Formalization of Formal Group Laws

Wenrong Zou

Born 3rd December 2001 in Fujian, China

5th June 2026

Master's Thesis Mathematics

Advisor: Prof. Dr. Floris van Doorn

Second Advisor: Dr. María Inés de Frutos Fernández

Mathematical Institute

Mathematisch-Naturwissenschaftliche Fakultät der

Rheinischen Friedrich-Wilhelms-Universität Bonn

Contents

Introduction

A *proof assistant* is a software tool that provides a formal language for defining mathematical objects, stating their properties, and constructing proofs to verify those claims. The system then rigorously verifies these proofs down to their fundamental logical axioms.

While traditionally used to verify software correctness, proof assistants are increasingly applied to abstract mathematics—a primary focus of the mathlib community. Through formalization, every definition becomes strictly unambiguous, ensuring that all resulting proofs are virtually guaranteed to be free of errors. The Lean theorem prover is a proof assistant developed principally by Leonardo de Moura. The latest version is called Lean 4. More of the technical details can be found in [10].

Built on this foundation is *mathlib*, a highly active, community-driven initiative aiming to create a comprehensive, unified library of formalized mathematics in Lean. Alongside its mathematical core, the library includes foundational definitions for programming, sustained by daily updates from a dedicated network of contributors.

The Lean Focused Research Organization [14] was established in 2023. This has made Lean an ideal tool not only for students undertaking smaller mathematical projects but also the platform of choice for scientists managing large-scale collaborative endeavors. Early landmark achievements, such as *Liquid Tensor Experiment* [7] and there is a related work about the categorical foundations of formalizated of condensed mathematics in [2].

Between 2024 and 2026, formalized mathematics witnessed several milestone breakthroughs. In the field of analysis, a team led by Floris van Doorn completed the formalization of Carleson's theorem and one of its generalizations. This was a large collaborative project and its blueprint can be found in [3]. In number theory, Fermat's Last Theorem for regular primes was completely formalized in 2025 [4].

Simultaneously, the intersection of auto-formalization and artificial intelligence has emerged as the frontier of growth in this field. An epoch-making development was the verification of the sphere packing problem in late 2025. Building upon Maryna Viazovska's work [18], researchers utilized large language models to assist in code translation, rapidly completing a "sorry-free" formalized verification of optimal sphere packings in dimensions 8 and 24 [11].

Given the results on the formalization of substitution and evaluation of multivariate power series in [6], this thesis aims to formalize the definitions and applications of formal group laws in Lean. A standard reference for formal group laws is [12]. Our mathematical discussion will mainly be based on [12, 5, 15, 17].

In Chapter 1, we will discuss the theory of formal group laws from a purely mathematical perspective. In Chapter 2, we will discuss Lubin–Tate theory, which is one of the most important applications of formal group laws. Chapter 3 provides a brief introduction to the aspects of Lean necessary for understanding the formalization of Chapter 1's content, which we will then present in Chapter 4. The code can be found at https://github.com/WenrongZou/FormalGroupLaws. Throughout this thesis, we will link to code from either our formalization or Mathlib. These links will be marked with this symbol: ↗.

1 Formal group laws

1.1 Definition and basic properties of one-dimensional formal group laws

Let R be a commutative ring throughtout this chapter.

Definition 1.1.1 ([12, Definition 1.1])**.** A *one-dimensional formal group law over* R is a formal power series $F(X,Y) \in R[\![X,Y]\!]$ with the following properties:

1. $F(X,Y)$ is of the form
$$F(X,Y) = X + Y + \sum_{i,j\geq 1} c_{ij} X^i Y^j.$$
2. $F(X,Y)$ satisfies the associativity condition, namely
$$F(F(X,Y),Z) = F(X,F(Y,Z)).$$

For convention, we use formal group law to refer to one-dimensional formal group law. Furthermore, if $F(X,Y)$ satisfies $F(X,Y) = F(Y,X)$, then it is called a *commutative formal group law.*

Remark 1.1.2. Under this definition, one can prove that $F(X,0) = X$ and $F(0,Y) = Y$, which are included as part of the definition in other references.

To see this, let $f(X) = F(X,0)$. Then we have

$$F(F(X,0),0) = F(X,F(0,0)) = F(X,0).$$

which means that $f \circ f = f$. Composing both sides with f^{-1} gives $f = \mathrm{id}$. This proves that $F(X,0) = X$. The equality $F(0,Y) = Y$ follows from an analogous argument.

Therefore, given a formal group law $F(X,Y) \in R[\![X,Y]\!]$, it can be written as $F(X,Y) = X + YG_1(X,Y)$ and $F(X,Y) = Y + XG_2(X,Y)$, for some $G_1(X,Y), G_2(X,Y) \in R[\![X,Y]\!]$.

Example 1.1.3. Two examples of commutative formal group laws are:

$$\mathbb{G}_a(X,Y) = X + Y \quad \text{(the additive formal group law)}$$
$$\mathbb{G}_m(X,Y) = X + Y + XY \quad \text{(the multiplicative formal group law)}$$

These names describe how each power series mimics standard operations with the identity element placed at 0. The *additive formal group law*, $\mathbb{G}_a(X,Y) = X + Y$, is simply standard addition. The *multiplicative formal group law*, $\mathbb{G}_m(X,Y) = X + Y + XY$, is standard multiplication shifted from an identity of 1 to 0. If we define variables centered at 1 as $U = 1 + X$ and $V = 1 + Y$, their standard product is $(1 + X)(1 + Y) = 1 + X + Y + XY$. Subtracting 1 to recenter the identity back to 0 yields exactly the multiplicative law: $X + Y + XY$.

Lemma 1.1.4. *Let $F(X,Y)$ be a formal group law over a ring R. Then there exists a power series $i(X) \in R[\![X]\!]$ such that $F(X,i(X)) = 0$ and $F(i(X),X) = 0$. In particular, the left inverse and right inverse are the same and the inverse is unique.*

Proof. The idea is to write $i(X) = \sum_{j=1}^{\infty} b_j X^j$, and choose b_j inductively so that $F(X,i(X)) = 0$.

First, we choose $b_1 = -1$, so that $F(X,i(X)) = 0 \pmod{\deg 2}$. Assume that $b_1, \dots, b_n$ are already chosen so that $F(X,i(X)) = 0 \pmod{\deg(n+1)}$. Then choose b_{n+1} to be minus the $(n+1)$-th coefficient of $F(X,i_n(X))$, where $i_n(X) = \sum_{j=1}^{n} b_j X^j$. Therefore, $F(X,i_{n+1}(X)) = 0 \pmod{\deg(n+2)}$, where $i_{n+1}(X) = i_n(X) + b_{n+1}X^{n+1}$. Repeating this process, we obtain the desired series $i(X)$ with $F(X,i(X)) = 0$. We can use a similar process to define $i'(X)$ such that $F(i'(X),X) = 0$.

Since $F(X,Y)$ is a formal group law, we have

$$i'(X) = F(i'(X),0) = F(i'(X),F(X,i(X))) = F(F(i'(X),X),i(X)) = F(0,i(X)) = i(X)$$

Therefore, the left inverse and right inverse of X are the same.

For the uniqueness, let $i_1(X), i_2(X) \in R[\![X]\!]$ such that $F(X,i_1(X)) = 0 = F(X,i_2(X))$, then

$$\begin{aligned} i_1(X) = F(i_1(X),0) &= F(i_1(X),F(X,i_2(X))) \\ &= F(F(i_1(X),X),i_2(X)) = F(0,i_2(X)) = i_2(X). \end{aligned}$$

□

Remark 1.1.5. Let S be commutative rings and S is an R-algebra. A formal group law $F(X,Y)$ over R defines a additive group structure on the set of multivariate power series in $S[\![X_I]\!]$ with nilpotent constant coefficient, where I is an idex set. Moreover, let S be a complete topological ring, $F(X,Y)$ defines a additive group structure on the set of topological nilpotents of S. Completeness of S ensure that the evaluation of power series $F(X,Y)$ converges to unique element in S.

Definition 1.1.6 ([12, Definition 1.4])**.** Let $F(X,Y)$ and $G(X,Y)$ be two formal group laws over a ring R. A *homomorphism of formal group laws* $F(X,Y) \to G(X,Y)$ is a power series $\alpha(X) = \sum_{i=1}^{\infty} a_i X^i \in R[\![X]\!]$ such that

$$\alpha(F(X,Y)) = G(\alpha(X),\alpha(Y))$$

An *endomorphism of formal group law* $F(X,Y)$ is a homomorphism $F(X,Y) \to F(X,Y)$. A homomorphism $\alpha(X)$ is an *isomorphism* if there exists a homomorphism $\beta(X) : G(X,Y) \to F(X,Y)$ such that $\alpha(\beta(X)) = \beta(\alpha(X)) = X$.

Lemma 1.1.7. *Let $f(X) = \sum_{i=1}^{\infty} a_i X^i \in R[\![X]\!]$, assume that a_1 is invertible in R, then there is an unique power series $g(X) \in R[\![X]\!]$ such that $f(g(X)) = X = g(f(X))$.*

Proof. The idea is to write $g(X) = \sum_{i=1}^{\infty} b_i X^i$ and choose inductively b_i so that $f(g(X)) = X$.

First, we choose that $b_1 = a_1^{-1}$ so that $f(g(X)) = X \pmod{\deg 2}$. Assume that $b_1, \dots, b_n$ is already chosen, we define c_{n+1} be the $n+1$-th coefficient of $f(\sum_{i=1}^{n} b_i X^i)$ and $b_{n+1} = -a_1^{-1}c_{n+1}$. Then the $n+1$-th coefficient of $f(\sum_{i=1}^{n+1} b_i X^i)$ is $c_{n+1} + a_1 b_{n+1} = 0$. Repeating this process, we get a power series $g(X) \in R[\![X]\!]$ so that $f(g(X)) = X$.

Secondly, since b_1 is invertible in R, then there is a power series $h(X) \in R[\![X]\!]$ so that $g(h(X)) = X$. Precomposing f at the both side of the equality $g(h(X)) = X$, then we have $f(g(h(X))) = f(X)$. Therefore, we conclude that $h(X) = f(X)$, i.e. $g(f(X)) = X$.

Let $g_1(X)$ and $g_2(X)$ be two power series so that $f(g_i(X)) = X = g_i(f(X))$. Then we have that

$$g_1(X) = g_1(f(g_2(X))) = g_2(X).$$

□

Lemma 1.1.8. *The homomorphism $\alpha(X) : F(X,Y) \to G(X,Y)$ is an isomorphism if and only if a_1 is a unit in R, where $\alpha(X) = \sum_{i=1}^{\infty} a_i X^i$.*

Proof. Necessity: given that $\alpha(X)$ is a power series with a_1 a unit, there exists a power series $\beta(X)$ such that $\alpha(\beta(X)) = \beta(\alpha(X)) = X$. We need to prove that this $\beta(X)$ is a formal group homomorphism from $G(X,Y)$ to $F(X,Y)$, namely $\beta(G(X,Y)) = F(\beta(X), \beta(Y))$.

Since $\alpha(X)$ is a homomorphism from $F(X,Y)$ to $G(X,Y)$, we have that

$$\alpha(F(\beta(X), \beta(Y))) = G(\alpha(\beta(X)), \alpha(\beta(Y))) = G(X,Y).$$

The desired equality follows by composing both sides with $\beta(X)$ and using $\beta \circ \alpha = \mathrm{id}$.

Sufficiency: assume that $\alpha(X)$ is an formal group isomorphism, then there is formal group homomorphism $\beta(X) : G(X) \to F(X)$ such that $\alpha(\beta(X)) = X$. If we denote that $\alpha(X) = \sum_{i=1}^{\infty} a_i X^i$ and $\beta(X) = \sum_{i=1}^{\infty} b_i X^i$, then the coefficient of X on the $\alpha(\beta(X))$ is $a_1 b_1$. Therefore, we conclude that a_1 is invertible in R, since $a_1 b_1 = 1$. □

One of the most important endomorphisms of a commutative formal group is the n-series.

Definition 1.1.9 (n-series, [12, Definition 1.4.5])**.** Let $F(X,Y)$ be a commutative formal group law over R, for natural number n, we define the n-series $[n](X) \in R[\![X]\!]$ inductively as follows:

1. $[0](X) = 0$.
2. $[n+1](X) = F([n](X), X)$.

Moreover, $[-n](X) \coloneqq i([n](X))$, where $i(X)$ is the power series in Lemma 1.1.4.

Proposition 1.1.10. *For any natural number n, the n-series $[n](X)$ is a formal group endomorphism of $F(X,Y)$, i.e. $[n](F(X,Y)) = F([n](X), [n](Y))$.*

Proof. We use induction on n to prove it. For the base cases $n = 0, 1$, the result follows immediately. Assume that the result holds for some natural number k. Now consider the case $k+1$.

$$\begin{aligned}
[k+1]F(X,Y) &= F([k]F(X,Y), F(X,Y)) \\
&= F(F([k](X), [k](Y)), F(Y,X)) \\
&= F([k](X), F([k](Y), F(Y,X))).
\end{aligned}$$

Next, we focus on the term $F([k](Y), F(Y,X))$.

$$F([k](Y), F(Y,X)) = F(F([k](Y), Y), X) = F([k+1](Y), X) = F(X, [k+1](Y)).$$

Hence, we have

$$\begin{aligned}
[k+1]F(X,Y) &= F([k](X), F(X, [k+1](Y))) \\
&= F(F([k](X), X), [k+1](Y)) \\
&= F([k+1](X), [k+1](Y)).
\end{aligned}$$

□

Remark 1.1.11. By induction, we deduce that $[n](X) = nX + O(X^2)$. As a result, for a ring R of characteristic p, we have $[p](X) = aX^k + O(X^{k+1})$ for some $k > 1$.

Example 1.1.12. The p-series of additive formal group law is $[p]_{\mathbb{G}_a}(X) = pX$. The p-series of multiplicative formal group law is $[p]_{\mathbb{G}_m}(X) = (1+X)^p - 1$.

Proposition 1.1.13. *Assume that R is of characteristic p and nontrivial. Then the additive formal group law $\mathbb{G}_a$ and the multiplicative formal group law $\mathbb{G}_m$ over R are not isomorphic.*

Proof. Assume that $\mathbb{G}_a$ is isomorphic to $\mathbb{G}_m$, then there is a formal group isomorphism $f(X)$ from $\mathbb{G}_a$ to $\mathbb{G}_m$. By the definition of homomorphism of formal group law, we have that

$$f([p]_{\mathbb{G}_a}(X)) = [p]_{\mathbb{G}_m}(f(X)).$$

Note that

$$[p]_{\mathbb{G}_a}(X) = pX = 0 \quad \text{and} \quad [p]_{\mathbb{G}_m} = (1+X)^p - 1 = X^p.$$

Then we have

$$f([p]_{\mathbb{G}_a}(X)) = f(0) = 0 \quad \text{and} \quad [p]_{\mathbb{G}_m}(f(X)) = (f(X))^p.$$

Therefore, we have the equality

$$(f(X))^p = 0. \tag{1.1.13.1}$$

If we write $f(X) = \sum_{n=1}^{\infty} a_n X^n$, then a_1 is invertible in R, because f is a formal group isomorphism. By equation (1.1.13.1), we have that $a_1^p = 0$, which contradicts to a_1 is invertible in R.

□

Proposition 1.1.14. *Assume R is of characteristic p. Then $[p](X) = 0$ or $[p](X) = aX^{p^n} + O(X^{p^n+1})$.*

Proof. Assume that $[p](X) \neq 0$. Let k be the order of $[p](X)$, i.e., the smallest natural number i such that the i-th coefficient of $[p](X)$ is nonzero, and let a be this leading coefficient. It is enough to show that k is of the form p^n for some n.

Since $[p](X)$ is an endomorphism of the formal group law F, we have

$$[p](F(X,Y)) = F([p](X), [p](Y)).$$

Reducing both sides of the above equation modulo degree $k+1$, we obtain the following congruences:

$$[p](F(X,Y)) \equiv a(X+Y)^k \pmod{\deg(k+1)},$$
$$F([p](X), [p](Y)) \equiv aX^k + aY^k \pmod{\deg(k+1)}.$$

Comparing the two sides, we have $a(X+Y)^k \equiv a(X^k + Y^k) \pmod{\deg(k+1)}$. Since $a \neq 0$ and R has characteristic p, it follows that the intermediate binomial coefficients must vanish, which means:

$$\forall i \in [1, k-1], \quad p \mid \binom{k}{i}. \tag{1.1.14.1}$$

Recall Lucas's theorem: let p be a prime number, and let b, m, n be natural numbers. Then

$$\binom{p^b m}{p^b n} \equiv \binom{m}{n} \pmod{p}.$$

If k is not a power of p, then there exists a natural number $m > 1$ such that $k = p^t m$ and $p \nmid m$. Taking $i = p^t$ in relation (1.1.14.1), we have

$$p \mid \binom{p^t m}{p^t}.$$

By Lucas's theorem, we have

$$\binom{p^t m}{p^t} \equiv \binom{m}{1} \equiv m \pmod{p}.$$

This yields the contradiction that $m \equiv 0 \pmod{p}$, since we originally assumed $p \nmid m$. Therefore, k must be a power of p. □

Definition 1.1.15 ([12, Definition 1.5])**.** Let $F(X,Y) = X + Y + \sum_{i,j\geq 1} c_{ij} X^i Y^j$ be a formal group law over a ring R, and let $\phi : R \to R'$ be a ring homomorphism. By applying ϕ to the coefficients of $F(X,Y)$, we obtain a formal group law over R'

$$\phi_* F(X,Y) = X + Y + \sum_{i,j\geq 1} \phi(c_{ij}) X^i Y^j.$$

Remark 1.1.16. There is a functor from the category of commutative rings to **Set**, which maps a commutative ring R to all formal group laws over R. Given a ring homomorphism $\phi : R \to S$, the definition above provides a map from formal group laws over R to formal group laws over S.

1.2 The functional equation lemma

In this section, we discuss the functional equation lemma for formal group laws. This lemma can be used to construct non-isomorphic formal group laws and a universal formal group law.

To begin, we need the following data. Let K be a commutative ring, R be a subring of K, $\sigma : K \to K$ be a ring homomorphism, I be an ideal of R, p be a prime number, q be a power of this prime number, and $s_1, s_2, \ldots$ be elements of K.

These ingredients are assumed to satisfy the following conditions:

1. $\sigma(R) \subset R$,
2. $\forall a \in R, \sigma(a) \equiv a^q \pmod{I}$,
3. $p \in I$,
4. $\forall i,\ s_i I \subset R$,
5. $\forall b \in K, r \in \mathbb{N},\ I^r b \subset I \Rightarrow I^r \sigma(b) \subset I$.

