_p$ to R, we define the eventually constant sequence `g` to be :

$$g(n) = \sum_{a\in\mathbb{Z}/(d*p^n)\mathbb{Z}} f(a)E_{c,n}(a)$$

for all natural numbers n. We shall look into the proof of g being eventually constant later. We now define the Bernoulli distribution to be the limit of this sequence g.

The proof of this distribution being closed under addition and scalar multiplication follows easily from these definitions and lemmas :

```
/-- The smallest number ‘m‘ for the sequence ‘a‘ such that ‘a n = a (n + 1)‘ for all ‘n ≥ m‘. -/
noncomputable def sequence_limit_index' {α : Type*} (a : @eventually_constant_seq α) : ℕ :=
  Inf { n | ∀ m, n ≤ m → a.to_seq m.succ = a.to_seq m }

/-- The limit of an ‘eventually_constant_seq‘. -/
noncomputable def sequence_limit {α : Type*} (a : @eventually_constant_seq α) :=
  a.to_seq (sequence_limit_index' a)

lemma sequence_limit_eq {α : Type*} (a : @eventually_constant_seq α) (m : ℕ)
  (hm : sequence_limit_index' a ≤ m) : sequence_limit a = a.to_seq m
```

Now, given a locally constant function `f : locally_constant (units (zmod d) × units ℤ_[p]) R`, the `bernoulli_meas` is given by :

```
  bernoulli_distribution p d R (loc_const_ind_fn _ p d f)
```

where `loc_const_ind_fn` is a locally constant function on $\mathbb{Z}/d\mathbb{Z} \times \mathbb{Z}_p$ that takes value f on the units of the domain, and 0 otherwise. We shall skip the proof that this function is locally constant, because it is long and cumbersome. For every open set s, one must look at the case when 0 is contained in s separately. We (prove and) use the fact that `coe : units (zmod d) × units ℤ_[p] → (zmod d) × ℤ_[p]` is an open embedding heavily.

We must now prove that `bernoulli_measure` is indeed a measure, that is, it is bounded. The bound we choose is $1 + \parallel c \parallel + \parallel \frac{c-1}{2} \parallel$. The proof is as follows : let BD denote the Bernoulli distribution, and ϕ denote `loc_const_ind_fn`. We want to show that :

$$\parallel BD(\phi(f)) \parallel \leq K \parallel inclusion(f) \parallel$$

where K is the constant given above. We know that, one can find an n such that

$$\phi(f) = \sum_{a \in \mathbb{Z}/d\mathbb{Z} \times \mathbb{Z}/p^n\mathbb{Z}} \phi(f)(a)\dot{}\chi_{n,a}$$

Since BD is a linear map, it suffices to show that $\parallel BD(\phi(f)(a)\chi_{n,a}) \parallel \leq K \parallel inclusion f \parallel$ for some a (the supremum is achieved since the set is finite). If a is not a unit, $\phi(f)(a) = 0$; by using the linearity of BD, we are done. In the case that a is a unit, it suffices to prove that $\parallel BD(\chi_{n,a}) \parallel \leq K$, and $\parallel \phi(f)(a) \parallel \leq \parallel inclusion f \parallel$. Both of these are easily verified, so we are done.

The implementation is similar, and is heavily dependent on the following lemma :

```
lemma loc_const_eq_sum_char_fn (f : locally_constant ((zmod d) × ℤ_[p]) R)
  (hd : d.gcd p = 1) : ∃ n : ℕ,
  f = Σ a in (finset.range (d * p^n)),
    f(a) · char_fn R (is_clopen_clopen_from p d n a)
```

The machinery used in this proof is similar to the one used to prove that g is eventually constant. Let us have a look at the latter first.

We must first take a look at discrete quotients. The discrete quotient on a topological space is given by an equivalence relation such that all equivalence classes are clopen :

```
structure (X : Type*) [topological_space X] discrete_quotient :=
 (rel : X → X → Prop)
 (equiv : equivalence rel)
 (clopen : ∀ x, is_clopen (set_of (rel x)))
```

The last statement translates to, $\forall x \in X, \{y | y \sim x\}$ is clopen. Given two discrete quotients A and B, $A \leq B$ means that $\forall x, y \in X$, $x \sim_A y \implies x \sim_B y$.

Any locally constant function induces a discrete quotient, since each of its fibers is clopen :

```
def discrete_quotient : discrete_quotient X :=
{ rel := λ a b, f b = f a,
  equiv := ⟨by tauto, by tauto, λ a b c h1 h2, by rw [h2, h1]⟩,
  clopen := λ x, f.is_locally_constant.is_clopen_fiber _ }
```

We now define a function :

```
def F : ℕ → discrete_quotient (zmod d × ℤ_[p]) := λ n,
  ⟨λ a b, to_zmod_pow n a.2 = to_zmod_pow n b.2 ∧ a.1 = b.1, _, _⟩
```

In other words, for $a = (a_1, a_2)$ and $b = (b_1, b_2)$ in $\mathbb{Z}/d\mathbb{Z} \times \mathbb{Z}_p$, $F(n)$ represents the relation

$$a \sim b \iff a_2 (\text{mod } p^n) = b_2 (\text{mod } p^n) a_1 = b_1$$

Then, given a locally constant function f on $\mathbb{Z}/d\mathbb{Z} \times \mathbb{Z}_p$, we have :

```
lemma factor_F (hd : d.gcd p = 1) (f : locally_constant (zmod d × ℤ_[p]) R) :
  ∃ N : ℕ, F N ≤ discrete_quotient f
```

`factor_F` states that, for N large enough, the fibers of f mod p^N are contained in the basic clopen sets of p^N. Here is the proof of `factor_F` : Since X is compact, there exists a finite set t in R such that $X \subseteq \cup_{i \in t} f^{-1}(i)$. Given an open set U of $\mathbb{Z}/d\mathbb{Z} \times \mathbb{Z}_p$, we now define the `bound_set` of U to be
$\{n \in \mathbb{N} \mid \forall a \in U, U_{n,(a\,:\,\text{zmod } p^n)} \subseteq U\}$. The `bound` of U is then defined to be the infimum of the `bound_set U`. Let n be the supremum of `bound` of $f^{-1}(i)$ for all $i \in t$. This is the required `N`.

The proofs of both g being eventually constant now follows. We want to show :

$$\exists N, \forall m \geq N, \sum_{a \in \mathbb{Z}/d*p^{m+1}\mathbb{Z}} f(a) E_{c,m+1}(a) = \sum_{a \in \mathbb{Z}/d*p^m\mathbb{Z}} f(a) E_{c,m}(a)$$

The N we choose is `classical.some (factor_F f) + 1`. We also define the following :

```
/-- Given ‘a ∈ zmod (d * p^n)‘, and ‘n < m‘, the set of all ‘b ∈ zmod (d * p^m)‘ such that ‘b = a
    mod (d * p^n)‘. -/
def equi_class (n m : ℕ) (h : n ≤ m) (a : zmod (d * p^n)) :=
 {b : zmod (d * p^m) | (b : zmod (d * p^n)) = a}
```

Then, we have the following lemma :

```
lemma succ_eq_bUnion_equi_class : zmod’ (d*p^(m + 1)) = (zmod’ (d*p^m)).bUnion
  (λ a : zmod (d * p ^ m), set.to_finset (equi_class m (m + 1)) a)
```

This lemma says that any element of $\mathbb{Z}/dp^{m+1}\mathbb{Z}$ comes from `equi_class m (m + 1) b` for some $b \in \mathbb{Z}/dp^m\mathbb{Z}$. Notice that we use `zmod’` instead of `zmod`, since that has type `finset`, that is, the property of `zmod` being finite is encoded in it. This is needed since we are working with finite sums, hence Lean demands elements of type `finset` instead of `set`.