Definition 1.2.1 ([12, Definition 2.1.8])**.** Let $g(X) = \sum_{i=1}^{\infty} b_i X^i$ be a power series in one variable with coefficients in R. Define the *recursive power series* $f_g(X)$ to be the unique solution of the following functional equation

$$f_g(X) = g(X) + \sum_{i=1}^{\infty} s_i \sigma_*^i f_g(X^{q^i}) \tag{1.2.1.1}$$

Remark 1.2.2. Denote $f_g(X) = \sum_{i=1}^{\infty} a_i X^i$ and let $n = q^r m$, where q does not divide m. By comparing coefficients on both sides, there is a recursive formula for the coefficients of $f_g(X)$:

$$a_n = b_n + s_1\sigma(a_{n/q}) + \cdots + s_r\sigma^r(a_{n/q^r}) = b_n + \sum_{i=1}^{r} s_i\sigma^i(a_{n/q^i}) \tag{1.2.2.1}$$

In particular, if q does not divide n, then $a_n = b_n$.

Theorem 1.2.3 (Functional Equation Lemma, [12, Chap. 1, Theorem 2.2])**.** *Let $g(X)$, $h(X)$ be two power series in one variable over R and suppose that the coefficient of X in $g(X)$ is invertible in R. Then*

1. *the power series $F_g(X,Y) = f_g^{-1}(f_g(X) + f_g(Y))$ has its coefficients in R,*
2. *the power series $f_g^{-1}(f_h(X))$ has its coefficients in R,*
3. *there is a power series $\hat{h}(X)$ over R such that $f_g(h(X)) = f_{\hat{h}}(X)$,*
4. *let $\alpha(X) \in R[\![X]\!]$, $\beta(X) \in K[\![X]\!]$, and let r be a positive natural number, then we have*

$$\begin{aligned} &\alpha(X) \equiv \beta(X) \pmod{I^r R[\![X]\!]} \\ \Leftrightarrow\ &f_g(\alpha(X)) \equiv f_g(\beta(X)) \pmod{I^r R[\![X]\!]}. \end{aligned}$$

Remark 1.2.4. Due to equation (1.2.2.1), we have that the coefficient of X in $f_g(X)$ and $g(X)$ is the same, denoted as a_1. Thus, the coefficient of X in $f_g(X)$ is invertible in R and the power series $f_g^{-1}(X)$ is well defined by Lemma 1.1.7. By computations, we have that the coefficient of X and Y in $F_g(X,Y)$ is $a_1^{-1}a_1 = 1$, since the coefficient of X in $f_g^{-1}(X)$ is a_1^{-1}. Moreover, we notice that

$$\begin{aligned} F_g(F_g(X,Y),Z) = f_g^{-1}(f_g(F_g(X,Y)) + f_g(Z)) &= f_g^{-1}(f_g(f_g^{-1}(f_g(X) + f_g(Y))) + f_g(Z)) \\ &= f_g^{-1}(f_g(X) + f_g(Y) + f_g(Z)). \end{aligned}$$

Similarly,

$$\begin{aligned} F_g(X,F_g(Y,Z)) = f_g^{-1}(f_g(X) + f_g(F_g(Y,Z))) &= f_g^{-1}(f_g(X) + f_g(f_g^{-1}(f_g(Y) + f_g(Z)))) \\ &= f_g^{-1}(f_g(X) + f_g(Y) + f_g(Z)). \end{aligned}$$

Therefore, we deduce that $F_g(X,Y)$ is a commutative formal group law over K. Moreover, according to the functional equation lemma, $F_g(X,Y)$ is a commutative formal group law over the ring R.

Before discussing the proof of the functional equation integrality lemma, we first present several technical lemmas.

Lemma 1.2.5 ([12, Chap. 1, Lemma 2.4.1])**.** *Let $f_g(X) = \sum_{i=1}^{\infty} a_n X^n$, and let $n = q^r m$ where m is not divisible by q, then $a_n I^r \subset R$.*

Proof. By the assumption (5) and induction, we have that for any given natural numbers j, r and any $b \in K$,

$$I^r b \subset I \Rightarrow I^r \sigma^j(b) \subset I. \tag{1.2.5.1}$$

We proceed by induction on r (the multiplicity of q in n).

Assume that the result holds for all numbers less than $r+1$; now consider those n such that the multiplicity of q in n is $r+1$. According to our recursive formula, the coefficient a_n can be expressed as

$$a_n = b_n + \sum_{i=1}^{r+1} s_i \sigma^i(a_{n/q^i}).$$

Combining the result (1.2.5.1) and the assumption (4), it is enough to prove

$$\forall i \in [1, r+1],\ a_{n/q^i} I^{r+1} \subset I.$$

According to the induction hypothesis and the fact that the multiplicity of q in n/q^i is less than $r+1$, we have $\forall i \in [1, r+1]$, $a_{n/q^i} I^r \subset R$. This implies that $a_n \in I$. □

Lemma 1.2.6. *Let S be a commutative ring, q be a power of a prime number p, for any $x, y \in S$, then there is an element $a \in S$ such that*

$$(x+y)^q = x^q + y^q + pxya.$$

Proof. Note that

$$(x+y)^q = x^q + y^q + \sum_{i=1}^{q-1} \binom{q}{i} x^i y^{q-i}.$$

Then the lemma follows from the fact that $p \mid \binom{q}{i}$ for all $i = 1, \ldots, n-1$. □

Lemma 1.2.7. *Let $F(X_J), G(X_J) \in R[\![X_J]\!]$, where J is a finite index set, let r be the multiplicity of q in n, namely $n = q^r m$ with $q \nmid m$, and let i be a positive natural number. Assume that $F(X_J) \equiv G(X_J) \pmod{I^i}$, then we have*

$$F(X_J)^n \equiv G(X_J)^n \pmod{I^{i+r}}.$$

Proof. We prove that $F(X_J)^{q^r} \equiv G(X_J)^{q^r} \pmod{I^{i+r}}$, from which then the lemma follows by taking m-th powers on both sides.

We do induction on r. The case for $r = 0$ is trivial. Assume that the result holds for r, namely $F(X_J)^{q^r} \equiv G(X_J)^{q^r} \pmod{I^{i+r}}$. Therefore, $F(X_J)^{q^r} = G(X_J)^{q^r} + H(X_J)$, where all coefficients of $H(X_J)$ are in I^{i+r}. Due to Lemma 1.2.6, there is a power series $\alpha(X_J) \in R[\![X_J]\!]$ such that

$$(G(X_J)^{q^r} + H(X_J))^q = (G(X_J)^{q^r})^q + H(X_J)^q + pG(X_J)^{q^r} H(X_J)\alpha(X_J).$$

Then we have the following equalities

$$\begin{aligned} F(X_J)^{q^{r+1}} = (F(X_J)^{q^r})^q &= (G(X_J)^{q^r} + H(X_J))^q \\ &= (G(X_J)^{q^r})^q + H(X_J)^q + pG(X_J)^{q^r} H(X_J)\alpha(X_J) \end{aligned}$$

Our result is given by the following two congruences: $H(X_J)^q \equiv 0 \pmod{I^{i+r+1}}$ and $pH(X_J) \equiv 0 \pmod{I^{i+r+1}}$. □

Lemma 1.2.8 (Generalization of [12, Chap. 1 Lemma 2.4.2]). *Let n, l be natrual numbers, q be a power of prime number p, J be a finite index set, and let $n = q^r m$ with $q \nmid m$. For all $G(X_J) \in R[\![X_J]\!]$, we have*

$$G(X_J)^{nq^l} \equiv (\sigma_*^l G(X_J^{q^l}))^n \pmod{I^{r+1}}.$$

Proof. The case for $I = R$ is trivial. Now, we assume that $I \subsetneq R$. Then R/I has characteristic p, since $p \in I$. Given property 2, then we have

$$G(X_J)^{q^l} \equiv \sigma_*^l G(X_J^{q^l}) \pmod{I}$$

We obtain the goal congruence from the above congruence and Lemma 1.2.7. □

Proof of the Functional Equation Lemma 1.2.3. For simplicity, we write $F(X, Y)$ and $f(X)$ for $F_g(X, Y)$ and $f_g(X)$ respectively. Let $F_i(X, Y)$ be the homogeneous component of $F(X, Y)$ of degree i. We denote $g(X) = \sum_{i=1}^{\infty} b_i X^i$ and $f(X) = \sum_{n=1}^{\infty} a_n X^n$ throughout the proof of this theorem.

Proof of part (1): Since $f(X) \equiv b_1 X \pmod{\deg 2}$ and $f^{-1}(X) \equiv b_1^{-1} X \pmod{\deg 2}$, we have $F(X, Y) \equiv X + Y \pmod{\deg 2}$. This implies that the coefficients of $F_1(X, Y)$ lie in R. We now proceed by induction to prove that $F_n(X, Y) \in R[\![X, Y]\!]$.

Suppose that $F_1(X, Y), \ldots, F_{n-1}(X, Y) \in R[\![X, Y]\!]$. Note that for $r \geq 2$, we have

$$(F(X, Y))^r \equiv (F_1(X, Y) + \cdots + F_{n-1}(X, Y))^r \pmod{\deg(n+1)}.$$

This follows from the fact that the difference between the expressions inside the parentheses consists only of terms of degree at least n. Upon taking the r-th power, any term in the expansion involving this difference will have a degree of at least $n + 1$.

By the definition of $F(X, Y) = f^{-1}(f(X) + f(Y))$, we have

$$f(F(X, Y)) = f(X) + f(Y)$$
$$\sigma_*^i f(\sigma_*^i F(X, Y)) = \sigma_*^i f(X) + \sigma_*^i f(Y).$$

Recall that f and g satisfy recursion formula (1.2.1.1),

$$f(X) = g(X) + \sum_{n=1}^{\infty} s_n \sigma_*^n f(X^{q^n}).$$

By substituting $F(X, Y)$ into X and writing $f(X) = \sum_{n=1}^{\infty} a_n X^n$, we obtain that

$$\begin{aligned} f(F(X, Y)) &= g(F(X, Y)) + \sum_{n=1}^{\infty} s_n \sigma_*^n f(F(X, Y)^{q^n}) \\ &= g(F(X, Y)) + \sum_{i=1}^{\infty} s_i \sum_{n=1}^{\infty} \sigma^i(a_n)(F(X, Y))^{q^i n}. \end{aligned}$$

Next, by Lemma 1.2.8, we have

$$s_i \sigma^i(a_n) F(X, Y)^{q^i n} \equiv s_i \sigma^i(a_n)(\sigma_*^i F(X^{q^i}, Y^{q^i}))^n \pmod{(R, \deg(n+1))}.$$

The notation $\pmod{(R, \deg(n+1))}$ indicates that the two expressions are congruent up to terms of degree n, and that their corresponding coefficients are congruent modulo the subring R. Then, modulo $(R, \deg(n+1))$, we have

$$\begin{aligned} f(F(X,Y)) &\equiv g(F(X,Y)) + \sum_{i=1}^{\infty} s_i \sum_{n=1}^{\infty} \sigma^i(a_n)(\sigma_*^i F(X^{q^i}, Y^{q^i}))^n \\ &\equiv g(F(X,Y)) + \sum_{i=1}^{\infty} s_i \sigma_*^i f(\sigma_*^i F(X^{q^i}, Y^{q^i})) \\ &\equiv g(F(X,Y)) + \sum_{i=1}^{\infty} s_i (\sigma_*^i f(X^{q^i}) + \sigma_*^i f(Y^{q^i})) \\ &\equiv g(F(X,Y)) + f(X) + f(Y) - g(X) - g(Y). \end{aligned}$$

Since $g(X) \equiv b_1 X \pmod{\deg 2}$, then

$$g(F(X,Y)) \equiv b_1 F_n(X,Y) \pmod{(R, \deg(n+1))}.$$

Using the equation $f(F(X,Y)) = f(X) + f(Y)$ and the congruence above, we have

$$\begin{aligned} f(X) + f(Y) &= f(F(X,Y)) \\ &\equiv g(F(X,Y)) + f(X) + f(Y) - g(X) - g(Y) \\ &\equiv b_1 F_n(X,Y) + f(X) + f(Y) - g(X) - g(Y) \pmod{(R, \deg(n+1))}. \end{aligned}$$

Since $g(X) \in R[\![X]\!]$, we have $g(X) \equiv 0 \pmod{R}$. This implies that

$$b_1 F_n(X,Y) \equiv 0 \pmod{(R, \deg(n+1))}.$$

Because b_1 is invertible in R, we conclude that $F_n(X,Y) \equiv 0 \pmod{(R, \deg(n+1))}$. By induction, the homogeneous component of degree n lies in $R[\![X,Y]\!]$ for all n. It follows that $F(X,Y) \in R[\![X,Y]\!]$.

Proof of part (2): Write $G(X) = f_g^{-1}(f_h(X))$. Let $G_i(X)$ be the homogeneous component of degree i in $G(X)$. We proceed by induction to prove that for all i, the coefficients of $G_i(X)$ lie in R. Assume that this result holds for all $i < n$. Now consider the case $i = n$. Modulo $(R, \deg(n+1))$, we have

$$\begin{aligned} f_h(X) &= f(G(X)) \\ &= g(G(X)) + \sum_{i=1}^{\infty} s_i \sigma_*^i f(G(X)^{q^i}) \\ &= g(G(X)) + \sum_{i=1}^{\infty} s_i \sigma_*^i \sum_{m=1}^{\infty} a_m G(X)^{q^i m} \\ &\equiv g(G(X)) + \sum_{i=1}^{\infty} s_i \sum_{m=1}^{\infty} \sigma_*^i(a_m)(\sigma_*^i G(X^{q^i}))^m \\ &= g(G(X)) + \sum_{i=1}^{\infty} s_i \sigma_*^i f(G(X^{q^i})) \\ &= g(G(X)) + \sum_{i=1}^{\infty} s_i \sigma_*^i (f_h(X^{q^i})) \\ &= g(G(X)) + f_h(X) - h(X). \end{aligned}$$

It follows that $g(G(X)) \equiv h(X) \pmod{(R, \deg(n+1))}$. We then have

$$h(X) \equiv g(G(X)) \equiv b_1 G_n(X) \pmod{(R, \deg(n+1))}.$$

Because b_1 is invertible, we conclude that the coefficients of $G_n(X)$ lie in R.

Proof of part (3): Let $\hat{f}(X) = f(h(X))$. By the functional equation (1.2.1.1), it suffices to show that the coefficients of $\hat{f}(X) - \sum_{i=1}^{\infty} s_i \sigma_*^i \hat{f}(X^{q^i})$ lie in R. Working modulo R, we have the following sequence of equalities and congruences:

$$\begin{aligned}
\hat{f}(X) - \sum_{i=1}^{\infty} s_i \sigma_*^i \hat{f}(X^{q^i}) &= f(h(X)) - \sum_{i=1}^{\infty} s_i \sigma_*^i f(\sigma_*^i h(X^{q^i})) \\
&= f(h(X)) - \sum_{i=1}^{\infty} s_i \sum_{n=1}^{\infty} \sigma^i(a_n)(\sigma_*^i h(X^{q^i}))^n \\
&\equiv f(h(X)) - \sum_{i=1}^{\infty} s_i \sum_{n=1}^{\infty} \sigma^i(a_n)(h(X)^{q^i n}) \\
&= f(h(X)) - \sum_{i=1}^{\infty} s_i \sigma_*^i f(h(X)^{q^i}) \\
&= g(h(X)) \equiv 0.
\end{aligned}$$

Since the expression reduces to 0 modulo R, we conclude that all its coefficients are indeed in R.

Proof of part (4): For the forward implication, assume that $\alpha(X) \equiv \beta(X) \pmod{I^r R[\![X]\!]}$. Then $\beta(X) \in R[\![X]\!]$. By Lemma 1.2.7, we have $\alpha(X)^n \equiv \beta(X)^n \pmod{I^{r+t}}$, where t is the multiplicity of q in n. Denote $f(X) = \sum_{i=1}^{\infty} a_i X^i$. By Lemma 1.2.5, we have $a_n \alpha(X)^t \equiv a_n \beta(X)^t \pmod{I^r}$, where t is the multiplicity of q in n. Therefore, $f(\alpha(X)) \equiv f(\beta(X)) \pmod{I^r}$.

For the reverse implication, assume that $f(\alpha(X)) \equiv f(\beta(X)) \pmod{I^r R[\![X]\!]}$. First, we prove the following result:

$$\alpha(X) \equiv 0 \pmod{I^r} \implies f^{-1}(\alpha(X)) \equiv 0 \pmod{I^r}. \tag{1.2.8.1}$$

Let $\gamma(X) = f^{-1}(\alpha(X))$; then $\alpha(X) = f(\gamma(X))$. Let $\gamma_n(X)$ be the truncation of $\gamma(X)$ at degree $n+1$. We proceed by induction to prove that $\gamma_n(X) \in I^r R[\![X]\!]$ for all n. The cases for $k = 1, 2$ are trivial. Assume that the result holds for k, namely $\gamma_k(X) \equiv 0 \pmod{I^r}$. Now consider $\gamma_{k+1}(X)$. By the functional equation and the definition of $\gamma(X)$, we have

$$\alpha(X) = g(\gamma(X)) + \sum_{i=1}^{\infty} s_i \sigma_*^i f(\gamma(X)^{q^i}). \tag{1.2.8.2}$$

We claim that:

$$\sum_{i=1}^{\infty} s_i \sigma_*^i f(\gamma(X)^{q^i}) \in I^r R[\![X]\!] \pmod{\deg(k+1)}. \tag{1.2.8.3}$$

Comparing the coefficients of degree $k+1$ on both sides of equation (1.2.8.2), we obtain $b_1 c_{k+1} \in I^r$, where $\gamma(X) = \sum_{i=1}^{\infty} c_i X^i$ and $g(X) = \sum_{i=1}^{\infty} b_i X^i$. Since b_1 is a unit in R, it follows that $c_{k+1} \in I^r$.

Proof of Claim (1.2.8.3)*:* By Lemma 1.2.7 and the induction hypothesis, we have $\gamma_k(X)^n \equiv 0 \pmod{I^{r+t}}$, where t is the multiplicity of q in n. By assumption (4), it suffices to show that

$$\sigma_*^i f(\gamma(X)^{q^i}) \in I^{r+1}R[\![X]\!] \pmod{\deg(k+2)},$$

which is equivalent to

$$\sigma^i(a_n)\gamma(X)^{q^{in}} \in I^{r+1}R[\![X]\!] \pmod{\deg(k+2)}. \tag{1.2.8.4}$$

Since $\gamma(X)^{q^{in}} \equiv \gamma_k(X)^{q^{in}} \pmod{\deg(k+2)}$, it follows from Lemma 1.2.5 that the congruence (1.2.8.4) holds. This proves the implication (1.2.8.1).