The proof is now complete with the following lemma :

```
lemma E_c_sum_equi_class (x : zmod (d * p^m)) :
  Σ (y : zmod (d * p ^ (m + 1))) in (λ a : zmod (d * p ^ m),
  set.to_finset ((equi_class m (m + 1)) a)) x, (E_c (m + 1) y) = E_c m x
```

which says that, for $x \in \mathbb{Z}/dp^m\mathbb{Z}$, $E_{c,m}(x) = \sum'_y E_{c,m+1}(y)$, where y belongs to `equi_class m (m + 1) x`.

Finally, we turn to the proof of `loc_const_eq_sum_char_fn`. We want to show that, for any locally constant function f from $\mathbb{Z}/d\mathbb{Z} \times \mathbb{Z}_p$ to R,

$$\exists n, f = \sum_{a \in \mathbb{Z}/dp^n\mathbb{Z}} f(a)\chi_{n,a}$$

We choose n to be `classical.some factor_F f`. The proof now follows easily, since each element belongs to exactly one clopen set of level n. One must go between elements of $\mathbb{Z}/d\mathbb{Z} \times \mathbb{Z}_p$ and $\mathbb{Z}/dp^n\mathbb{Z}$. This requires use of multiple coercions, which makes the proof lengthy.

Notice that `bernoulli_distribution` takes locally constant functions on $\mathbb{Z}/d\mathbb{Z} \times \mathbb{Z}_p$, while `bernoulli_measure` takes locally constant functions on $\mathbb{Z}/d\mathbb{Z}^* \times \mathbb{Z}_p^*$. This had to be done since our clopen basis was defined on $\mathbb{Z}/d\mathbb{Z} \times \mathbb{Z}_p$, and while it is easy to show the same results for the units on paper, it requires a bit of work in Lean.

3.5 p-adic integrals and the p-adic L-function

3.5.1 p-adic Integrals

The last piece in the puzzle is the p-adic integral. We use the same notation as in the previous section. Given a measure μ, and a function $f \in LC(X, R)$, $\int f d\mu := \mu(f)$. As in Theorem 12.1 of [5], this can be extended to a continuous R-linear map :

$$\int_X f d\mu : C(X, R) \to R$$

This follows from the fact that $LC(X, R)$ is dense in $C(X, R)$; as a result, the map *inclusion* : $LC(X, R) \to C(X, R)$ is `dense_inducing`, that is, it has dense range and the topology on $LC(X, R)$ is the one induced by `inclusion` from the topology on $C(X, R)$.

For the linearity of the map, we use `dense_range.induction_on₂`, which states that, given a map $e : \alpha \to \beta$ which has dense range, and a property p (taking two elements as input), in order for any two elements of β to satisfy p, it is sufficient to show that any two elements in the range of e satisfy p; and, the set of elements satifying p is closed with respect to the topology of β :

```
lemma dense_range.induction_on₂ {α β : Type*} [topological_space β] {e : α → β}
{p : β → β → Prop} (he : dense_range e) (hp : is_closed {q:β×β | p q.1 q.2})
(h : ∀a₁ a₂, p (e a₁) (e a₂)) (b₁ b₂ : β) : p b₁ b₂
```

In particular, for every continuous function f, we can take the property p to be preservation of addition (and scalar multiplication respectively) under $\mu(f)$, and `inclusion` to be the map with dense range. The first condition is satisfied due to the following lemma :

```
lemma is_closed_eq [t2_space α] {f g : β → α} (hf : continuous f)
(hg : continuous g) : is_closed {x:β | f x = g x}
```

The second condition follows easily due to the linearity of the measure and the map `inclusion`.