By assumption, $f(\beta(X)) - f(\alpha(X)) \in I^r R[\![X]\!]$. Thus, applying (1.2.8.1), we have

$$\delta(X) := f^{-1}(f(\beta(X)) - f(\alpha(X))) \in I^r R[\![X]\!].$$

Then

$$f(\beta(X)) = f(\delta(X)) + f(\alpha(X)), \quad \text{so that}$$
$$\beta(X) = f^{-1}(f(\delta(X)) + f(\alpha(X))).$$

By Remark 1.2.4, $f^{-1}(f(X) + f(Y))$ is a formal group law. Hence, we can write

$$\beta(X) = f^{-1}(f(\delta(X)) + f(\alpha(X))) = \alpha(X) + \delta(X)G(\delta(X), \alpha(X))$$

for some $G(X, Y) \in R[\![X, Y]\!]$. Since $\delta(X) \in I^r R[\![X]\!]$. We conclude that

$$\beta(X) \equiv \alpha(X) \pmod{I^r R[\![X]\!]}.$$

□

1.3 Universal formal group laws

In this section, we discuss the construction of universal formal group laws and their applications.

Definition 1.3.1 ([12, Chap. 1, Definition 1.5.2])**.** Let L be a commutative ring. A formal group law $F(X, Y)$ over L is called a *universal formal group law* if, for any formal group law $G(X, Y)$ over a commutative ring R, there exists a unique ring homomorphism $\phi : L \to R$ such that $\phi_* F(X, Y) = G(X, Y)$.

Remark 1.3.2. The ring L is uniquely determined up to isomorphism. Indeed, let $F(X, Y)$ and $G(X, Y)$ be universal formal group laws over rings L and L', respectively. Then there exist ring homomorphisms $\phi : L \to L'$ and $\psi : L' \to L$ such that $\phi_* F(X, Y) = G(X, Y)$ and $\psi_* G(X, Y) = F(X, Y)$. By uniqueness, we deduce that $\phi \circ \psi = \mathrm{id}_{L'}$ and $\psi \circ \phi = \mathrm{id}_L$. In particular, ϕ and ψ are isomorphisms.

We now show that there exists a universal formal group law over some ring L. Let $\tilde{L} = \mathbb{Z}[C_{i,j}]$, where $C_{i,j}$ are indeterminates with $i, j \in \mathbb{N}_{>0}$. Consider the power series $F_C(X, Y) = X + Y + \sum_{i,j\geq 1} C_{i,j} X^i Y^j$. We write

$$F_C(F_C(X, Y), Z) - F_C(X, F_C(Y, Z)) = \sum_{i,j,k\geq 1} P_{i,j,k}(C) X^i Y^j Z^k,$$

where $P_{i,j,k}(C)$ is a polynomial in the variables $C_{i,j}$. Let I be the ideal of $\tilde{L}$ generated by $C_{i,j} - C_{j,i}$ and $P_{i,j,k}(C)$, where $i, j, k \in \mathbb{N}_{>0}$. Define $L = \tilde{L}/I$, and let $F(X, Y) = p_* F_C(X, Y)$, where $p : \tilde{L} \to L$ is the natural quotient map.

Given a formal group law $F(X,Y)$ over a ring R, write $F(X,Y) = \sum_{i,j\geq 0} a_{i,j}X^iY^j$. Then there exists a unique ring homomorphism $\phi : L \to R$ such that $\phi(C_{i,j}) = a_{i,j}$ and $\phi_* F_C(X,Y) = F(X,Y)$. The uniqueness follows from the condition $\phi_* F_C(X,Y) = F(X,Y)$.

However, the structure of the ring L and the formal group law $F(X,Y)$ constructed above is not immediately clear. In fact, one can prove that L is isomorphic to $\mathbb{Z}[U] := \mathbb{Z}[U_2, U_3, \ldots]$, where U_i are indeterminates, and that $F(X,Y) \in \mathbb{Z}[U][\![X,Y]\!]$ is of the form $f^{-1}(f(X)+f(Y))$ for some $f(X) \in \mathbb{Q}[U][\![X]\!]$. The integrality of the coefficients of $F(X,Y)$ then follows from the functional equation lemma.

Remark 1.3.3. There is a functor from the category of commutative rings to **Set**, which assigns to each commutative ring R the set of formal group laws over R, and assigns to each ring homomorphism $\phi : R \to S$ the pushforward map from formal group laws over R to formal group laws over S, as defined in Definition 1.1.15. This is a covariant functor and is corepresented by the ring L and the formal group law $F_C(X,Y)$. More precisely, there is a bijection between $\mathrm{Hom}_{\mathbf{Ring}}(L,R)$ and the set of formal group laws over R.

1.4 Commutativity theorems

In this section, we provide a commutativity criterion for formal group laws. More precisely,

Theorem 1.4.1 ([12, Chap. 1 Theorem 6.1])**.** *Let R be a ring. Then every one-dimensional formal group law over R is commutative if and only if R has no nonzero element a that is both nilpotent and $\mathbb{Z}$-torsion.*

To prove this theorem, we need some auxiliary lemmas.

Lemma 1.4.2. *Let F be a formal group law over a commutative ring R, then we have the following results:*

1. *F is commutative if and only if $F(X, F(Y, i(X))) = Y$.*
2. *F is commutative if and only if $F(X, F(Y, F(i(X), i(Y)))) = 0$.*

Proof. By definition,

$$F\text{is commutative} \iff F(X,Y) = F(Y,X).$$

Proof of Part (1):

$$\begin{aligned} F(X,Y) = F(Y,X) &\iff F(F(X,Y), i(X)) = F(F(Y,X), i(X)) \\ &\iff F(X, F(Y, i(X))) = Y. \end{aligned}$$

Proof of Part (2):

$$\begin{aligned} F(X,Y) = F(Y,X) &\iff F(X, F(Y, i(X))) = Y \\ &\iff F(F(X, F(Y, i(X))), i(Y)) = F(Y, i(Y)) \\ &\iff F(X, F(Y, F(i(X), i(Y)))) = 0. \end{aligned}$$

□

Lemma 1.4.3 ([12, Chap. 1 Lemma 6.2.1]). *Let $F(X,Y)$ be a non-commutative one-dimensional formal group law over a ring R. Then there exists a nonzero homomorphism from $F(X,Y)$ into $\mathbb{G}_a(X,Y)$ or into $\mathbb{G}_m(X,Y)$.*

Proof. For simplicity, we denote $F(X,Y)$ by $X +_F Y$ and its inverse $i(X)$ by $-_F X$. Let $H(X,Y) = F(X, F(Y, i(X)))$. By Lemma 1.4.2 and $F(X,Y)$ is non-commutative, we have

$$H(X,Y) = Y + \sum_{i=1}^{\infty} r_n(X) Y^n,$$

with $r_n(X) \neq 0$ for some n. Since $H(0,Y) = F(0, F(Y, i(0))) = Y$, we have $r_n(0) = 0$ for all n. By associativity of $F(X,Y)$, we have

$$\begin{aligned} H(X_1, H(X_2,Y)) &= X_1 +_F H(X_2,Y) +_F (-_F(X_1)) \\ &= X_1 +_F (X_2 +_F Y +_F (-_F X_2)) +_F (-_F(X_1)) \\ &= (X_1 +_F X_2) +_F Y +_F (-_F(X_1 +_F X_2)) \\ &= H(X_1 +_F X_2, Y) \end{aligned}$$

Let i be the least integer such that $r_i(X) \neq 0$.
For the case $i = 1$: we have

$$H(X,Y) \equiv (1 + r_1(X))Y \pmod{Y^2}$$

Combining this with equation $H(X_1, H(X_2,Y)) = H(X_1 +_F X_2, Y)$, we have

$$\begin{aligned} H(X_1, H(X_2,Y)) &\equiv (1 + r_1(X_1))H(X_2,Y) \\ &\equiv (1 + r_1(X_1))(1 + r_1(X_2))Y \pmod{Y^2}, \\ H(X_1 +_F X_2, Y) &\equiv (1 + r_1(X_1 +_F X_2))Y \pmod{Y^2}. \end{aligned}$$

Therefore, $(1 + r_1(X_1))(1 + r_1(X_2)) = 1 + r_1(X_1 +_F X_2)$, i.e.

$$r_1(X_1 +_F X_2) = r_1(X_1) + r_1(X_2) + r_1(X_1) r_1(X_2).$$

This means $r_1(X)$ is a formal group homomorphism from F to $\mathbb{G}_m$.
For the case $i > 1$: we have

$$H(X,Y) \equiv Y + r_i(X) Y^i \pmod{Y^{i+1}}.$$

Note that we have

$$\begin{aligned} H(X_1, H(X_2,Y)) &\equiv H(X_2,Y) + r_i(X_1) H(X_2,Y)^i \\ &\equiv Y + r_i(X_2) Y^i + r_i(X_1) Y^i \pmod{Y^{i+1}}, \\ H(X_1 +_F X_2, Y) &\equiv Y + r_i(X_1 +_F X_2) Y^i \pmod{Y^{i+1}}, \\ H(X_1, H(X_2,Y)) &= H(X_1 +_F X_2, Y). \end{aligned}$$

Therefore, $r_i(X_1 +_F X_2) = r_i(X_1) + r_i(X_2)$, which means $r_i(X)$ is a formal group homomorphism from F to $\mathbb{G}_a$. □

Lemma 1.4.4 ([12, Chap. 1 Lemma 6.2.11]). *Let R be an integral domain, and let F, F' be one-dimensional formal group laws over R. Suppose there exists a homomorphism $\alpha(X) : F(X,Y) \to F'(X,Y)$ and $F'(X,Y)$ is commutative. Then $F(X,Y)$ is commutative.*

Proof. Let r be the order of $\alpha(X) = \sum_{n=1}^{\infty} a_n X^n$, i.e. the least n such that a_n is nonzero. Let $C(X,Y) = X +_F Y +_F (-_F X) +_F (-_F Y)$ be the commutator of X and Y with respect to F.

By Lemma 1.4.2, a formal group law F is commutative if and only if the commutator $C(X,Y)$ with respect to F is zero.

Since F' is commutative, we have

$$\alpha(C(X,Y)) = \alpha(X) +_G \alpha(Y) +_G (-_G \alpha(X)) +_G (-_G \alpha(Y)) = 0.$$

Suppose F is non-commutative, i.e. $C(X,Y) \neq 0$. Let $C_n(X,Y)$ be the homogeneous component in $C(X,Y)$ of degree n, and let m be the least integer such that $C_m(X,Y)$ is non-zero. We have

$$\alpha(C(X,Y)) \equiv a_r C_m(X,Y)^r \pmod{\deg(mr+1)}.$$

Since $a_r \in R$ and R is an integral domain, this term is non-zero, which contradicts the assumption that $\alpha(C(X,Y)) = 0$. □

Proof of Theorem 1.4.1. First, we prove the necessity of the condition on R.

Let $a \in R$ be an nonzero element of R that is nilpotent and $\mathbb{Z}$-torsion. We need to show that there exists a formal group law $F(X,Y)$ over R such that $F(X,Y)$ is non-commutative.

Let n be the least positive integer such that $na = 0$. Then $n > 1$ and there exists a prime number p such that $p \mid n$. Let $b = \frac{n}{p}a$. Then $b \neq 0$ and $pb = 0$, since n is minimal. Let m be the least integer such that $b^m = 0$ and let $c = b^{m-1}$. Therefore, this c satisfies $c \neq 0, c^2 = 0, pc = 0$. Denote $F(X,Y) = X + Y + cXY^p$. Then

$$F(F(X,Y),Z) = X + Y + Z + c(XY^p + XZ^p + YZ^p) = F(X,F(Y,Z)).$$

Therefore, $F(X,Y)$ is a non-commutative formal group law over R.

Now we prove sufficiency of the condition on R. Suppose first that R is a integral domain. Let $F(X,Y)$ be a formal group law over R, where R is an integral domain. Suppose that $F(X,Y)$ is non-commutative. Then, by Lemma 1.4.3, there exists a nonzero formal group homomorphism from $F(X,Y)$ to $\mathbb{G}_a(X,Y)$ or $\mathbb{G}_m(X,Y)$. Since $\mathbb{G}_a(X,Y)$ and $\mathbb{G}_m(X,Y)$ are commutative formal group laws, Lemma 1.4.4 implies that $F(X,Y)$ is commutative. This leads to a contradiction.

Suppose that R is $\mathbb{Z}$-torsion free. A proof that F is commutative over R is given later in Corollary 1.6.4. Also, there is a different proof in [12, Chap. 1 Lemma 6.1.3].

Finally, let R be a ring such that R has no nonzero element a that is both nilpotent and $\mathbb{Z}$-torsion.

Given a prime ideal $\mathfrak{p}$ in R, consider the ring homomorphism $\phi : R \to R/\mathfrak{p}$. Since $R/\mathfrak{p}$ is an integral domain, $\phi_* F(X,Y)$ is commutative. If we denote $F(X,Y) = \sum_{i,j} a_{ij} X^i Y^j$, then it follows that $\phi_*(a_{ij} - a_{ji}) = 0 \in R/\mathfrak{p}$, which is equivalent to $a_{ij} - a_{ji} \in \mathfrak{p}$. Since this holds for all prime ideals $\mathfrak{p}$, we have

$$a_{ij} - a_{ji} \in \bigcap_{\mathfrak{p}\ prime} \mathfrak{p}.$$

Therefore, for any i, j, $a_{ij} - a_{ji}$ is nilpotent in R.

Consider the ring homomorphism $\phi_\infty : R \to R \otimes_{\mathbb{Z}} \mathbb{Q}$. Since $R \otimes_{\mathbb{Z}} \mathbb{Q}$ is $\mathbb{Z}$-torsion free, $\phi_{\infty *} F(X,Y)$ is commutative. Therefore, $\phi_\infty(a_{ij} - a_{ji}) = 0 \in R \otimes_{\mathbb{Z}} \mathbb{Q}$, which means $a_{ij} - a_{ji}$ is $\mathbb{Z}$-torsion. Since there is no nonzero element in R that is nilpotent and $\mathbb{Z}$-torsion at the same time, it follows that for any i, j, $a_{ij} - a_{ji} = 0$, which implies that $F(X,Y)$ is commutative.

□

1.5 Invariant differentials

Let F be a formal group law over R. In this setting, a differential form is an expression of the form $P(X)\,dX$ for some $P(X) \in R[\![X]\!]$. In this section, we discuss those differential forms that respect the group structure of F.

Definition 1.5.1 ([17, Chap. 4 Definition 4.1])**.** An *invariant differential* on a formal group law F over a ring R is a differential form $\omega(X) = P(X)\,dX$ with $P(X) \in R[\![X]\!]$ satisfying

$$\omega(X) \circ F(X, Y) = \omega(X).$$

To be precise, $P(F(X,Y))F_X(X,Y) = P(X)$, where $F_X(X,Y)$ is the partial derivative of F with respect to its first variable. An invariant differential is called *normalized* if $P(0) = 1$.

Recall that a power series f over R is invertible in $R[\![X]\!]$ if and only if its constant coefficient is invertible in R.

Proposition 1.5.2 ([17, Chap. 4 Proposition 4.2])**.** *Let F be a formal group law over R. Then there exists a unique normalized invariant differential on F. It is given by the formula*

$$\omega(X) = F_X(0, X)^{-1}\,dX.$$

Moreover, every invariant differential on F can be written as $a\omega$ for some $a \in R$.

Proof. Write $\omega(X) = P(X)\,dX$. Since ω is invariant, we must have

$$P(F(X,Y))F_X(X,Y) = P(X).$$

Let $X = 0$ in the above equation and use the fact that $F(0, Y) = Y$. Then we have

$$P(Y)F_X(0,Y) = P(0).$$

Since $F_X(0,Y)$ is of the form $(1 + O(Y))$, $F_X(0,Y)$ is invertible in $R[\![Y]\!]$. Here $O(Y)$ is a power series of order at least 1. Then

$$P(Y) = P(0)F_X(0,Y)^{-1}. \tag{1.5.2.1}$$

i.e. $P(X)$ is completely determined by the value $P(0)$. Therefore, it is enough to verify that

$$\omega = F_X(0, X)^{-1}\,dX.$$

is a normalized invariant differential.

It is normalized since $F_X(0, X)$ is of the form $(1 + O(X))$. Hence, we just need to show that ω is invariant, i.e.

$$F_X(0, F(Y, Z))^{-1}F_X(Y, Z) = F_X(0, Y)^{-1}.$$

Consider the associativity condition $F(F(X, Y), Z) = F(X, F(Y, Z))$ and differentiate it with respect to X. Then we have

$$F_X(X, F(Y, Z)) = F_X(F(X, Y), Z)F_X(X, Y).$$

Taking $X = 0$ above, we have

$$F_X(0, F(Y, Z)) = F_X(Y, Z)F_X(0, Y).$$

This is exactly what we want.

Finally, by equation (1.5.2.1), it follows that every invariant differential is of form $a\omega$. □

Example 1.5.3. The normalized invariant differential on $\mathbb{G}_a$ is dX, and the normalized invariant differential on $\mathbb{G}_m$ is $(1+X)^{-1}\,dX = \sum_{n=0}^{\infty}(-1)^n X^n\,dX$.

Corollary 1.5.4 ([17, Chap. 4 Corollary 4.3])**.** *Let F, G be formal group laws over R with normalized invariant differentials ω_F and ω_G, respectively. Let f be a homomorphism from F to G. Then*

$$\omega_G \circ f = f'(0)\omega_F.$$

where $f'(X)$ is the formal derivative of $f(X)$.

Proof. According to the definition, we have

$$\omega_G \circ f(F(X,Y)) = \omega_G \circ G(f(X), f(Y)) = \omega_G \circ f(X).$$

Then $\omega_G \circ f$ is an invariant differential on F. It follows that $\omega_G \circ f$ can be expressed as $a\omega_F$ for some $a \in R$. By checking the coefficient of X, we can deduce that $a = f'(0)$. □

Corollary 1.5.5 ([17, Chap. 4 Corollary 4.4])**.** *Let F be a formal group law over R and let p be a prime number. Then there are power series $f(X), g(X)$ over R with zero constant coefficient such that*

$$[p]_F(X) = pf(X) + g(X^p).$$

Proof. Let ω be the normalized invariant differential on F. We have discussed before that $[p](X) = pX + O(X^2)$. Then by Corollary 1.5.4, together with $[p]'(0) = p$, we have

$$\omega \circ [p] = p\omega.$$

If we write $\omega(X) = P(X)\,dX$, then

$$\omega \circ [p](X) = P(X)[p]'(X)\,dX.$$

It follows that $P(X)[p]'(X)\,dX = p\omega(X) = pP(X)\,dX$. Since ω is a normalized invariant differential on F, then $P(X)$ is of the form $(1 + O(X))$. It follows that $P(X)$ is invertible in $R[\![X]\!]$. Therefore, we have

$$[p]'(X) \in pR[\![X]\!].$$

If we denote $[p](X) = \sum_{n=1}^{\infty} a_n X^n$. Then for each $n \in \mathbb{N}_{>0}$ such that $p \nmid n$, then $a_n \in pR$. □

Example 1.5.6. Assume that R is a ring of characteristic p. Then the p-series $[p]$ has the form $[p](X) = g(X^p)$ for some $g(X) \in R[\![X]\!]$.

1.6 The formal logarithm

In this section, we discuss the formal logarithm of a formal group law. By integrating an invariant differential form, we could hope to obtain a homomorphism to the additive group. However, this does not work out in general, because integration introduces denominators. In the case that R of characteristic 0, or torsion-free rings, let F be a formal group over R, we obtain a formal group homomorphism from F to additive formal group law $\mathbb{G}_a$ over a ring $K \coloneqq R \otimes \mathbb{Q}$.

From now on, we fix a torsion-free ring R and let $K = R \otimes \mathbb{Q}$.

Definition 1.6.1 ([17, Chap. 4 Definition 5.1])**.** Let F be a formal group law over R, and let $\omega(X)$ be its normalized invariant differential. Write $\omega(X) = (1 + a_1X + a_2X^2 + \cdots)dX$. Then *the formal logarithm* of F is defined by

$$\log_F(X) := \int \omega = X + \frac{a_1}{2}X^2 + \frac{a_2}{3}X^3 + \cdots \in K[\![X]\!]$$

Since the coefficient of X in $\log_F(X)$ is 1, which is invertible in K. Then $\log_F(X)$ is invertible w.r.t. composition in $K[\![X]\!]$ and we can define $\exp_F(X)$ to be the unique power series over K satisfying

$$\exp_F(\log_F(X)) = X.$$

Example 1.6.2. The normalized invariant differential on $\mathbb{G}_a$ is dX, so we have

$$\log_{\mathbb{G}_a}(X) = X,\ \exp_{\mathbb{G}_a}(X) = X.$$

The normalized invariant differential on $\mathbb{G}_m$ is $(1 + X)^{-1}dX$, so we have

$$\log_{\mathbb{G}_m}(X) = \sum_{n=1}^{\infty} \frac{(-1)^{n-1}X^n}{n} \quad \text{and}$$

$$\exp_{\mathbb{G}_m}(X) = \sum_{n=1}^{\infty} \frac{X^n}{n!}.$$

Proposition 1.6.3 ([17, Chap. 4 Proposition 5.2])**.** *Let F be a formal group law over R. Then $\log_F(X)$ is a formal group isomorphism from F to $\mathbb{G}_a$ over K.*

Proof. Let $\omega(X)$ be the normalized invariant differential for F. Then we have

$$\omega \circ F = \omega.$$

Integrating this with respect to X, we have

$$\log_F(F(X, Y)) = \log_F(X) + C(Y).$$

where $C(Y) \in K[\![Y]\!]$. Substitute $X = 0$ in the above equation. Then we have

$$C(Y) = \log_F(0) + C(Y) = \log_F(F(0, Y)) = \log_F(Y).$$

Hence,

$$\log_F(F(X, Y)) = \log_F(X) + \log_F(Y).$$

This proves that $\log_F(X)$ is a homomorphism from F to $\mathbb{G}_a$. Since the coefficient of X in $\log_F$ is invertible in K, by Lemma 1.1.8, $\log_F$ is a formal group isomorphism from F to $\mathbb{G}_a$. □

Corollary 1.6.4. *Let R be a torsion-free ring. Then every formal group law over R is commutative.*

Proof. The assumption that R is torsion-free guarantees the existence of $\log_F$ and $\exp_F$. Then

$$F(X, Y) = \exp_F(\log_F(F(X, Y))) = \exp_F(\log_F(X) + \log_F(Y)).$$

Therefore, we have $F(X, Y) = F(Y, X)$. □

Further discussion of the coefficients of $\log_F$ and $\exp_F$ can be found in [17].

1.7 Heights

In this section, we fix a prime number p and a field k of characteristic p. In this section, unless otherwise specified, all formal group laws are assumed to be commutative.

In this section, we discuss the definition and properties of the height of a formal group law.

Definition 1.7.1 (Height of a homomorphism, [17, Chap. 4 Definition 7.1])**.** Let F, G be formal group laws over k, and let $f \in k[\![X]\!]$ be a homomorphism from $F(X,Y)$ to $G(X,Y)$. Then the *height of the homomorphism* f, denoted by $\mathrm{ht}(f)$, is the largest number h such that

$$f(X) = g\left(X^{p^h}\right),$$

for some power series $g(X)$ over k. If $f = 0$, then we set $\mathrm{ht}(f) = \infty$.

Remark 1.7.2. Let m be a natural number prime to p, then $\mathrm{ht}([m]) = 0$, since

$$[m](X) = mX + \cdots$$

Proposition 1.7.3 ([17, Chap. 4 Proposition 7.2])**.** *Let F, G be formal group laws over k, and let f be a nonzero homomorphism from F to G defined over k. Then*

1. *If $f'(0) = 0$, then $f(X) = f_1(X^p)$ for some $f_1(X) \in k[\![X]\!]$. That is,* $\mathrm{ht}(f) \geq 1$.
2. *There is a natural number m such that $f(X) = h\left(X^{p^m}\right)$ for some h such that $h'(0) \neq 0$.*
3. *Write $f(X) = g\left(X^{p^{\mathrm{ht}(f)}}\right)$. Then $g'(0) \neq 0$.*

Proof. Proof of Part (1): let ω_F, ω_G be the normalized invariant differentials of F and G, respectively. According to Corollary 1.5.4, we have

$$\begin{aligned} 0 &= f'(0)\omega_F \\ &= \omega_G \circ f \end{aligned}$$

If we write $\omega_G(X) = P_G(X)dX$, then we have

$$\omega_G \circ f(X) = \omega_G(f(X)) = P_G(f(X))f'(X)dX.$$

Then $f'(X) = 0$, since $P_G(X)$ is of the form $(1 + O(X))$. It follows that there is $f_1(X)$ such that $f(X) = f_1(X^p)$.

Proof of Part (2): if $f'(0) \neq 0$, then take m to be 0. If $f'(0) = 0$, then by Part (1) of this proposition, we have that $f(X) = f_1(X^p)$ for some $f_1(X) \in k[\![X]\!]$. If $f_1'(0) = 0$, then we just take $m = 1$. If $f_1'(0) = 0$, we denote $F_p = \varphi_* F$, where φ is an endomorphism of k, mapping $a \mapsto a^p$. Then F_p is also a formal group law. Now we claim that $f_1(X)$ is a homomorphism from F_p to G. We prove this claim by the following calculation:

$$\begin{aligned} f_1(F_p(X,Y)) &= f_1(F(X,Y)^p) \\ &= f_1(X^p) \circ F \\ &= f(X) \circ F \\ &= f_1 \circ F. \end{aligned}$$

Then again by Part (1) of this proposition, we have that there is an $f_2(X) \in k[\![X]\!]$, such that $f_1(X) = f_2(X^p)$. Recall that the order of a nonzero power series is defined to be the least natural number n sucht that the n-th coefficient of it is nonzero. Notice that the order f_i decline after each step. Then repeating this process at most n times where n is the order of $f(X)$, we will get a natural number m satisfying the condition.

Proof of Part (3): we first denote $q = p^{\mathrm{ht}(f)}$ and $F_q = \phi_* F$, where ϕ is an endomorphism of k, mapping $a \mapsto a^q$. Then F_q is also a formal group law. Now we claim that $g(X)$ is a homomorphism from F_q to G. We prove this claim by the following calculation:

$$\begin{aligned} g(F_q(X,Y)) &= g(F(X,Y)^q) \\ &= g(X^q) \circ F \\ &= f(X) \circ F \\ &= g \circ F. \end{aligned}$$

The first equation is guaranteed by the characteristic of k. If $g'(0) \neq 0$, then by the first assertion of the proposition, there is a $g_1(X)$ such that $g(X) = g'(X^p)$. It follows that $f(X) = g_1(X^{p^{\mathrm{ht}(f)+1}})$, which contradicts the definition of the height of f. □

Remark 1.7.4. Write $f(X) = \sum_{n=n_0}^{\infty} a_n X^n$, with $a_{n_0} \neq 0$. By Part (2) and (3) of the Proposition 1.7.3, n_0 is of the form p^t for some natural number t. Then we have a alternative description of the height of a formal group homomorphism $f(X)$ from F to G, that is the height of $f(X)$ can be defined to be the this t.

Definition 1.7.5 (Height of a formal group)**.** The *height of a formal group law* F is defined to be the height of the p-series $[p]_F(X)$, denoted by $\mathrm{ht}(F)$.

Example 1.7.6. The p-series for the multiplicative formal group $\mathbb{G}_m(X,Y) = (1+X)(1+Y)-1$ is $[p](X) = (1+X)^p - 1 = X^p$, so $\mathbb{G}_m(X,Y)$ has height exactly 1. For the additive formal group law $\mathbb{G}_a(X,Y)$, the p-series is $[p](X) = pX = 0$, so $\mathbb{G}_a(X,Y)$ has infinite height.

Proposition 1.7.7. *Let $F(X,Y)$ and $G(X,Y)$ be isomorphic formal group laws over k. Then $F(X,Y)$ and $G(X,Y)$ have the same height.*

Proof. Let $g(X)$ be a formal group isomorphism from $F(X,Y)$ to $G(X,Y)$, and let $[p]_F(X)$ and $[p]_G(X)$ be the p-series for $F(X,Y)$ and $G(X,Y)$, respectively. Note that

$$[p]_G(X) = (g \circ [p]_F \circ g^{-1})(X).$$

Therefore, $[p]_F(X)$ and $[p]_G(X)$ have the same order, and the coefficients of their lowest-degree non-zero terms are equal. By Remark 1.7.4, it follows that the heights of $F(X,Y)$ and $G(X,Y)$ are the same. □

Proposition 1.7.8. *Let F, G be formal group laws over k. Assume that $\mathrm{ht}(F) \neq \mathrm{ht}(G)$, then the only homomorphism from F to G, is the zero homomorphism.*

Proof. We prove the proposition by contradiction. Assume that there is a nonzero homomorphism f from F to G. Note that

$$f([p]_F(X)) = [p]_G(f(X)).$$

This is impossible, since the orders of $[p]_F(X)$ and $[p]_G(X)$ are different and $f(X) \neq 0$. □

Proposition 1.7.9 ([17, Chap. 4 Proposition 7.3]). *Let F, G, H be formal group laws over k, and let*

$$F \xrightarrow{f} G \xrightarrow{g} H$$

be a chain of homomorphisms defined over k. Then

$$\mathrm{ht}(g \circ f) = \mathrm{ht}(f) + \mathrm{ht}(g).$$

Proof. Denote $f(X) = f_1\left(X^{p^{\mathrm{ht}(f)}}\right)$ and $g(X) = g_1\left(X^{p^{\mathrm{ht}(g)}}\right)$. Then

$$(g \circ f)(X) = g_1\left(f_1(X^{p^{\mathrm{ht}(f)}})^{\mathrm{ht}(g)}\right) = g_1\left(\tilde{f}_1(X^{\mathrm{ht}(f)+\mathrm{ht}(g)})\right),$$

where $\tilde{f}_1(X)$ is obtained from $f_1(X)$ raising each coefficient of $f_1(X)$ to the $p^{\mathrm{ht}(g)}$-th power. □

Lemma 1.7.10. *Let R be a commutative ring, I, J be two index sets and $F \in R[\![X_I]\!]$. Let $\{f_i\}_{i\in I}$ be a set of power series in $R[\![X_J]\!]$ with the property that the constant coefficient of each f_i is nilpotent in R. Let $G(X_J)$ be the power series obtained by substituting f_i into F. We denote the order of power series f to be $o(f)$. Then we have that*

$$o(G(X_J)) \geq o(F) \cdot \min_{i\in I}\{o(f_i)\}.$$

Proof. Recall that for any power series $f(X)$, $n \leq o(f)$ is equivalent to for all $j < n$, the j-th homogeneous component $f(X)$ is zero.

Then it is enough to show that, for any j less than $o(F)\cdot\min_{i\in I}\{o(f_i)\}$, the j-th homogeneous component of G is zero. However, the least possible nonzero homogeneous component in $G(X)$ is at least of degree the product of the order of F and the least of the order of the f_i. □

Proposition 1.7.11 ([12], Definition 18.3.2). *Let F be a formal group law over k, and let f and g be two formal group endomorphisms of F. Then $F(f, g)$ is an endomorphism of F and* $\mathrm{ht}(F(f, g)) \geq \min\{\mathrm{ht}(f), \mathrm{ht}(g)\}$.

Proof. First, we show that $F(f, g)$ is an endomorphism of F. This follows from the following calculation:

$$\begin{aligned} F(f(X), g(X)) \circ F(X, Y) &= F(f(F(X, Y)), g(F(X, Y))) \\ &= F(F(f(X), f(Y)), F(g(X), g(Y))) \\ &= F(F(f(X), g(X)), F(f(Y), g(Y))) \\ &= F(X, Y) \circ F(f(X), g(X)). \end{aligned}$$

By Lemma 1.7.10 and the fact that order of $F(X, Y)$ is 1, then we have that

$$o(F(f(X), g(X))) \geq o(F) \cdot \min\{o(f), o(g)\} = \min\{o(f), o(g)\}.$$

Therefore, by Remark 1.7.4, it follows that $\mathrm{ht}(F(f, g)) \geq \min\{\mathrm{ht}(f), \mathrm{ht}(g)\}$. □

Remark 1.7.12. Let $i(X)$ be the additive inverse of the formal group law F, i.e. the power series such that $F(X, i(X)) = 0$. Note that

$$i(X) \circ F(X, Y) = i(F(X, Y)) = F(i(Y), i(X)) = F(i(X), i(Y)).$$

Then $i(X)$ is an endomorphism of F. By Proposition 1.7.11 and Proposition 1.7.9, we have that the set of all endomorphisms over k of F form a ring, which we denote as $\mathrm{End}_k(F)$. The multiplication is given by composition of power series and the addition of $f, g \in \mathrm{End}_k(F)$ is defined to be $F(f, g)$. The height function ht defines a valuation on the ring $\mathrm{End}_k(F)$.

If k is an algebraically closed field of characteristic p, then the following result shows that formal group laws over k are classified by their heights.

Theorem 1.7.13 (Lazard [13, Theorem IV])**.** *Let k be an algebraically closed field of characteristic p. Then two formal group laws $F(X,Y), G(X,Y)$ over k are isomorphic if and only if they have the same height.*

Since the proof of this theorem depends on many auxiliary results, we do not include the details here.

2 Lubin–Tate theory

2.1 Lubin–Tate formal group laws

Let K be a field complete with respect to a discrete valuation, with finite residue field. For instance, K is a nonarchimedean local field. Let $\mathcal{O}_K$ be the unit ball of K, let q be the cardinality of the residue field, and let π be a *uniformizer* of $\mathcal{O}_K$. A uniformizer of K is an element $\pi \in \mathcal{O}_K$ such that $\mathfrak{m}_K = (\pi)$. Equivalently, if v_K is the normalized discrete valuation on K, then π is a uniformizer if $v_K(\pi) = 1$. Thus every nonzero element $x \in K^\times$ can be written uniquely as $x = u\pi^n$, where $u \in \mathcal{O}_K^\times$ is a unit and $n \in \mathbb{Z}$.

Definition 2.1.1 ([5, Chap. 6 Definition 3.3])**.** Let $\mathcal{F}_\pi$ denote the set of power series $f(X) \in \mathcal{O}_K[\![X]\!]$ which satisfy the two conditions:

$$f(X) \equiv \pi X \pmod{\deg 2}, \tag{2.1.1.1}$$
$$f(X) \equiv X^q \pmod{\pi}. \tag{2.1.1.2}$$

Lemma 2.1.2 ([5, Chap. 6 Proposition 5])**.** *Let $f(X)$ and $g(X)$ be elements of $\mathcal{F}_\pi$ and let $L(X_1, \ldots, X_n) = \sum_{i=1}^n a_i X_i$ be a linear form with coefficients in $\mathcal{O}_K$.*

Then there is a unique power series $F(X_1, \ldots, X_n)$ over $\mathcal{O}_K$ such that

$$F(X_1, \ldots, X_n) \equiv L(X_1, \ldots, X_n) \pmod{\deg 2}, \tag{2.1.2.1}$$
$$f(F(X_1, \ldots, X_n)) = F(g(X_1), \ldots, g(X_n)). \tag{2.1.2.2}$$

Proof. First, we claim that for all $k \in \mathbb{N}$, there is a unique polynomial $F_k(X_1, \ldots, X_n)$ of degree k such that

$$F_k(X_1, \ldots, X_n) \equiv L(X_1, \ldots, X_n) \pmod{\deg 2} \quad \text{and}$$
$$f(F_k(X_1, \ldots, X_n)) \equiv F_k(g(X_1), \ldots, g(X_n)) \pmod{\deg(k+1)}.$$

Then we obtain a sequence of polynomials $F_k(X_1, \ldots, X_n)$. Define F to be $\lim F_k$. Then F satisfies the equation (2.1.2.1) and (2.1.2.2). Indeed, for equation (2.1.2.1), it follows that $F_k(X_1, \ldots, X_n) \equiv L(X_1, \ldots, X_n) \pmod{\deg 2}$ for all k. For equation (2.1.2.2), we have that for all k,

$$f(F(X_1, \ldots, X_n)) \equiv f(F_k(X_1, \ldots, X_n)) \equiv F_k(g(X_1), \ldots, g(X_n)) \pmod{\deg(k+1)}.$$

The first congruence follows from the fact that f has constant coefficient zero, together with

$$F(X_1, \ldots, X_n) \equiv F_k(X_1, \ldots, X_n) \pmod{\deg(k+1)}. \tag{2.1.2.3}$$

Moreover, we have that

$$F(g(X_1),\dots,g(X_n)) \equiv F_k(g(X_1),\dots,g(X_n)) \pmod{\deg(k+1)}.$$

Then we conclude that for all k,

$$f(F(X_1,\dots,X_n)) \equiv F(g(X_1),\dots,g(X_n)) \pmod{\deg(k+1)}.$$

It follows that $f(F(X_1,\dots,X_n)) = F(g(X_1),\dots,g(X_n))$. Finally, the uniqueness in this proposition follows from equation (2.1.2.3) and the uniqueness in the claim.

Proof of the claim: For $k=1$, we take $F_1(X_1,\dots,X_n) = L(X_1,\dots,X_n)$.

Now assume that the result holds for k. Write

$$f(F_k(X_1,\dots,X_n)) \equiv F_k(g(X_1),\dots,g(X_n)) + E_{k+1} \pmod{\deg(k+2)}.$$

According to the induction hypothesis, E_{k+1} is homogeneous of degree $k+1$.