The continuity of the extension of the integral follows from the fact that every measure μ is uniformly continuous. Uniform continuity is a product of the boundedness of the measure : We want to show that, for any measure ϕ, given an $\epsilon > 0$, $\exists \delta > 0$ such that for any $a, b \in LC(X, R)$, with $||a - b|| < \delta$, $||\phi(a) - \phi(b)|| < \epsilon$. Assuming that the constant in the definition of ϕ is K, it is clear that ϵ/K is the required δ. This is the proof of the following lemma :

```
lemma uniform_continuous (φ : measures X A) : uniform_continuous ⇑φ
```

3.5.2 Construction

There are several possible definitions for the p-adic L-functions (fixing an embedding of $\bar{\mathbb{Q}}$ into $\mathbb{C}_p$), including :

1. (Theorem 5.11, [5]) The p-adic meromorphic function $L_p(s,\chi)$ on $\{s \in \mathbb{C}_p \mid |s| < p\}$ obtained by analytic continuation, such that
$$L_p(1-n,\chi) = -(1-\chi\omega^{-n}(p)p^{n-1})\frac{B_{n,\chi\omega^{-n}}}{n}$$
for $n \geq 1$.
2. (Proof of Theorem 5.11, [5]) $L_p(s,\chi) = \sum_{a=1,p\nmid a}^{F} \chi(a)H_p(s,a,F)$, where $H_p(s,a,F)$ is a meromorphic function satifying $H_p(1-n,a,F) = -\frac{F^{n-1}\omega^{-n}(a)}{n}B_n\left(\frac{a}{F}\right)$ for all natural numbers n.
3. (Theorem 12.2, [5]) For $s \in \mathbb{Z}_p$, and Dirichlet character χ with conductor dp^m, with $gcd(d,p) = 1$ and $m \geq 0$, for a choice of $c \in \mathbb{Z}$ with $gcd(c,dp) = 1$:
$$(1-\chi(c) < c >^{s+1})L_p(-s,\chi) = \int_{(\mathbb{Z}/d\mathbb{Z})^\times \times \mathbb{Z}_p^\times} \chi\omega^{-1}(a) < a >^s \, dE_c$$
where $< a >= \omega^{-1}(a)a$, and $b^s = exp(log_p(b))$ (the exponential and logarithm are defined in terms of power series expansions).

It is beyond the scope of this article to explain all the notation in the points above. I chose to define the p-adic L-function as a reformulation of (3). This is because it is the most optimal definition that helps with stating the Iwasawa Main Conjecture.

Instead of using the variable s (which takes values in a subset of $\mathbb{C}_p$), we choose to use an element of the weight space. We replace $< a >^s$ with `w:weight_space A`. The advantage is that our p-adic L-function can now be defined over a more general space.

Given a primitive Dirichlet character χ of character dp^m with $gcd(d,p) = 1$ and $m \geq 0$, we now define the p-adic L-function to be :
$$L_p(w,\chi) := \int_{(\mathbb{Z}/d\mathbb{Z})^\times \times \mathbb{Z}_p^\times} \chi\omega^{p-2}(a)w \, dE_c$$

```
  def p_adic_L_function (hc : gcd c p = 1) :=
  integral (units (zmod d) × units ℤ_[p]) R _
  (bernoulli_measure_of_measure p d R hc)
⟨(λ (a : (units (zmod d) × units ℤ_[p])), ((pri_dir_char_extend p d R) a) *
  (inj (üTeichmller_character p a.snd))^(p - 2) * (w.to_fun a : R)),
    cont_paLf p d R inj w ⟩
```

Note that we have absorbed the constant term given in (3). This was done because Theorem 12.2 lets $L_p(-s,\chi)$ take values in $\mathbb{C}_p$. In a general ring R, as we have chosen, division need not exist. One would then need the factor to be a unit, which may not always happen (for example, consider $R = \mathbb{Q}_p$). Thus, our p-adic L-function differs from the original by a constant factor. However, this factor appears as it is in the Iwasawa Main Conjecture, and can be easily removed if one assumes R to have division.

Lean (implicitly) interprets the placeholder `_` as a proof that $(\mathbb{Z}/d\mathbb{Z})^\times \times \mathbb{Z}_p^\times$ = `units (zmod d) × units ℤ_[p]` is nonempty. Note that `pri_dir_char_extend` extends χ from $(\mathbb{Z}/dp^m\mathbb{Z})^\times$ to $(\mathbb{Z}/d\mathbb{Z})^\times \times \mathbb{Z}_p^\times$ via the restriction map.