On the other hand, by the uniqueness of $F_k(X_1,\dots,X_n)$, $F_{k+1}(X_1,\dots,X_n)$ should be of the form $F_{k+1}(X_1,\dots,X_n) = F_k(X_1,\dots,X_n) + \phi_{k+1}(X_1,\dots,X_n)$, where $\phi_{k+1}(X_1,\dots,X_n)$ is homogeneous of degree $(k+1)$, due to the definition of $\mathcal{F}_\pi$. Then we have

$$\begin{aligned} f(F_{k+1}(X_1,\dots,X_n)) &\equiv F_k(g(X_1),\dots,g(X_n)) + \pi\phi_{k+1}(X_1,\dots,X_n) \pmod{\deg(k+2)},\\ F_{k+1}(g(X_1),\dots,g(X_n)) &\equiv F_k(g(X_1),\dots,g(X_n)) + \pi^{k+1}\phi_{k+1}(X_1,\dots,X_n) \pmod{\deg(k+2)}. \end{aligned}$$

The last congruence follows from the fact that ϕ_{k+1} is homogeneous of degree $(k+1)$, that is,

$$\phi_{k+1}(g(X_1),\dots,g(X_n)) \equiv \phi_{k+1}(\pi X_1,\dots,\pi X_n) \equiv \pi^{k+1}\phi_{k+1}(X_1,\dots,X_n) \pmod{\deg(k+1)}.$$

Therefore, we conclude that

$$E_{k+1} + (\pi - \pi^{k+1})\phi_{k+1}(X_1,\dots,X_n) = 0.$$

Note that $1-\pi^k$ is invertible in $\mathcal{O}_K$. And

$$E_{k+1} \equiv f(F_k(X_1,\dots,X_n)) - F_k(g(X_1),\dots,g(X_n)) \equiv F_k^q - F_k(X_1^q,\dots,X_n^q) \equiv 0 \pmod{\pi}.$$

Then $\phi_{k+1}(X_1,\dots,X_n) = -E_{k+1}/(\pi-\pi^{k+1})$. □

Definition 2.1.3. For any $f \in \mathcal{F}_\pi$, we let $F_f(X,Y)$ be the unique solution of

$$\begin{aligned} F_f(X,Y) &\equiv X+Y \pmod{\deg 2},\\ f(F_f(X,Y)) &= F_f(f(X),f(Y)). \end{aligned}$$

Theorem 2.1.4 (Lubin–Tate Formal Group Laws [5, Chap. VI.3.3 Proposition 1])**.** *For power series $f \in \mathcal{F}_\pi$, we have*

$$F_f(X,Y) = F_f(Y,X) \tag{2.1.4.1}$$

$$F_f(F_f(X,Y),Z) = F_f(X,F_f(Y,Z)) \tag{2.1.4.2}$$

As a result, $F_f(X,Y)$ is a commutative formal group law over $\mathcal{O}_K$.

Proof. For equation (2.1.4.1), let $G(X,Y) = F(Y,X)$. Then $G(X,Y) \equiv X+Y \pmod{\deg 2}$ and

$$f(G(X,Y)) = f(F(Y,X)) = F(f(Y),f(X)) = G(f(X),f(Y)).$$

Therefore, by the uniqueness in Lemma 2.1.2, $F(X,Y) = G(X,Y) = F(Y,X)$.

For equation (2.1.4.2), let

$$G_1(X,Y,Z) = F_f(F_f(X,Y),Z) \quad \text{and} \quad G_2(X,Y,Z) = F_f(X,F_f(Y,Z)).$$

We can check that $G_i(X,Y,Z) \equiv X+Y+Z \pmod{\deg 2}$ and

$$f(G_i(X,Y,Z)) = G_i(f(X),f(Y),f(Z)).$$

Indeed,

$$f(G_1(X,Y,Z)) = f(F_f(F_f(X,Y),Z)) = F_f(f(F_f(X,Y)),f(Z)) = F_f(F_f(f(X),f(Y)),f(Z)).$$

$$f(G_2(X,Y,Z)) = f(F_f(X,F_f(Y,Z))) = F_f(f(X),f(F_f(Y,Z))) = F_f(f(X),F_f(f(Y),f(Z))).$$

Then, by uniqueness in Lemma 2.1.2, we conclude the desired equation (2.1.4.2). □

Definition 2.1.5. For each $a \in \mathcal{O}_K$ and $f,g \in \mathcal{F}_\pi$, let $[a]_{f,g}(X)$ be the unique solution of

$$[a]_{f,g}(X) \equiv aX \pmod{\deg 2}, \tag{2.1.5.1}$$

$$f([a]_{f,g}(X)) = [a]_{f,g}(g(X)). \tag{2.1.5.2}$$

In particular, we write $[a]_f$ instead of $[a]_{f,f}$ to simplify notation.

Remark 2.1.6. Note that for any $f \in \mathcal{F}_\pi$,

$$f(X) \equiv \pi X \equiv [\pi]_f \pmod{\deg 2} \quad \text{and} \quad f \circ [\pi]_f = [\pi]_f \circ f.$$

Then by uniqueness in Lemma 2.1.2, it follows that $[\pi]_f = f$.

Theorem 2.1.7 ([5, Chap. VI 3.3 Proposition 2])**.** *For each $a \in \mathcal{O}_K$ and $f,g \in \mathcal{F}_\pi$, the power series $[a]_{f,g}(X)$ is a formal group homomorphism from $F_g(X,Y)$ to $F_f(X,Y)$, i.e.*

$$[a]_{f,g}F_g(X,Y) = F_f([a]_{f,g}(X),[a]_{f,g}(Y)).$$

Proof. For simplicity, we denote $[a]_{f,g}$ by h. We have $h(F_g(X,Y)) \equiv aX+aY \pmod{\deg 2}$, $F_f(h(X),h(Y)) \equiv aX+aY \pmod{\deg 2}$ and

$$h \circ F_g(g(X),g(Y)) = h \circ g(F_g(X,Y)) = f \circ h(F_g(X,Y))$$
$$F_f(h(g(X)),h(g(Y))) = F_f(f(h(X)),f(h(Y))) = f \circ F_f(h(X),h(Y))$$

Then the result follows from uniqueness in Lemma 2.1.2. □

Theorem 2.1.8 ([15, Theorem 1])**.** *Let $a,b \in \mathcal{O}_K$, $f,g,h \in \mathcal{F}_\pi$, then*

$$[a]_{f,g}([b]_{g,h}(X)) = [ab]_{f,h}(X).$$

Proof. We have $[a]_{f,g}([b]_{g,h}(X)) \equiv abX \pmod{\deg 2}$ and

$$f \circ ([a]_{f,g} \circ [b]_{g,h}) = [a]_{f,g} \circ (g \circ [b]_{g,h}) = [a]_{f,g} \circ [b]_{g,h} \circ h.$$

Thus, $[ab]_{f,h}$ and $[a]_{f,g} \circ [b]_{g,h}$ are solutions satisfying these conditions. By uniqueness in Lemma 2.1.2, they coincide. □

Theorem 2.1.9 ([15, Theorem 1]). *Let $a, b \in \mathcal{O}_K$, $f, g \in \mathcal{F}_\pi$, then*

$$F_f([a]_{f,g}(X), [b]_{f,g}(X)) = [a+b]_{f,g}(X).$$

Proof. Note that $F_f([a]_{f,g}(X), [b]_{f,g}(X)) \equiv (a+b)X \pmod{\deg 2}$ and

$$f(F_f([a]_{f,g}(X), [b]_{f,g}(X))) = F_f(f \circ [a]_{f,g}(X), f \circ [b]_{f,g}(X)) = F_f([a]_{f,g} \circ f(X), [b]_{f,g} \circ f(X)).$$

Therefore, by the uniqueness in Lemma 2.1.2, we have

$$F_f([a]_{f,g}(X), [b]_{f,g}(X)) = [a+b]_{f,g}(X).$$

□

Theorem 2.1.10 ([5, Chap. VI 3.3 Proposition 4]). *Let $f, g \in \mathcal{F}_\pi$, then F_f is isomorphic to F_g, and there exists a unique strong isomorphism from F_f to F_g. This means that the Lubin–Tate formal group F_f is independent of $f \in \mathcal{F}_\pi$, and depends only on π.*

Proof. Take a unit $a \in \mathcal{O}_K$, then

$$[a]_{f,g} \circ [a^{-1}]_{g,f} = [aa^{-1}]_f = [1]_f. \quad \text{and} \quad [a^{-1}]_{g,f} \circ [a]_{f,g} = [a^{-1}a]_g = [1]_g.$$

It follows that $[a]_{f,g}$ is a formal group isomorphism from F_g to F_f with its inverse $[a^{-1}]_{g,f}$. And the unique strong isomorphism is $[1]_{f,g}$. □

Remark 2.1.11. Let L be an algebraic extension of K, let $\mathfrak{m}_L$ be the maximal ideal in the ring of integers of L. Given elements $x_1, \ldots, x_n \in \mathfrak{m}_L$ and a formal series $G(X_1, \ldots, X_n) \in \mathcal{O}_K[\![X_1, \ldots, X_n]\!]$, the series $G(x_1, \ldots, x_n)$ converges to an element in $\mathfrak{m}_L$ if the constant term of $G(X_1, \ldots, X_n)$ is zero. By the theorem above, for each power series $f \in \mathcal{F}_\pi$, this determines an $\mathcal{O}_K$-module structure on the set $\mathfrak{m}_L$. More precisely, addition and scalar multiplication are defined by

$$x + y = F_f(x, y),$$
$$ax = [a]_f(x).$$

2.2 Applications

Fix an algebraic closure $\overline{K}$ of K, and let K^{sep} be a separable closure of K. Given $f \in \mathcal{F}_\pi$, for each integer $n \geq 1$, let $f^n(X)$ be the n-th iterate of $f(X)$. Let $\Lambda_{f,n}$ be the set of all roots of $f^n(X)$ in $\overline{K}$ and let $K(\Lambda_{f,n})$ be the field generated over K by these elements. Denote $\Lambda_f = \bigcup_{n=1}^{\infty} \Lambda_{f,n}$.

Lemma 2.2.1. *For any $f, g \in \mathcal{F}_\pi$, $\lambda \in \Lambda_{f,n}$ if and only if $[1]_{g,f}(\lambda) \in \Lambda_{g,n}$.*

Proof. By equation (2.1.5.2) and induction, we have that for any $n \in \mathbb{N}_{>0}$,

$$g^n \circ [1]_{g,f} = [1]_{g,f} \circ f^n.$$

Suppose that $\lambda \in \Lambda_{f,n}$, so that $f^n(\lambda) = 0$. Then $g^n([1]_{g,f}(\lambda)) = [1]_{g,f}(f^n(\lambda)) = [1]_{g,f}(0) = 0$. We conclude that $[1]_{g,f}(\lambda) \in \Lambda_{g,n}$.

Suppose $[1]_{g,f}(\lambda) \in \Lambda_{g,n}$. Denote $[1]_{g,f}(\lambda)$ by μ. Then by the discussion above, we have that $[1]_{f,g}(\mu) \in \Lambda_{f,n}$. Note that

$$[1]_{f,g}(\mu) = [1]_{f,g}([1]_{g,f}(\lambda)) = \lambda.$$

Then we conclude that $\lambda \in \Lambda_{f,n}$. □

Remark 2.2.2. It follows that the field $K(\Lambda_{f,n})$ is independent of f and depends only on the uniformizer π. Therefore, we denote $K(\Lambda_{f,n})$ by $K_{\pi,n}$. Note that

$$K(\Lambda_f) = K\left(\bigcup_{n=1}^{\infty} \Lambda_{f,n}\right) = \bigcup_{n=1}^{\infty} K(\Lambda_{f,n}).$$

The second equality follows from the fact that $\{K(\Lambda_{f,n})\}_n$ is an increasing chain of fields. It follows that $K(\Lambda_f)$ is independent of f, so we denote $K(\Lambda_f)$ by K_π. Moreover, we denote $\mathrm{Gal}(K_{\pi,n}/K)$ by $G_{\pi,n}$ and $\varprojlim G_{\pi,n}$ by G_π.

According to Remark 2.1.11, F_f defines an $\mathcal{O}_K$-module structure on $\mathfrak{m}_{K^{\mathrm{sep}}}$ and we denote this $\mathcal{O}_K$-module as $M_f(K^{\mathrm{sep}})$. Then $\Lambda_{f,n}$ consists of those elements $\lambda \in \mathfrak{m}_{K^{\mathrm{sep}}}$ such that $\pi^n \lambda = 0$.

Theorem 2.2.3 ([5, Chap. VI 3.3 Theorem 3])**.** *$\Lambda_{f,n}$ is isomorphic to $\mathcal{O}_K/\pi^n\mathcal{O}_K$ as an $\mathcal{O}_K$-module and Λ_f is isomorphic to $K/\mathcal{O}_K$ as an $\mathcal{O}_K$-module.*

In order to prove this theorem, we need a result about divisible modules over a PID.

Definition 2.2.4 ([1], Chap. 5 Definition 18.10)**.** Let R be a ring and S be a simple left R-module. An *injective hull* (or *injective envelope*) of S, denoted by $E(S)$, is an injective R-module E together with an *essential monomorphism* $\iota\colon S \to E$. Here, *essential* means that every nonzero submodule of $E(S)$ has nonzero intersection with S.

Equivalently, $E(S)$ can be characterized as:

1. The maximal essential extension of S.
2. The minimal injective extension of S.

Remark 2.2.5. The injective hull $E(S)$ of a simple module S is necessarily an *indecomposable* injective module; that is, $E(S)$ cannot be written as a direct sum of two nonzero submodules.

Theorem 2.2.6 (Structure theorem of divisible modules over a PID, [16], Chap 10, Prop 3.1)**.** *Let R be a principal ideal domain and K be its field of fractions. Then every divisible R-module is isomorphic to a direct sum of K and copies of the injective hulls of the simple module $R/(p)$, varying over prime elements $p \in R$.*

Let M be a divisible R-module, then M can be written as follows:

$$M \cong \left(\bigoplus_{i\in I} K\right) \oplus \left(\bigoplus_{p\ prime} \bigoplus_{j\in J_p} E(R/(p))\right), \quad \textit{where}$$

- *I is an index set measuring how many copies of the torsion-free part of M are needed.*
- *p varies over all prime elements in R.*
- *J_p is an index set measuring how many copies of p-primary torsion part $E(R/(p))$ are needed for each p.*
- *$E(R/(p))$ is the injective hull of $R/(p)$. Over a PID, this is isomorphic to $K/R_{(p)}$.*

Proof of theorem 2.2.3. The first claim follows from the fact that $\Lambda_{f,n}$ is the π^n-torsion of the $\mathcal{O}_K$-module $M_f(K^{\text{sep}})$.

According to Theorem 2.1.10, we are free to choose $f \in \mathcal{F}_\pi$ as we please. We take $f(X) = \pi X + X^q$. Since $f'(X) = qX^{q-1} + \pi \neq 0$, $\Lambda_{f,1}$ consists of q distinct roots of the equation $X^q + \pi X = 0$. For any given $\alpha \in \mathfrak{m}_{K^{\text{sep}}}$, since $\pi X + X^q = \alpha$ is separable, $\Lambda_{f,1} \subset \mathfrak{m}_{K^{\text{sep}}}$. This means that for any given $\alpha \in \mathfrak{m}_K^{\text{sep}}$, there is a β such that $\pi \cdot \beta = \alpha$ in $M_f(K^{\text{sep}})$. It follows that for any given $r \in \mathcal{O}_K$, $rM_f(K^{\text{sep}}) = M_f(K^{\text{sep}})$, which means that $M_f(K^{\text{sep}})$ is a divisible $\mathcal{O}_K$-module. Then Λ_f is a divisible torsion $\mathcal{O}_K$-module.

By Theorem 2.2.6 and the fact that $\mathcal{O}_K$ is a PID, we have that

$$\Lambda_f \cong \left(\bigoplus_{i \in I} K \right) \oplus \left(\bigoplus_{p \text{ prime}} \bigoplus_{j \in J_p} E(R/(p)) \right).$$

Since Λ_f is a torsion module, the index I should be empty, i.e. there are no copies of K in this decomposition. Note that $E(R/(p))$ is isomorphic to $K/R_{(p)}$, since $\mathcal{O}_K$ is a PID. Since π is the only prime element in $\mathcal{O}_K$ and $(\mathcal{O}_K)_{(\pi)} \cong \mathcal{O}_K$, it is enough to determine the index set J.

Moreover, $\Lambda_{f,1}$ is a one-dimensional vector space over the residue field k (since $\Lambda_{f,1}$ has the same cardinality as the residue field k). It follows that $|J| = \dim_k M_f(K^{\text{sep}})[\pi] = \dim_k \Lambda_{f,1} = 1$, where J is the index set of copies of $K/\mathcal{O}_K$. We conclude that $\Lambda_f \cong K/\mathcal{O}_K$. □

There is a natural homomorphism $\text{Gal}(K^{\text{sep}}/K) \to \text{End}(\Lambda_f)$; this follows from the action of the Galois group on the set of roots of a polynomial. Moreover, since $\Lambda_f \cong K/\mathcal{O}_K$ and $\text{End}(K/\mathcal{O}_K) \cong \mathcal{O}_K$, then this induces a homomorphism $\text{Gal}(K^{\text{sep}}/K) \to \text{Aut}(K/\mathcal{O}_K) \cong U_K$.

Theorem 2.2.7 ([5, Chap. VI 3.3 Theorem 3])**.** *The natural homomorphism* $\text{Gal}(K^{\text{sep}}/K) \to \text{Aut}(\Lambda_f)$ *is an isomorphism.*

Proof. This map is injective by definition; it remains to prove that it is surjective. We have an injection $\text{Gal}(K_{\pi,n}/K) \to U_K/U_K^{(n)}$, where $U_K^{(n)} = 1 + \pi^n \mathcal{O}_K$. Let $\alpha \in \Lambda_{f,n}$ be a primitive element, that is, an element such that $[\pi^{n-1}]_f \alpha \neq 0$ or $\pi^{n-1} \cdot \alpha \neq 0$ in $M_f(K^{\text{sep}})$. Then α is one of the roots of the polynomial $f^{(n)}(X)/f^{(n-1)}(X)$, where $f^{(n)}(X)$ is n-th iterate of $f(X)$. Since $f(X)/X = X^{q-1} + \pi$, then

$$\frac{f^{(n)}(X)}{f^{(n-1)}(X)} = f^{(n-1)}(X)^{q-1} + \pi.$$

By the Eisenstein criterion, this is an irreducible polynomial with degree $q^n - q^{n-1}$. Thus, according to Galois theory, the order of $\text{Gal}(K_{\pi,n}/K)$ is at least $(q-1)q^{n-1}$. On the other hand, $U_K/U_K^{(n)}$ has exactly $(q-1)q^{n-1}$ elements. Therefore, $\text{Gal}(K_{\pi,n}/K) \cong U_K/U_K^{(n)}$. Taking an inverse limit, it follows that

$$\text{Gal}(K^{\text{sep}}/K) \cong \varprojlim \text{Gal}(K_{\pi,n}/K) \cong \varprojlim U_K/U_K^{(n)} \cong U_K.$$

□

Corollary 2.2.8 ([5, Chap. VI 3.3 Theorem 3])**.** *The element* π *is a norm from* $K_{\pi,n}$*.*

Proof. Let $\alpha \in \Lambda_{f,n}$ be a primitive element, then $f^{(n)}(X)/f^{(n-1)}(X)$ is a monic polynomial with constant coefficient π, and the norm of $-\alpha$ is π. □

Let K^{ur} be the maximal unramified extension of K. Since K_π is a totally ramified extension of K, K_π and K^{ur} are linearly disjoint. As a result, the Galois group $\mathrm{Gal}(K_\pi K^{\mathrm{ur}}/K)$ is the direct product of $\mathrm{Gal}(K_\pi/K)$ and $\mathrm{Gal}(K^{\mathrm{ur}}/K)$.