By construction, the last term given as input to `integral` has type `C((units (zmod d) × units ℤ_[p]),R)`. This is the term in angular brackets `⟨_, _⟩`, the former is the integrand, and the latter is a proof that the integrand is continuous :

```
  lemma cont_paLf : continuous (λ (a : (units (zmod d) × units ℤ_[p])),
  ((pri_dir_char_extend p d R) a) * (inj (üTeichmller_character p (a.snd)))^(p - 2)
  * (w.to_fun a : R))
```

What remains to be proved is that f is invariant with respect to c. Once these are proved, we can formalise properties of p-adic L-functions. One of the most important properties is, for an integer n with $n \geq 1$,

$$L_p(1-n,\chi) = -(1-\chi\omega^{-n}(p)p^{n-1})\frac{B_{n,\chi\omega^{-n}}}{n}$$

Recall that the function corresponding to $1-n$ is $< a >^{1-n}$. One must show that $< a >^{1-n}$ is an element of the weight space. formalising the first definition would have made showing this result a lot tougher, since showing the analytic continuation of these functions is nontrivial, even on paper.

4 Evaluation at negative integers

We shall now prove that our chosen definition of the p-adic L-function is equivalent to the original one, that is, it takes the same values at negative integers : for $n > 1$,

$$L_p(1-n,\chi) = -(1-\chi\omega^{-n}(p)p^{n-1})\frac{B_{n,\chi\omega^{-n}}}{n}$$

For this section, we assume that R is a normed commutative $\mathbb{Q}_p$-algebra and $\mathbb{Q}$-algebra, which is complete, nontrivial, and has no zero divisors. The scalar multiplication structure obtained from $\mathbb{Q}$ and $\mathbb{Q}_p$ are compatible, given by the condition `is_scalar_tower ℚ ℚ_[p] R` (see [2]). It is also non-archimedean, which are encapsulated by the following hypotheses:

```
na : ∀ (n : ℕ) (f : ℕ → R),
  ‖Σ (i : ℕ) in finset.range n, f ‖i ≤ ⨆ (i : zmod n), ‖f i.‖val
na' : ∀ (n : ℕ) (f : (zmod n)ˣ → R),
  ‖Σ i : (zmod n)ˣ, f ‖i ≤ ⨆ (i : (zmod n)ˣ), ‖f ‖i
```

The prime p is odd, and we choose positive natural numbers d and c which are mutually coprime and are also coprime to p. The Dirichlet character χ has level dp^m, where m is positive. We also assume χ is even and d divides its conductor. Let us first explain why we need the latter condition.

4.1 Factors of the conductor

We explain here why we need d to divide the conductor of χ. In this section, we do not differentiate between the associated Dirichlet character and the Dirichlet character.

Recall that $\chi\omega^{-1}$ actually denotes the Dirichlet character multiplication of χ and ω^{-1}. As mentioned in the previous section, in order to translate between sums on $\mathbb{Z}/dp^n\mathbb{Z}^\times$ and $\mathbb{Z}/dp^n\mathbb{Z}$, one needs that, for all $x \in \mathbb{Z}/dp^n\mathbb{Z}$ such that x is not a unit, $\chi\omega^{-k}(x) = 0$ for all $k > 0$. This is equivalent to saying, $\forall y \in \mathbb{N}$, such that $gcd(y,d) \neq 1$ and $gcd(y,p) \neq 1$, $gcd(y, (\chi\omega^{-k})$`.conductor`$) \neq 1$.

Given coprime natural numbers k_1, k_2 and a Dirichlet character ψ of level k_1k_2, one can find primitive Dirichlet characters ψ_1 and ψ_2 of levels k_1 and k_2 respectively such that $\psi = \psi_1\psi_2$:

```
lemma dirichlet_character.eq_mul_of_coprime_of_dvd_conductor {m n : ℕ}
[fact (0 < m * n)] (χ : dirichlet_character R (m * n)) (hχ : m | χ.conductor)
(hcop : m.coprime n) : ∃ (χ₁ : dirichlet_character R m)
(χ₂ : dirichlet_character R n), χ₁.is_primitive ∧
χ = χ₁.change_level (dvd_mul_right m n) * χ₂.change_level (dvd_mul_left n m)
```

Thus, given $k > 0$, we can find primitive Dirichlet characters χ_1 and χ_2 with conductors z_1 and z_2 such that $z_1|d$ and $z_2|p^m$ and $\chi_1\chi_2 = \chi\omega^{-k}$. The condition that d divides the conductor of χ ensures that $z_1 = d$. As a result, if $gcd(y,d) \neq 1$, then $gcd(y, z_1z_2) \neq 1$, so $\chi\omega^{-k}(y) = 0$.

4.2 Main Result

Note that the same result holds when χ is odd or when $p = 2$, the proofs differ slightly. We shall skip most of the details of the proof, since these are very calculation intensive. We shall instead highlight the key concepts that are used.

The proof consists of two steps : breaking up the integral in the LHS into three sums, and evaluating each of these sums. This is very calculation intensive, and was the longest part of the project. The proof is very similar to the proof of Theorem 12.2 in [5].

Using the fact that the space of locally constant functions is dense in $C((\mathbb{Z}/d\mathbb{Z})^\times \times \mathbb{Z}_p^\times, R)$, we observe that the integral $L_p(1-n,\chi)$ is the same as :

$$L_p(1-n,\chi) = \lim_{j\to\infty} \sum_{a\in(\mathbb{Z}/dp^j\mathbb{Z})^\times} E_{c,j}(\chi\omega^{-1}(a) < a >^{n-1})$$

$$= \lim_{j\to\infty}\left(\sum_{a\in(\mathbb{Z}/dp^j\mathbb{Z})^\times} \chi\omega^{-n}a^{n-1}\left\{\frac{a}{dp^j}\right\} - \sum_{a\in(\mathbb{Z}/dp^j\mathbb{Z})^\times} \chi\omega^{-n}a^{n-1}\left(c\left\{\frac{c^{-1}a}{dp^j}\right\}\right) + \left(\frac{c-1}{2}\right) \sum_{a\in(\mathbb{Z}/dp^j\mathbb{Z})^\times} \chi\omega^{-n}a^{n-1}\right)$$

Going from the first equation to the second took about 600 lines of code, which can be found in `try.lean` (add link). While the proof (on paper) is only a page long, this is very calculation heavy in Lean, because one needs to shift between elements coerced to different types, such as $\mathbb{Z}/(dp^j)\mathbb{Z}$, $\mathbb{Z}/d\mathbb{Z}\times\mathbb{Z}/p^j\mathbb{Z}$, $\mathbb{Z}/d\mathbb{Z}\times\mathbb{Z}_p$, R and their units. Moreover, when each of these types occur as locally constant or continuous functions, one needs to separately prove that each of these functions is also (respectively) locally constant or continuous. Some other difficulties include several different ways to write obtain the same term, such as `equiv.inv_fun`, `equiv.symm`, `ring_equiv.symm` and `ring_equiv.to_equiv.inv_fun`. We have constructed several lemmas to simplify traversing between these terms.

Each of these sums are then evaluated separately in `UVW.lean`. This is dependent on the following lemma :

$$\lim_{j\to\infty}\frac{1}{dp^j}\sum_{i\in\mathbb{Z}/dp^j\mathbb{Z}} \chi\omega^{-n}(i)i^n = B_{n,\chi\omega^{-n}}$$