Let $\pi \in \mathcal{O}_K$ be a uniformizer and let L_π be the field $K_\pi K^{\mathrm{ur}}$. We define a group homomorphism $r_\pi : K^\times \to \mathrm{Gal}(L_\pi/K)$ as follows

1. $r_\pi(\pi)$ acts as identity on K_π and acts as Frobenius automorphism σ on K^{ur}.
2. Given a $u \in U_K$, we define $r_\pi(u)$ to act as $[u^{-1}]_f$ on K_π and as the identity on K^{ur}.

Next, we prove that the field L_π and the homomorphism r_π are independent of the choice of uniformizer π. Moreover, L_π is a maximal abelian extension of K.

To prove these results, we first prove two auxiliary lemmas.

Lemma 2.2.9 ([5, Chap. VI 3.3 Lemma 2])**.** *Let E be an algebraic extension, finite or infinite, of a local field, and let $\alpha \in \widehat{E}$, where $\widehat{E}$ denotes the completion of E. If α is separable algebraic over E, then $\alpha \in E$.*

Proof. Let E^{sep} be the separable closure of E, and let E' be the adherence, that is, the topological closure of E in E^{sep}. Since α is separable algebraic over E, after choosing an E-embedding of $E(\alpha)$ into E^{sep}, we may view α as an element of E'. Hence it is enough to show that $E' = E$.

Let $g \in \mathrm{Gal}(E^{\mathrm{sep}}/E)$. Since g is continuous and is the identity on E, it is also the identity on the closure E'. Therefore $\mathrm{Gal}(E^{\mathrm{sep}}/E) = \mathrm{Gal}(E^{\mathrm{sep}}/E')$. By the fundamental theorem of Galois theory, applied to the Galois extension E^{sep}/E, the fixed fields of these two groups are equal. Thus $E' = E$. □

Let $\sigma \in \mathrm{Gal}(K^{\mathrm{ur}}/K)$ be the Frobenius automorphism. We claim that σ extends uniquely to $\widehat{K^{\mathrm{ur}}}$ by continuity.

Proof of the claim. Let v denote the normalized valuation on K. Since K is complete, v extends uniquely to every algebraic extension of K, and hence in particular to K^{ur}. Therefore every K-automorphism of K^{ur} preserves this valuation. Thus, for all $x, y \in K^{\mathrm{ur}}$,

$$v(\sigma(x) - \sigma(y)) = v(\sigma(x - y)) = v(x - y).$$

Equivalently, σ is an isometry for the corresponding nonarchimedean absolute value.

Hence σ is uniformly continuous. Since K^{ur} is dense in $\widehat{K^{\mathrm{ur}}}$, the map σ extends uniquely to a continuous map $\widehat{\sigma} : \widehat{K^{\mathrm{ur}}} \to \widehat{K^{\mathrm{ur}}}$. The inverse σ^{-1} is also an isometry, so it extends as well, and its extension is inverse to $\widehat{\sigma}$. Thus $\widehat{\sigma}$ is an automorphism of $\widehat{K^{\mathrm{ur}}}$ extending σ. By abuse of notation, we shall again denote this extension by σ. □

Let π be a uniformizer of $\mathcal{O}_K$, and let ω be another uniformizer. Write $\omega = u\pi$ with $u \in U_K$. Let $f \in \mathcal{F}_\pi$ and $g \in \mathcal{F}_\omega$.

Lemma 2.2.10 ([15, Lemma 2])**.** *There exists a power series $\phi(X)$ over $\widehat{\mathcal{O}_{K^{\mathrm{ur}}}}$ such that $\phi(X) \equiv \epsilon X \pmod{\deg 2}$ for some unit $\epsilon \in \widehat{\mathcal{O}_{K^{\mathrm{ur}}}}^\times$, and such that*

$$^\sigma\phi = \phi \circ [u]_f, \tag{2.2.10.1}$$

$$\phi(F_f(X, Y)) = F_g(\phi(X), \phi(Y)), \tag{2.2.10.2}$$

$$\phi \circ [a]_f = [a]_g \circ \phi, \qquad \forall a \in \mathcal{O}_K. \tag{2.2.10.3}$$

Here $^\sigma\phi$ denotes the power series obtained by applying σ to each coefficient of ϕ.

Proof. We first construct a power series $\varphi(X) \in \widehat{\mathcal{O}_{K^{\mathrm{ur}}}}[\![X]\!]$ satisfying (2.2.10.1). We will then modify it so that it also satisfies (2.2.10.2) and (2.2.10.3).

Recall that $\sigma - 1$ is surjective on the additive group of $\widehat{\mathcal{O}_{K^{\mathrm{ur}}}}$, and also on the group $\widehat{\mathcal{O}_{K^{\mathrm{ur}}}}^{\times}$ of units. Hence there exists a unit $\epsilon \in \widehat{\mathcal{O}_{K^{\mathrm{ur}}}}^{\times}$ such that ${}^{\sigma}\epsilon = \epsilon u$.

Set $\varphi_1(X) = \epsilon X$. Then ${}^{\sigma}\varphi_1(X) \equiv \varphi_1([u]_f(X)) \pmod{\deg 2}$.

Suppose inductively that

$$ {}^{\sigma}\varphi_n(X) \equiv \varphi_n([u]_f(X)) \pmod{\deg(n+1)}. $$

Let c be the coefficient of X^{n+1} in ${}^{\sigma}\varphi_n(X) - \varphi_n([u]_f(X))$. Choose $a \in \widehat{\mathcal{O}_{K^{\mathrm{ur}}}}$ such that

$$ a - {}^{\sigma}a = \frac{c}{({}^{\sigma}\epsilon)^{n+1}}, $$

which is possible by the surjectivity of $\sigma - 1$. Define

$$ \varphi_{n+1}(X) = \varphi_n(X) + a\epsilon^{n+1}X^{n+1}. $$

Since ${}^{\sigma}\epsilon = \epsilon u$, the coefficient of X^{n+1} in ${}^{\sigma}\varphi_{n+1}(X) - \varphi_{n+1}([u]_f(X))$ is

$$ c + {}^{\sigma}a({}^{\sigma}\epsilon)^{n+1} - a\epsilon^{n+1}u^{n+1} = c - \big(a - {}^{\sigma}a\big)({}^{\sigma}\epsilon)^{n+1} = 0. $$

Therefore

$$ {}^{\sigma}\varphi_{n+1}(X) \equiv \varphi_{n+1}([u]_f(X)) \pmod{\deg(n+2)}. $$

Passing to the coefficient-wise limit, we obtain a power series $\varphi = \lim_n \varphi_n$ satisfying ${}^{\sigma}\varphi = \varphi \circ [u]_f$. Since $\varphi(X) \equiv \epsilon X \pmod{\deg 2}$ with ϵ a unit, φ has a substitution inverse, denoted by φ^{-1}.

Now define

$$ h = {}^{\sigma}\varphi \circ f \circ \varphi^{-1}. $$

Since ${}^{\sigma}\varphi = \varphi \circ [u]_f$ and $f = [\pi]_f$, we have

$$ h = \varphi \circ [u]_f \circ [\pi]_f \circ \varphi^{-1} = \varphi \circ [\omega]_f \circ \varphi^{-1}. $$

Applying σ to this equality gives

$$ {}^{\sigma}h = {}^{\sigma}\varphi \circ [\omega]_f \circ ({}^{\sigma}\varphi)^{-1} = \varphi \circ [u]_f \circ [\omega]_f \circ [u^{-1}]_f \circ \varphi^{-1} = h. $$

Hence the coefficients of h are fixed by σ, and therefore lie in $\mathcal{O}_K$.

Moreover, $h(X) \equiv \omega X \pmod{\deg 2}$. Also, using $f(X) \equiv X^q \pmod{\pi}$ and the fact that σ lifts the q-power Frobenius on the residue field, we get

$$ h(X) = {}^{\sigma}\varphi(f(\varphi^{-1}(X))) \equiv {}^{\sigma}\varphi\big((\varphi^{-1}(X))^q\big) \equiv {}^{\sigma}\varphi\big(({}^{\sigma}\varphi)^{-1}(X^q)\big) = X^q \pmod{\pi}. $$

Since π and ω generate the same maximal ideal, this shows that $h \in \mathcal{F}_{\omega}$.

We now replace φ by

$$ \phi = [1]_{g,h} \circ \varphi. $$

Since $[1]_{g,h}$ has coefficients in $\mathcal{O}_K$, it is fixed by σ. Thus

$$ {}^{\sigma}\phi = [1]_{g,h} \circ {}^{\sigma}\varphi = [1]_{g,h} \circ \varphi \circ [u]_f = \phi \circ [u]_f, $$

so (2.2.10.1) still holds.

Next, since $[1]_{g,h}$ is an isomorphism from F_h to F_g, it satisfies $g \circ [1]_{g,h} = [1]_{g,h} \circ h$. Hence

$$^{\sigma}\phi \circ f \circ \phi^{-1} = [1]_{g,h} \circ {}^{\sigma}\varphi \circ f \circ \varphi^{-1} \circ [1]_{g,h}^{-1} = [1]_{g,h} \circ h \circ [1]_{g,h}^{-1} = g.$$

Combining this with (2.2.10.1), we also have $g = \phi \circ [\omega]_f \circ \phi^{-1}$.

To prove (2.2.10.2), define

$$F(X,Y) = \phi\big(F_f(\phi^{-1}(X), \phi^{-1}(Y))\big).$$

Then $F(X,Y) \equiv X + Y \pmod{\deg 2}$. Here and below, for any one-variable power series h, we abuse notation by writing hF for $h \circ F$ and Fh for $F(h(X), h(Y))$. Moreover, using $g = \phi \circ [\omega]_f \circ \phi^{-1}$ and the fact that $[\omega]_f$ is an endomorphism of F_f, we have

$$gF = \phi[\omega]_f\phi^{-1}\phi F_f\phi^{-1} = \phi[\omega]_f F_f\phi^{-1} = \phi F_f[\omega]_f\phi^{-1} = \phi F_f\phi^{-1}(\phi[\omega]_f\phi^{-1}) = \phi F_f\phi^{-1}g = Fg.$$

By the uniqueness statement in Lemma 2.1.2, it follows that $F = F_g$. This proves (2.2.10.2).

Finally, fix $a \in \mathcal{O}_K$ and set $A_a = \phi \circ [a]_f \circ \phi^{-1}$. Then $A_a(X) \equiv aX \pmod{\deg 2}$. Also,

$$A_a \circ g = \phi \circ [a]_f \circ [\omega]_f \circ \phi^{-1} = \phi \circ [\omega]_f \circ [a]_f \circ \phi^{-1} = g \circ A_a.$$

Again by the uniqueness statement in Lemma 2.1.2, we get $A_a = [a]_g$. Hence (2.2.10.3) holds. □

Theorem 2.2.11 ([15, Theorem 3])**.** *The field L_π is independent of the choice of uniformizer π.*

Proof. In this proof, write

$$L_\pi = K^{\mathrm{ur}}(\Lambda_f) \qquad \text{and} \qquad L_\omega = K^{\mathrm{ur}}(\Lambda_g),$$

where $f \in \mathcal{F}_\pi$ and $g \in \mathcal{F}_\omega$.

By Lemma 2.2.10, there exists a power series ϕ giving an isomorphism from the formal $\mathcal{O}_K$-module associated with f to the one associated with g. In particular, $\phi(\Lambda_f) = \Lambda_g$.

Since the coefficients of ϕ lie in $\widehat{\mathcal{O}_{K^{\mathrm{ur}}}}$, for every $\lambda \in \Lambda_f$ we have $\phi(\lambda) \in \widehat{K^{\mathrm{ur}}(\Lambda_f)} = \widehat{L_\pi}$. Thus $\Lambda_g \subset \widehat{L_\pi}$. Each element of Λ_g is separable algebraic over L_π, so Lemma 2.2.9 implies that $\Lambda_g \subset L_\pi$. Therefore $L_\omega \subset L_\pi$.

By symmetry, $L_\pi \subset L_\omega$. Hence $L_\pi = L_\omega$. □

Theorem 2.2.12 ([15, Theorem 3])**.** *The homomorphism r_π is independent of the choice of uniformizer π.*

Proof. It is enough to prove the following assertion: if $\omega = u\pi$ is another uniformizer, then

$$r_\pi(\omega) = r_\omega(\omega).$$

Indeed, applying this assertion with ω replaced by an arbitrary uniformizer shows that all the maps r_π have the same value on every uniformizer. Since the set of uniformizers generates $K^\times$, the homomorphisms r_π are equal.

Fix $f \in \mathcal{F}_\pi$ and $g \in \mathcal{F}_\omega$. By the previous theorem, we may identify

$$L_\pi = L_\omega = K^{\mathrm{ur}}(\Lambda_g).$$

On K^{ur}, both $r_\pi(\omega)$ and $r_\omega(\omega)$ induce the Frobenius automorphism. Hence it remains to compare their actions on Λ_g. By definition, $r_\omega(\omega)$ is the identity on Λ_g.

By Lemma 2.2.10, there exists a power series ϕ such that $\phi(\Lambda_f) = \Lambda_g$ and ${}^{\sigma}\phi = \phi \circ [u]_f$. Therefore it is enough to prove that, for every $\lambda \in \Lambda_f$,

$$ {}^{r_\pi(\omega)}\phi(\lambda) = \phi(\lambda). $$

Write $r_\pi(\omega) = r_\pi(u)r_\pi(\pi)$, and set $\tau = r_\pi(u)$ and $\sigma = r_\pi(\pi)$. By definition, σ acts as Frobenius on $\widehat{K^{\mathrm{ur}}}$ and fixes Λ_f, while τ fixes $\widehat{K^{\mathrm{ur}}}$ and sends $\lambda \in \Lambda_f$ to $[u^{-1}]_f(\lambda)$. Since the coefficients of ϕ lie in $\widehat{\mathcal{O}_{K^{\mathrm{ur}}}}$, we get

$$ {}^{r_\pi(\omega)}\phi(\lambda) = {}^{\tau\sigma}\phi(\lambda) = {}^{\sigma}\phi({}^{\tau}\lambda) = \phi([u]_f([u^{-1}]_f(\lambda))) = \phi(\lambda). $$

Thus $r_\pi(\omega) = r_\omega(\omega)$, and the theorem follows. □

Corollary 2.2.13 ([15, Corollary])**.** *Let π be a uniformizer of $\mathcal{O}_K$. Then $L_\pi K^{\mathrm{ur}}$ is the maximal abelian extension of K, and r_π is the local reciprocity homomorphism for $L_\pi K^{\mathrm{ur}}/K$. More precisely, for every $a \in K^\times$, $r_\pi(a) = (a, L_\pi K^{\mathrm{ur}}/K)$, with respect to our normalization of the reciprocity map.*

Proof. By the previous theorem, r_π is independent of the choice of uniformizer. Thus we may denote this common homomorphism simply by $r : K^\times \to \mathrm{Gal}(L_\pi K^{\mathrm{ur}}/K)$.

We first explain why the maximality of $L_\pi K^{\mathrm{ur}}$ follows once we know that r is the local reciprocity homomorphism. Since $L_\pi K^{\mathrm{ur}}$ contains K^{ur}, if $r(a)$ is trivial on $L_\pi K^{\mathrm{ur}}$, then its restriction to K^{ur} is trivial. But this restriction is $\sigma^{v_K(a)}$, where σ is the Frobenius automorphism. Hence $v_K(a) = 0$, so $a \in U_K$.

Now let $u \in U_K$. If $r(u)$ is trivial on $L_{\pi,m}$, then, by the definition of r_π on units, the endomorphism $[u^{-1}]_f$ acts trivially on $\Lambda_{f,m}$. Equivalently, $u \equiv 1 \pmod{\pi^m}$. If $r(u)$ is trivial on all of $L_\pi = \bigcup_{m\geq 1} L_{\pi,m}$, then this congruence holds for every m, and therefore $u = 1$. Hence the restriction of r to U_K is injective, and consequently r itself is injective.

Therefore, if r is the restriction of the local reciprocity map to $L_\pi K^{\mathrm{ur}}$, then the corresponding kernel is trivial. By the Galois correspondence in local class field theory, this means that $L_\pi K^{\mathrm{ur}}$ corresponds to the trivial subgroup of $K^\times$. Hence $L_\pi K^{\mathrm{ur}}$ is the maximal abelian extension of K.

It remains to show that r agrees with the local reciprocity map. Let $s : K^\times \to \mathrm{Gal}(L_\pi K^{\mathrm{ur}}/K)$ be the local reciprocity homomorphism, so that $s(a) = (a, L_\pi K^{\mathrm{ur}}/K)$.

Let π be any uniformizer of $\mathcal{O}_K$. By Theorem 2.2.8, the element π is a norm from $L_{\pi,m}/K$ for every m. Hence, by the defining property of the local reciprocity map, $s(\pi)$ acts trivially on $L_\pi = \bigcup_{m\geq 1} L_{\pi,m}$. On the other hand, its restriction to K^{ur} is the Frobenius automorphism σ.

By definition, $r_\pi(\pi)$ has exactly the same two properties: it is the identity on L_π, and it acts as σ on K^{ur}. Therefore $s(\pi) = r_\pi(\pi)$. Since $r_\pi = r$ by the independence of the uniformizer, we get $s(\pi) = r(\pi)$ for every uniformizer π.

Finally, the uniformizers generate $K^\times$. Indeed, after fixing one uniformizer π, every element of $K^\times$ can be written as $u\pi^n$ with $u \in U_K$. Thus s and r agree on a set of generators of $K^\times$, and hence $s = r$. This proves that r_π is the local reciprocity homomorphism. □

2.3 Connection with functional equation lemma

In this section, we discuss the connection between the formal group laws obtained from the functional equation lemma and Lubin–Tate formal group laws. Throughout this section, we

fix a nonarchimedean local field K. Let $\mathcal{O}_K$ be its ring of integers, π a uniformizer, and q the cardinality of its residue field.