This is similar to what we need to compute the first sum in (4.2) :

```
tendsto (λ (j : ℕ), Σ (x : (zmod (d * p ^ j))ˣ),
((asso_dirichlet_character (χ.mul (teichmuller_character_mod_p' p R ^ n))) ↑x *
↑(↑x.val) ^ (n - 1)) · (algebra_map ℚ R) (int.fract (↑x / (↑d * ↑p ^ j))))
at_top
(nhds ((1 - (asso_dirichlet_character (χ.mul (teichmuller_character_mod_p' p R ^ n))) ↑p * ↑p ^
  (n - 1)) *
general_bernoulli_number (χ.mul (teichmuller_character_mod_p' p R ^ n)) n))
```

In order to convert this sum into a sum over `finset.range`, we show that :

```
lemma helper_U_3 (x : ℕ) : finset.range (d * p^x) =
set.finite.to_finset (set.finite_of_finite_inter (finset.range (d * p^x))
({x | ¬ x.coprime d})) ∪ ((set.finite.to_finset (set.finite_of_finite_inter
(finset.range (d * p^x)) ({x | ¬ x.coprime p}))) ∪ set.finite.to_finset
(set.finite_of_finite_inter (finset.range (d * p^x))
({x | x.coprime d} ∩ {x | x.coprime p})))
```

This is equivalent to saying :

$$\mathbb{Z}/dp^k\mathbb{Z} \simeq \{x\in\mathbb{N} | gcd(x,d)\neq 1\} \cup \{x\in\mathbb{N}|gcd(x,p)\neq 1\}\cup(\mathbb{Z}/dp^k\mathbb{Z})^\times$$

The condition that d divides the conductor is then used to show that the associated Dirichlet character is 0 everywhere except $(\mathbb{Z}/dp^k\mathbb{Z})^\times$.

Evaluating the middle sum is the most tedious. It is first broken into two sums, so that the previous result can be used. Then, a change of variable from a to $c^{-1}a$ is applied. The variable c is coerced to $\mathbb{Z}/dp^{2k}\mathbb{Z}$, increasing the number of coercions significantly, thus lengthening the calculations.

Finally, the last sum is 0. This follows by substituting a in the summand with $-a$. This is where one uses that χ is even.

There were two ways to do calculations with respect to inequalities on the norm : working with lemmas regarding `filter.tendsto`, or using the following lemma :

```
metric.tendsto_at_top : ∀ {α : Type u_1} {β : Type} [pseudo_metric_space α]
[nonempty β] [semilattice_sup β] {u : β → α} {a : α},
tendsto u at_top (nhds a) ↔ ∀ (ε : ℝ), ε > 0 →
(∃ (N : β), ∀ (n : β), n ≥ N → dist (u n) a < ε)
```

We would like to point out that working with `filter.tendsto` instead of `metric.tendsto_at_top` really simplified calculations. This is because, often, the ε we would choose would be complicated, making our calculations more complicated. As an example, suppose we want to prove :

```
(h : filter.tendsto (λ x, f x) at_top (nhds 0)) → filter.tendsto (λ x, c * f x) at_top (nhds 0)
```

This is a one-line proof using `filter.tendsto_const_mul`. However, if done using `metric.tendsto_at_top`, given $\varepsilon > 0$, we must pick an N such that $\|fx\| < \varepsilon/c$, and use N to complete the proof. Most of such issues can be dealt with using the lemma `filter.tendsto_congr'`.

Hence, we try to avoid using `metric.tendsto_at_top` when possible. The only cases where it is used is when direct inequalities need to be dealt with; this happens precisely when the non-archimedean condition on R needs to be used. Hence, this is a good indicator of where the non-Archimedean condition is needed.

5 Conclusion

5.1 Analysis

We list some of the observations that arose while working on this paper. Throughout this paper, `abbreviation` has played an important part. They help to reduce the number of `def` for a prticular type. A technical difficulty is that one cannot use tactic `rw` to unfold the underlying definition. One must either use `delta`, which often slows compilation time, or make a lemma unfolding the `abbreviation`.