Proposition 2.3.1. *The Lubin–Tate formal group laws F_f associated to $f \in \mathcal{F}_\pi$ are in one-to-one correspondence with formal group laws obtained by the functional equation with parameters $I = (\pi), \sigma = \mathrm{id}, s_1 = \pi^{-1}$ and $\forall i \geq 2, s_i = 0$. The functional equation in the sense of these parameters is*

$$f_g(X) = g(X) + \pi^{-1} f_g(X^q).$$

This correspondence is given by

$$g(X) \mapsto f_g^{-1}(\pi f_g(X)) \in \mathcal{F}_\pi.$$

Proof. The integrality of the coefficients in $f_g^{-1}(\pi f_g(X))$ follows from the functional equation lemma. Multiplying both sides of the functional equation by π, we have that

$$\pi f_g(X) = \pi g(X) + \pi^{-1}\pi f_g(X).$$

Then $\pi f_g(X) = f_{\pi g}(X)$. According to the second part of the functional equation lemma, we have that

$$f_g^{-1}(\pi f_g(X)) = f_g^{-1}(f_{\pi g}(X)) \in \mathcal{O}_K[\![X]\!].$$

The formal group law obtained by the functional equation lemma is

$$F(X, Y) = f_g^{-1}(f_g(X) + f_g(Y)).$$

By the existence and uniqueness in Theorem 2.1.4, it is enough to check that $f_g^{-1}(\pi f_g(X))$ is an endomorphism for $F(X, Y)$. This can be done as follows:

$$\begin{aligned} f_g^{-1}(\pi f_g(F(X,Y))) &= f_g^{-1}(\pi f_g(f_g^{-1}(f_g(X) + f_g(Y)))) \\ &= f_g^{-1}(\pi f_g(X) + \pi f_g(Y)) \\ &= F(f_g^{-1}(\pi f_g(X)), f_g^{-1}(\pi f_g(Y))). \end{aligned}$$

□

3 Lean and mathlib

The practice of mathematics is fundamentally rooted in the rigorous verification of proofs. However, as mathematical theories grow increasingly complex, the traditional peer-review process becomes strained. The digitization and formal verification of mathematics offer a modern solution to this problem, allowing mathematicians to verify their results with absolute certainty using software.

This chapter introduces the Lean theorem prover and its mathematical library, called `Mathlib`.

3.1 The Lean theorem prover

Lean is a functional programming language and interactive theorem prover initially developed by Leonardo de Moura at Microsoft Research in 2013 [9]. The current version, Lean 4, is a complete reimplementation that serves simultaneously as a proof assistant and a general-purpose, practically efficient programming language.

At its logical core, Lean is based on a variant of dependent type theory known as the Calculus of Inductive Constructions. In this framework, propositions are represented as types, and proofs are represented as terms of those types—a correspondence known as the Curry–Howard isomorphism.

To prove a theorem in Lean is to construct a term of the corresponding type. Lean's kernel, a small and highly trusted piece of code, checks that this term is well-typed. If the kernel accepts the term, the proof is logically sound. This minimizes the trusted code base; users do not need to trust the complex tactics or automation used to generate the proof, only the kernel that verifies the final proof term.

3.2 Dependent type theory

Traditional set theory (such as Zermelo–Fraenkel with Choice, ZFC) has historically served as the foundation of mathematics. However, dependent type theory provides a more natural environment for machine verification.

In Lean's type theory, every expression has a specific type. The defining feature of *dependent type theory* is that types can depend on terms. For example, the type of a vector of length n, denoted `Vector α n`, depends on the natural number n, where α is a type. This allows for highly expressive typing where the properties of mathematical objects are baked directly into their definitions.

Moreover, even types themselves have types. The type of propositions `Prop` is also of type `Type`. There is a *hierarchy of type* to prevent to not run into a paradox called *Girard's paradox*. Girard's paradox is an inconsistency that arises in certain very expressive type systems, especially those allowing a "type of all types." It is the type-theoretic analogue of Russell's paradox in naive set theory. Roughly speaking, if a system allows types to quantify over themselves too freely, then one can construct a self-referential type that leads to a contradiction. More discussion of this can be found in [8].

In hierarchy of type, the type of `Type` is `Type 1`, the type of `Type 1` is `Type 2`, and so forth. When we want to declare that `R` is a type residing in an arbitrary universe within this hierarchy, we write `R : Type*`.

3.3 A brief introduction to Lean syntax

Lean is both an interactive theorem prover and a functional programming language. Definitions, statements, and proofs are written in a syntax that is subsequently verified by the Lean kernel. In this section, we introduce several foundational pieces of Lean syntax that appear frequently in later chapters.

3.3.1 Basic declarations

A typical Lean file consists of a series of declarations. Beyond the foundational `def`, `theorem`, and `axiom`, Lean provides powerful organizational tools such as `abbrev`, `structure`, `class`, and `instance`.

The keyword `def` is used to define new data, functions, or values. For example, the following code defines the successor of a natural number:

```
def succ (n : Nat) : Nat := n + 1
```

Here, `succ` is the name; `(n : Nat)` declares an explicit input `n` of type `Nat`; the final `: Nat` gives the return type; and `:=` introduces the defining expression on the right.

Closely related is the keyword `abbrev`. It also creates a definition, but instructs Lean's elaborator to treat the new name as fully transparent during unification. It is frequently used for type aliases:

```
abbrev Vector3 := Float × Float × Float
```

Lean then automatically unfolds `Vector3` into a triplet of floats during type checking, avoiding the strict type-mismatch errors that a standard `def` might cause.

The keyword `theorem` is reserved for stating and proving propositions. A theorem requires a name, a logical statement, and a formal proof. For example:

```
theorem add_zero (a : Nat) : a + 0 = a := by
  rw [Nat.add_zero]
```

This theorem asserts that for every natural number `a`, the equality `a + 0 = a` holds. The portion before `:=` is the mathematical statement, while the portion after `:= by` enters "tactic mode" to construct the proof.

The keyword `axiom` introduces a statement assumed to be true without providing a proof:

```
axiom my_axiom : P
```

Axioms instruct Lean's kernel to accept `P` unconditionally and must be used sparingly to avoid compromising the soundness of the logical system.

To bundle multiple pieces of data together into a single mathematical object or record, Lean uses the `structure` keyword.

```
structure Point where
  x : Float
  y : Float
```

This declaration creates a new type called `Point`. It automatically generates a constructor to build points and projection functions (`Point.x` and `Point.y`) to access the underlying fields.

Finally, Lean handles algebraic hierarchies and overloading via the `class` and `instance` keywords. A `class` is essentially a `structure` that registers itself with Lean's typeclass synthesis mechanism (which resolves the square brackets `[]` discussed in Section 3.3.2).

```
class Add (α : Type) where
  add : α → α → α
```

This defines an interface requiring an addition operation for some type α. To fulfill this requirement for a specific type, we use the `instance` keyword to provide the concrete implementation:

```
instance : Add Nat where
  add x y := x + y
```

Once this instance is declared, any function requiring an `Add` instance for `Nat` will automatically find and use this implementation.

3.3.2 The meaning of different brackets

Lean utilizes different types of brackets to distinguish how arguments should be provided or inferred by the system.

Parentheses `()` denote *explicit arguments*. These are arguments that the user must manually provide when calling a function or applying a theorem.

```
def double (n : Nat) : Nat := n + n
```

In this example, `(n : Nat)` signifies that `n` is an explicit argument. Whenever `double` is used, the user must explicitly pass a natural number.

Curly brackets `{}` denote *implicit arguments*. Lean attempts to infer these arguments automatically from the surrounding context using unification.

```
def identity {α : Type} (x : α) : α := x
```

Here, `{α : Type}` indicates that α is an implicit type parameter. If we apply `identity` to a specific natural number, Lean automatically infers that α must be `Nat`, saving the user from writing redundant types.

Square brackets `[]` are designated for *typeclass arguments*. While also inferred automatically, they rely on Lean's typeclass synthesis mechanism rather than simple unification.

```
def addSelf [Add α] (x : α) : α := x + x
```

The expression `[Add α]` specifies that Lean must find a registered addition structure (a typeclass instance) for the type α. Lean searches its database to satisfy this requirement automatically when the function is invoked.

3.3.3 Attribute tags

Lean supports attribute tags, which append special metadata to declarations to guide automated tools. An attribute is declared directly above a definition, theorem, or structure using square brackets preceded by an `@` symbol.

The `simp` tactic invokes Lean's simplification engine, which applies a large library of preregistered lemmas to reduce a goal, often solving straightforward goals automatically.

A widely used attribute is `@[simp]`. Tagging a theorem with `@[simp]` adds it to this simplification database, allowing the `simp` tactic to apply it autonomously in future proofs.

```
@[simp]
theorem add_zero (a : Nat) : a + 0 = a := by
  rw [Nat.add_zero]
```

Once marked, whenever the simplifier encounters an expression matching `a + 0`, it will automatically rewrite it to `a` without requiring explicit user instruction.

Another essential attribute is `@[ext]`, which establishes extensionality principles. Extensionality asserts that two objects are equal whenever all their corresponding components are equal. Applying `@[ext]` directly to a `structure` automatically generates the corresponding extensionality lemma.

```
@[ext]
structure Point where
  x : Float
  y : Float
```

Tagging the `Point` structure with `@[ext]` instructs Lean's metaprogramming framework to generate a theorem behind the scenes. Written manually, it would look like this:

```
theorem Point.ext {p q : Point} (h₁ : p.x = q.x) (h₂ : p.y = q.y) : p = q
```

The `@[ext]` tag registers this logic so that invoking the `ext` tactic on a goal of `p = q` will automatically apply `Point.ext`, splitting the goal into the simpler subgoals `p.x = q.x` and `p.y = q.y`.

Attributes are indispensable for scaling mathematical libraries. Rather than manually proving and referencing hundreds of structural lemmas by name, developers register them with tags like `@[simp]` and `@[ext]`, enabling Lean's automation to carry the burden in complex proofs.

3.3.4 Namespaces

As mathematical libraries grow, multiple theorems or definitions with similar names become common. Lean uses namespaces to organize declarations and prevent such naming conflicts. When working extensively within a specific context, such as multivariate power series, repeatedly typing the full prefix `MvPowerSeries` becomes tedious and clutters the code. Enclosing our code in a `namespace` block lets Lean resolve local names automatically, so we may omit the prefix for any definition or theorem belonging to that namespace.

For instance, consider the following explicit definition of a variable X:

```
def MvPowerSeries.X (s : σ) : MvPowerSeries σ R :=
    (MvPowerSeries.monomial (Finsupp.single s 1)) 1
```

If we declare that we are operating inside the `MvPowerSeries` namespace, the code becomes significantly cleaner. We can drop the prefix from both the new definition's name (`X`) and any internal references to existing functions in that namespace (such as `monomial`):

```
namespace MvPowerSeries

def X (s : σ) : MvPowerSeries σ R :=
    monomial (Finsupp.single s 1) 1

end MvPowerSeries
```

Once outside the `end MvPowerSeries` boundary, a user would once again need to refer to this definition by its fully qualified name: `MvPowerSeries.X`.

A complementary tool for managing scopes is the `open` keyword. While `namespace` is typically used when *defining* new objects within a specific theory, `open` is used when we simply want to

access existing results from another namespace without typing its prefix. For example, suppose we have a definition that relies on `Finsupp.single`. If we want to use results from the finitely supported functions library frequently, we can open the `Finsupp` namespace directly:

```
open Finsupp
namespace MvPowerSeries

def X (s : σ) : MvPowerSeries σ R :=
    monomial (single s 1) 1

end MvPowerSeries
```

By writing `open Finsupp`, we instruct Lean to look inside the `Finsupp` namespace during name resolution, allowing us to reduce `Finsupp.single` to just `single`.

3.4 Structures, classes, and type class inference

To formalize mathematics effectively, Lean needs a way to organize abstract mathematical objects such as groups, rings, modules, and topological spaces. These objects usually consist of both data, such as operations, and properties, such as associativity or distributivity. Lean organizes this information using `structure`, `class`, and `instance`.

3.4.1 Structures and classes

A `structure` in Lean is similar to a record in programming. It bundles several pieces of data into a single object. In mathematics, this is useful because many objects are naturally described by a collection of operations together with axioms. For example, a group structure consists of a multiplication operation, an identity element, an inverse operation, and proofs of the group axioms.

A `class` is a special kind of structure which is registered with Lean's type class inference system. The fields of a class are still ordinary fields, just as in a structure, but Lean is allowed to search for instances of the class automatically. For this reason, algebraic and topological structures in mathlib, such as `Group`, `Ring`, `CommRing`, and `TopologicalSpace`, are usually implemented as classes.

For example, a theorem may be stated for an arbitrary type `R` together with an assumption `[CommRing R]`:

```
variable (R : Type*) [CommRing R]
```

The square brackets mean that Lean should treat `CommRing R` as an instance argument. In practice, this means that the ring operations and ring axioms on `R` are available automatically.

This is different from passing an ordinary proposition explicitly. A class is appropriate when the structure should be found automatically and used throughout many definitions and theorems. By contrast, an ordinary proposition is more suitable when the assumption is local and should be passed explicitly.

3.4.2 Instances

An `instance` provides a particular realization of a class for a specific type. For example, mathlib contains instances showing that the integers form a ring, that polynomial rings inherit ring structures, and that many constructions preserve algebraic structures.

Once an instance is available, Lean can use it without the user having to name it explicitly. For instance, if Lean knows `[CommRing R]`, then it can use the addition, multiplication, zero, one, negation, and all the ring laws associated with `R`. This is why ordinary mathematical notation such as `x + y`, `x * y`, and `-x` can be used in a general ring.

This mechanism is also important in the formalization of formal group laws. For example, after constructing an `AddCommGroup` instance on `F.Point` σ, Lean can use the notation `x + y`, `-x`, and `n · x` for points of a formal group law without any additional manual specification.

3.4.3 Type class inference

Type class inference is the mechanism by which Lean automatically finds the required instances. When a definition or theorem requires an assumption such as `[Ring R]` or `[TopologicalSpace X]`, Lean searches its environment for an appropriate instance and inserts it automatically.

For example, the notation `a + b` requires Lean to know an addition operation on the type of `a` and `b`. If `a b : R` and Lean has an instance `[Ring R]`, then Lean can infer the required addition operation from the ring structure. Thus the user can write expressions in a style close to ordinary mathematics, while Lean fills in the underlying algebraic data.

Type class inference is especially useful for generic mathematics. A single theorem can be stated for all rings, groups, or modules, and then applied to any concrete type for which Lean can synthesize the required instances.

3.4.4 The algebraic hierarchy

In mathlib, algebraic structures are organized into a large hierarchy; we refer to [19] for a more detailed discussion of its design and the subtleties of multiple inheritance in a dependently-typed setting. More complicated structures extend simpler ones. For example, a ring contains an additive commutative group structure and a multiplicative monoid structure, together with distributivity laws. A commutative ring further extends a ring by adding commutativity of multiplication.

This hierarchy is implemented using class extension. For example:

```
class Semiring (α : Type u) extends NonUnitalSemiring α,
  NonAssocSemiring α, MonoidWithZero α

class Ring (R : Type u) extends Semiring R, AddCommGroup R,
  AddGroupWithOne R

class CommRing (α : Type u) extends Ring α, CommMonoid α
```

Because of this hierarchy, an instance of `CommRing R` also provides access to the structures inherited from its parent classes, such as `Ring R` and `AddCommGroup R`. A theorem proved for a more general structure therefore applies automatically to a more specific one: a theorem about additive commutative groups can be used for any commutative ring. This design is essential for theorem reuse in mathlib. Instead of reproving the same basic facts separately for groups, rings, and fields, one proves a theorem at the appropriate level of generality, and type class inference makes it available in all more specialized contexts.

4 Lean formalization of formal group laws

In this chapter we discuss our formalization of formal group laws.

4.1 Multivariate power series in mathlib

In this section, we discuss the formalization of multivariate power series in mathlib.

A multivariate power series is defined to be a function from `(σ →₀ ℕ)` to `R`, where `(σ →₀ ℕ)` is a finite supported function from an index type `σ` to the natural numbers `ℕ` and `R` is usually a ring.

```
def MvPowerSeries (σ : Type*) (R : Type*) := (σ →₀ ℕ) → R
```

Mathlib provides several basic constructors for multivariate power series. The constructor `MvPowerSeries.monomial` builds a single monomial term. More precisely, if `n : σ →₀ ℕ` is a multi-index, then `MvPowerSeries.monomial n` is an `R`-linear map from `R` to `MvPowerSeries σ R`, which maps `n` to `1` and other to `0`.

Throughout this section, unless otherwise stated, all Lean code examples are written inside the namespace `MvPowerSeries`.

```
noncomputable def monomial (n : σ →₀ ℕ) : R →ₗ[R] MvPowerSeries σ R :=
    LinearMap.single R (fun _ ↦ R) n
```

The notation `→ₗ[R]` means that it is a `R`-linear map.

The keyword `noncomputable` means that Lean should not try to extract executable computational content from the declaration. This often appears in mathlib when a construction depends on classical choice or on definitions that are mathematically valid but not intended to be computed by evaluation. If many declarations in the same part of a file are noncomputable, one can write `noncomputable section` at the beginning of the section, so that later declarations in that section do not need to be individually marked with `noncomputable def`. For theorem proving, this is usually not a problem: one can still use these definitions in statements and proofs in the same way as other definitions.

A special case of monomial is the indeterminate indexed by `s : σ`, denoted as `X s`. In mathlib, `X s` is defined as the monomial whose exponent is `1` at `s` and `0` elsewhere.

```
def X (s : σ) : MvPowerSeries σ R :=
    monomial (Finsupp.single s 1) 1
```

`Finsupp.single s 1` is a function with single support at `s : σ` with value `1`.

The constant multivariate power series `MvPowerSeries.C` is defined to be the ring homomorphism from a semiring `R` to `MvPowerSeries σ R`, whose underlying function is `monomial (0 : σ →₀ ℕ)`.

Since a multivariate power series is represented as a function from multi-indices to coefficients, the most basic way to inspect such a series is to extract one of its coefficients. In mathlib this is done using `MvPowerSeries.coeff` . For a multi-index `n : σ →₀ ℕ`, the term `MvPowerSeries.coeff n` is an `R`-linear map from `MvPowerSeries σ R` to `R`. Applying it to a power series returns the coefficient of the monomial with exponent `n`.

```
def coeff (n : σ →₀ ℕ) : MvPowerSeries σ R →ₗ[R] R := LinearMap.proj n
```

Since `MvPowerSeries σ R` is definitionally the function type `(σ →₀ ℕ) → R`, this coefficient operation is essentially evaluation at the multi-index `n`.

The coefficient API gives convenient ways to describe the constructors above. We list below several relevant lemmas from mathlib, which express the behavior of `C`, `X`, and `monomial` in terms of their coefficients. The coefficient of `C a` is `a` at the zero multi-index and zero elsewhere; the coefficient of `X s` is `1` at the multi-index `Finsupp.single s 1` and zero elsewhere; and the coefficient of `monomial n a` at `n` is exactly `a`.

```
theorem coeff_C (n : σ →₀ ℕ) (a : R) :
    coeff n (C a) = if n = 0 then a else 0

theorem coeff_X (n : σ →₀ ℕ) (s : σ) :
    coeff n (X s) = if n = Finsupp.single s 1 then 1 else 0

theorem coeff_monomial_same (n : σ →₀ ℕ) (a : R) : coeff n (monomial n a) = a
```

Coefficient lemmas are also the standard way to prove equality of multivariate power series. The extensionality theorem `MvPowerSeries.ext` says that two power series are equal if all of their coefficients are equal.

Another important operation on multivariate power series is *substitution*. Suppose that `f : MvPowerSeries σ R` is a power series in variables indexed by σ. To substitute into `f`, we give, for each old variable `s : σ`, a new power series `a s : MvPowerSeries τ S`. Then `subst a f` should be interpreted as the result of replacing each variable `X s` in `f` by the power series `a s`. The resulting series has variables indexed by τ.