The tactic `rw` does not always work inside sums. As a result, one must use the `conv` tactic to get to the expression inside the sum. While using the `conv` tactic, one is said to be working in `conv` mode. Using the `conv` tactic not only lengthens the proof, but also limits the tactics one can use; the only tactics one can use inside `conv` mode are `rw`, `apply_congr` (similar to `apply`), `simp` and `norm_cast`. Another way around sums is to use `simp_rw`, however, this increases compilation time of the proof. Moreover, `simp_rw` rewrites the lemma as many times as applicable, and is an unsuitable choice if one wants to apply the lemma just once.

Another problem that was recurring was the ratio of implicit to explicit variables. The p-adic L-function, for example, has 19 arguments, of which 7 are explicit, and p, d and R are implicit. This is problematic because 7 is already a large number of hypotheses. Excluding R often means that either Lean guesses or abstracts the correct term, or it asks for them explicitly. In the latter case, one also gets as additional goals all the hypotheses that are dependent on R and implicit, such as `normed_comm_ring R`. Moreover, one cannot get out of `conv` mode unless all these goals are solved. This is difficult since `apply_instance` does not work in `conv` mode. The other alternative is to explicitly provide terms using `@`, however this leads to very large expressions.

Working with Dirichlet characters was challenging. This is because, given $a, b \in \mathbb{N}$ such that $a = b$, Lean does not identify `dirichlet_character R a = dirichlet_character R b`. Tactics such as `subst` also fail. A workaround was putting `h:a = b` as a local hypothesis and then using `congr'`. We then end up with the goal `dirichlet_character R a == dirichlet_character R b`. The API for `heq` suggests that this should be avoided as much as possible. Another workaround was to define and use `dirichlet_character R a ≃ dirichlet_character R b`. However, this would slow down the compilation a lot, sometimes also leading to deterministic timeouts.

5.2 Statistics

When initially completed, this project consisted of around 18,000 lines of code. A major refactor was then done, keeping in mind several of the points mentioned above, specifically using properties of filters instead of metric space calculations wherever possible. All properties regarding particular types (such as Dirichlet characters) were also compiled together in separate files wherever possible. Keeping in mind the spirit of `mathlib`, these properties have been made in the greatest generality as much as possible. After the refactor, there are about 15000 lines of code, put into 10 files (check!).

While most properties regarding Bernoulli numbers and polynomials have been put into `mathlib`, the rest of the work is on a private repository. The author hopes to push the work directly to Lean 4, once the required port is complete.

5.3 Related and future work

There are several projects that require Dirichlet characters and properties of the p-adic integers. These include the project on the formalisation of Fermat's last theorem (link). There is also an effort by Prof David Loeffler which involves formalisation of the classical Dirichlet L-function, that is somewhat dependent on this work.

In the future, the author hopes to be able to work on Iwasawa theory, for which the p-adic L-function is a key ingredient. She also hopes to formalise more properties of Bernoulli numbers, that are a fundamental component of number theory. Mention dangerous instance problem - first made p an implicit variable, then linter said it is a metavariable, so changed it to lemma instead, but it should be an instance.

References

1 Jeremy Avigad, Leonardo de Moura, and Soonho Kong. Theorem proving in lean, Jun 2018. URL: https://kilthub.cmu.edu/articles/journal_contribution/Theorem_Proving_in_Lean/6492902/1, doi:10.1184/R1/6492902.v1.

2 Anne Baanen, Sander R. Dahmen, Ashvni Narayanan, and Filippo A. E. Nuccio Mortarino Majno di Capriglio. A formalization of dedekind domains and class groups of global fields. *Journal of Automated Reasoning*, 66(4):611–637, Nov 2022. doi:10.1007/s10817-022-09644-0.

3 Tomio Kubota and Heinrich-Wolfgang Leopoldt. Eine p-adische Theorie der Zetawerte. I. Einführung der p-adischen Dirichletschen L-Funktionen. *J. Reine Angew. Math.*, 214(215):328–339, 1964.

4 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.

5 Lawrence C. Washington. *Introduction to cyclotomic fields*, volume 83 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, second edition, 1997. doi:10.1007/978-1-4612-1934-7.