In mathlib, substitution is defined as a total function. However, for an infinite power series, not every formal substitution is mathematically well-behaved. Mathlib therefore introduces the predicate `HasSubst a` , which records the conditions under which the family `a : σ → MvPowerSeries τ S` can be substituted into a multivariate power series.

```
structure HasSubst (a : σ → MvPowerSeries τ S) : Prop where
  const_coeff (s : σ) : IsNilpotent (constantCoeff (a s))
  coeff_zero (d : τ →₀ ℕ) : {s : σ | coeff d (a s) ≠ 0}.Finite
```

The first condition says that the constant coefficient of each `a s` is nilpotent. The second condition says that, for each fixed exponent `d : τ →₀ ℕ`, finitely many of the series `a s` have a nonzero coefficient at `d`. These conditions ensure that the coefficients arising after substitution are finite sums.

A common special case is when the index type σ is finite and each substituted series has zero constant coefficient. In this situation mathlib provides the following theorem.

```
theorem hasSubst_of_constantCoeff_zero [Finite σ] {a : σ → MvPowerSeries τ S}
    (ha : ∀ s : σ, constantCoeff (a s) = 0) : HasSubst a
```

The substitution operation itself is written as `MvPowerSeries.subst` . In the general mathlib version, the coefficient ring may change from `R` to an `R`-algebra `S`.

```
def subst (a : σ → MvPowerSeries τ S) (f : MvPowerSeries σ R) :
    MvPowerSeries τ S
```

When the substituted family satisfies `HasSubst a`, the operation `subst a` is an `R`-algebra homomorphism. This algebra homomorphism is called `substAlgHom` in mathlib.

```
def substAlgHom {a : σ → MvPowerSeries τ S} (ha : HasSubst a) :
    MvPowerSeries σ R →ₐ[R] MvPowerSeries τ S
```

There is also a coefficient formula for substitution. It expresses the coefficient of `subst a f` at a multi-index `e` as a finite sum over the coefficients of the original series `f`.

```
theorem coeff_subst {a : σ → MvPowerSeries τ S} (ha : HasSubst a)
    (f : MvPowerSeries σ R) (e : τ →₀ ℕ) :
    coeff e (subst a f) = ∑ᶠ d : σ →₀ ℕ,
        coeff d f · coeff e (d.prod fun s n => a s ^ n)
```

Here $\sum^{\mathtt{f}}$ denotes a finitely supported sum, called `finsum` in mathlib. If a function is finitely supported, then `finsum` of this function is defined to be the finite sum over its support, otherwise it is defined to be zero.

The operation `MvPowerSeries.expand` is a special case of substitution. Given a nonzero natural number `p`, `expand p hp` replaces every variable `X i` by `X i ^ p`. In other words, it sends a monomial with exponent `d` to the monomial with exponent `p · d`, where the scalar multiplication `p · d` multiplies every exponent in the multi-index by `p`.

```
def expand (p : ℕ) (hp : p ≠ 0) :
    MvPowerSeries σ R →ₐ[R] MvPowerSeries σ R
```

The main computational lemmas for `expand` say that constants are unchanged, variables are sent to their `p`-th powers, and monomials have their multi-index scaled by `p`.

```
theorem expand_C (p : ℕ) (hp : p ≠ 0) (r : R) : expand p hp (C r) = C r

theorem expand_X (p : ℕ) (hp : p ≠ 0) (i : σ) : expand p hp (X i) = X i ^ p

theorem expand_monomial (p : ℕ) (hp : p ≠ 0) (d : σ →₀ ℕ) (r : R) :
    expand p hp (monomial d r) = monomial (p · d) r
```

In mathlib, `PowerSeries` is defined to be a special case of multivariate power series where σ is `Unit`.

For convenience, mathlib provides the command `name_power_vars`, which introduces local names for the variables of a multivariate power series ring of the form `MvPowerSeries (Fin n) R`. Here `Fin n` is the set containing `n` natural numbers starting at `0`. This is especially useful when working with a fixed small number of variables, since one can write familiar names such as `X`, `Y`, and `Z` instead of `MvPowerSeries.X (0 : Fin 3)`, `MvPowerSeries.X (1 : Fin 3)`, and `MvPowerSeries.X (2 : Fin 3)`. In Lean, these notations can be introduced as follows:

```
name_power_vars X, Y, Z over R
```

The notation introduced by this command is local, so it is only available in the surrounding scope.

4.2 Definitions

Let R be a commutative ring. Recall that a formal group law is defined to be a multivariate power series $F \in R[\![X, Y]\!]$ satisfies following two conditions:

1. $F(X, Y) = X + Y + \sum_{i,j \geq 1}^{\infty} c_{ij} X^i Y^j$.
2. $F(F(X, Y), Z) = F(X, F(Y, Z))$.

In this section, we fix some local notation for multivariate power series variables using the tactic `name_power_vars`. For power series in two variables, we write $\mathtt{X}_0$ and $\mathtt{X}_1$ for the two coordinate variables in `MvPowerSeries (Fin 2) R`. For power series in three variables, we write

Y_0, Y_1, and Y_2 for the three coordinate variables in `MvPowerSeries (Fin 3) R`. In Lean, these notations can be introduced as follows:

```
name_power_vars X₀, X₁ over R

name_power_vars Y₀, Y₁, Y₂ over R
```

A formal group laws is defined as

```
@[ext]
structure FormalGroup (R : Type*) [CommRing R] where
  /-- The underlying power series F(X,Y) in two variables. -/
  toPowerSeries : MvPowerSeries (Fin 2) R
  /-- The constant coefficient of the formal group law is zero. -/
  zero_constantCoeff : toPowerSeries.constantCoeff = 0
  /-- The coefficient of X in F(X,Y) is 1. -/
  lin_coeff_X : toPowerSeries.coeff (single 0 1) = 1
  /-- The coefficient of Y in F(X,Y) is 1. -/
  lin_coeff_Y : toPowerSeries.coeff (single 1 1) = 1
  /-- Associativity condition: F(F(X,Y),Z) = F(X,F(Y,Z)). -/
  assoc : toPowerSeries.subst ![toPowerSeries.subst ![Y₀, Y₁], Y₂] =
    toPowerSeries.subst ![Y₀, toPowerSeries.subst ![Y₁, Y₂]] (S := R)
```

In mathlib, suppose that `a b c` is of type α, then `![a, b]` can be viewed as the function from `Fin 2` sending `0` to `a` and `1` to `b`. Therefore, `![Y₀, Y₁]` is a function from `Fin 2` to `MvPowerSeries (Fin 2) R`, sending `0` to Y_0 and `1` to Y_1.

```
/-- The natural inclusion from formal group law into formal power series. -/
instance FormalGroup.coeToPowerSeries :
  Coe (FormalGroup R) (MvPowerSeries (Fin 2) R) := ⟨toPowerSeries⟩

/-- Given a formal group `F`, `F.IsComm` is a proposition that F(X,Y) = F(Y,X). -/
class FormalGroup.IsComm (F : FormalGroup R) : Prop where
  comm : F = (F : MvPowerSeries (Fin 2) R).subst ![X₁, X₀]
```

The instance `FormalGroup.coeToPowerSeries` allows Lean to coerce a formal group law to its underlying power series. Thus, if `F : FormalGroup R`, then Lean can interpret `F` as a term of type `MvPowerSeries (Fin 2) R` by using the field `F.toPowerSeries`.

The class `FormalGroup.IsComm F` records the commutativity condition for a formal group law. It states that `F` is equal to the power series obtained by substituting X_1 for the first variable and X_0 for the second variable. In ordinary mathematical notation, this says

$$F(X_0, X_1) = F(X_1, X_0).$$

In the early stages of this project, I define commutative formal group laws as a separate structure `CommFormalGroup` extending `FormalGroup`. There are some reasons for me to change this definition.

First, if commutative formal group laws were packaged as a new structure, then a commutative formal group law and its underlying formal group law would have different types. As a result, many definitions and lemmas stated for `FormalGroup R` would require additional coercions or separate restatements for `CommFormalGroup R`.

By contrast, the typeclass approach keeps the object itself as a term `F : FormalGroup R` and records commutativity as an additional property `[F.IsComm]`. Thus all constructions and theorems for general formal group laws can still be applied directly to `F`, while commutativity

can be requested only in the places where it is needed. Moreover, once an instance `[F.IsComm]` is available, Lean can find the commutativity assumption automatically by typeclass inference.

Recall that the additive inverse of a formal group law is a power series $i(X)$ satisfying that $F(X, i(X)) = 0$. In Lean, we construct this power series by induction as

```
abbrev addInv_aux (F : FormalGroup R) : ℕ → R
  | 0 => 0
  | 1 => -1
  | n + 1 => - (coeff (n + 1) (F.toPowerSeries.subst
    ![X, (∑ i : Fin (n + 1), C (addInv_aux F i.1) * X ^ i.1)]))

def addInv_X : PowerSeries R := PowerSeries.mk (addInv_aux F ·)

def addInv (φ : MvPowerSeries σ R) : MvPowerSeries σ R := subst φ (addInv_X F)
```

Here `addInv_aux F n` is the coefficient of degree `n` in the formal inverse of `F`. The first two coefficients are defined to be `0` and `1`. For the higher coefficients, we defines them recursively. Suppose the coefficients up to degree `n` have already been constructed. Then we have a truncated inverse series `∑ i : Fin (n + 1), C (addInv_aux F i.1) * X ^ i.1` and substitutes it into the second variable of `F`. And we define the $n+1$-th coefficient of additive inverse to be the minus $n+1$-th coefficient of this power series.

The function `PowerSeries.mk` constructs a one-variable formal power series from a sequence of coefficients. The function `addInv` φ is a multivariate power series obtained by substituting φ into the power series `addInv_X`. Mathematically, it is the additive inverse of φ in the sense of formal group law `F`.

We now use a formal group law `F` to define a new group structure on certain multivariate power series. More precisely, we consider power series satisfying `PowerSeries.HasSubst`, which means that their constant coefficient is nilpotent and hence they can be substituted into another power series. These power series will be called the *points* of the formal group law `F`.

In Lean, this type is defined as follows:

```
def Point (F : FormalGroup R) (σ : Type*) :=
  {f : MvPowerSeries σ R // PowerSeries.HasSubst f}
```

Here `F.Point` σ is a subtype of `MvPowerSeries` σ `R`. An element `x : F.Point` σ consists of an underlying power series `x.val` with a proof `x.prop` that it satisfies `PowerSeries.HasSubst`.

The formal group law `F` then defines an addition operation on these points. If `x y : F.Point` σ, their sum is obtained by substituting `x.val` and `y.val` into the two variables of `F`.

In Lean this is written as:

```
instance : Add (F.Point σ) where
  add x y := ⟨(F : MvPowerSeries (Fin 2) R).subst ![x.val, y.val], ...⟩
```

The expression `(F : MvPowerSeries (Fin 2) R).subst ![x.val, y.val]` means that we regard `F` as its underlying bivariate power series and substitute `x.val` for the first variable and `y.val` for the second variable. The second component of the pair proves that the resulting power series again satisfies `PowerSeries.HasSubst`.

The zero element is given by the zero power series:

```
instance : Zero (F.Point σ) where
  zero := ⟨0, PowerSeries.HasSubst.zero⟩
```

Thus `F.Point` σ has a natural zero element and an addition operation defined by the formal group law `F`. This addition is not ordinary addition of power series; it is the operation obtained by evaluating the formal group law on two points.

After defining addition and zero on `F.Point σ`, Mathlib then shows that these operations satisfy the usual algebraic laws. The main idea is that the formal group law axioms for `F` induce the corresponding group laws on the type of points `F.Point σ`.

Now we discuss how to obtain the group instance for `F.Point σ`.

The additive monoid structure bundles together the identity and associativity laws. The zero point is a left and right identity for the addition defined by `F`, and associativity of the formal group law gives associativity of the addition on points:

```
instance : AddMonoid (F.Point σ) where
  zero_add x := Subtype.ext (zero_add F x.prop)
  add_zero x := Subtype.ext (add_zero F x.prop)
  nsmul := nsmulRec
  add_assoc x y z := Subtype.ext <| F.assoc' x.prop y.prop z.prop
```

Here `zero_add` and `add_zero` say that adding the zero point on either side gives back the original point. Since `F.Point σ` is a subtype, Lean proves equality of two points by using `Subtype.ext`, reducing the goal to an equality of their underlying power series. The theorem `F.assoc'` is the pointwise form of the associativity condition of the formal group law: it says that substituting three valid points into the two sides of the associativity identity gives the same power series. The field `nsmul := nsmulRec` tells Lean to use the usual recursive definition of repeated addition by natural numbers.

If `F` is commutative, then the addition on `F.Point σ` is also commutative:

```
instance [F.IsComm] : AddCommMonoid (F.Point σ) where
  add_comm x y := Subtype.ext <| F.comm' x.prop y.prop
```

Here the assumption `[F.IsComm]` means that the formal group law itself satisfies `F(X,Y) = F(Y,X)`. The theorem `F.comm'` transfers this identity to points, proving that `x + y = y + x` for `x y : F.Point σ`.

The next important step is the additive group structure:

```
instance : Neg (F.Point σ) where
  neg f := ⟨F.addInv f.val, ...⟩

instance : AddGroup (F.Point σ) where
  nsmul := nsmulRec
  zsmul := zsmulRec
  neg_add_cancel x := ...
```

This instance says that every point has an additive inverse with respect to the formal group law operation. The proof of `neg_add_cancel` shows that `-x + x = 0`. The lemma `F.add_neg_cancel` is the corresponding inverse identity for the formal group law. The proof rewrites the expression using associativity, the zero laws, and the inverse law until the desired identity follows. The fields `nsmul := nsmulRec` and `zsmul := zsmulRec` specify the usual recursive definitions of multiplication by natural numbers and integers.

Finally, if `F` is commutative, the additive group is upgraded to an additive commutative group:

```
instance [F.IsComm] : AddCommGroup (F.Point σ) where
  add_comm x y := Subtype.ext <| F.comm' x.prop y.prop
```

Thus the formal group law `F` gives a natural group structure on `F.Point σ`. Without assuming commutativity, the points form an additive group. If the additional typeclass assumption `[F.IsComm]` is available, then this group is an additive commutative group.

4.2.1 Formal groups constructions

The two first examples of formal group laws are the additive formal group law $\mathbb{G}_a$ and the multiplicative formal group law $\mathbb{G}_m$. We formalize them using our definition of `FormalGroup R` and show that they are commutative.

```
def 𝔾_a : FormalGroup R where
  toPowerSeries := X₀ + X₁

instance : (𝔾_a (R := R)).IsComm where ...

def 𝔾_m : FormalGroup R where
  toPowerSeries := X₀ + X₁ + X₀ * X₁

instance : (𝔾_m (R := R)).IsComm where ...
```

Another way to construct new formal group laws is to change the coefficient ring of an existing one. Recall that given two commutative rings R and R', a ring homomorphism $f : R \to R'$, and a formal group law `F` over R, the coefficientwise image of `F` under f is again a formal group law over R'. This new formal group law is usually denoted by f_*F.

The construction of this pushforward on the coefficient ring is defined as

```
def map {R : Type*} [CommRing R] {R' : Type*} [CommRing R']
    (f : R →+* R') (F : FormalGroup R) : FormalGroup R' where
  toPowerSeries := F.toPowerSeries.map f
```

There is also a converse kind of construction. Sometimes a formal group law is originally defined over a larger ring R, but all of its coefficients actually lie in a smaller subring T. In this situation, the formal group law can be regarded as being defined over T.

In other words, given a formal group law `F` over a ring R, if every coefficient of `F` belongs to a subring T, then `F` descends to a formal group law over T. This is useful in our formalization of the functional equation lemma.

The construction of this is defined as:

```
def FormalGroup.toSubring (F : FormalGroup R) (T : Subring R)
    (hF : ∀ n, F.toFun n ∈ T) : FormalGroup T where
  toPowerSeries := F.toPowerSeries.toSubring _ hF
```

The function `MvPowerSeries.toSubring` is used to regard a power series whose coefficients lie in a subring as a power series over that subring. In this way, it changes the coefficient ring from the ring R to a subring T, provided that all coefficients of the power series actually belong to T.

Moreover, the constructions above are compatible with commutativity. More precisely, we prove that if the original formal group law `F` is commutative, then the formal group laws obtained by changing the coefficient ring are also commutative. In Lean, this gives instances showing that, under the assumption `F.IsComm`, both `(F.map f).IsComm` and `(F.toSubring T hF).IsComm` hold.

4.2.2 Homomorphism and isomorphism

Let R be a commutative ring and F, G be two formal group laws over R. Recall that a formal group homomorphism f from F to G is defined to be a power series with constant coefficient such that

$$f(F(X,Y)) = G(f(X), f(Y)).$$

It is defined in Lean as

```
@[ext]
structure FormalGroupHom (F G : FormalGroup R) where
  /-- The underlying power series of a formal group homomorphism. -/
  toPowerSeries : PowerSeries R
  /-- Constant coefficient of underlying power series is zero. -/
  zero_constantCoeff : toPowerSeries.constantCoeff = 0
  /-- The homomorphism condition: f(F(X,Y)) = G(f(X),f(Y)). -/
  hom : toPowerSeries.subst F =
    G.toPowerSeries.subst (toPowerSeries.toMvPowerSeries ·)
```

Here `toPowerSeries` is a power series, while `G.toPowerSeries` is a bivariate power series. Therefore, in the expression `G.toPowerSeries.subst (toPowerSeries.toMvPowerSeries ·)`, we need to regard the one-variable power series `toPowerSeries` as a multivariable power series in each variable of `G`. This is done using `PowerSeries.toMvPowerSeries`, whose definition is

```
def PowerSeries.toMvPowerSeries (i : σ) : PowerSeries R →ₐ[R] MvPowerSeries σ R :=
  MvPowerSeries.rename (fun _ => i)
```

The map `PowerSeries.toMvPowerSeries` is defined to be a special case of `MvPowerSeries.rename`. This operation renames the variables of a multivariable power series according to a map between indexing types. In this case, the map `fun _ => i` sends the variable of a power series to the variable `i` in `MvPowerSeries σ R`.

Thus, in the homomorphism condition

```
hom : toPowerSeries.subst F =
  G.toPowerSeries.subst (toPowerSeries.toMvPowerSeries ·)
```

The left-hand side represents $f(F(X,Y))$, while the right-hand side represents $G(f(X),f(Y))$. And the usage of the tag `@[ext]` can be found in Section 3.3.3.

A formal group homomorphism does more than relate the two underlying power series: it induces an actual map between the groups of points `F.Point σ` and `G.Point σ`. Given a homomorphism `f : FormalGroupHom F G` and a point `x : F.Point σ`, we substitute the power series `x.val` into the power series of `f` to obtain a point of `G`:

```
def FormalGroupHom.applyPoint (f : FormalGroupHom F G) (x : F.Point σ) :
    G.Point σ := ⟨f.toPowerSeries.subst x.val, ...⟩
```

To make this induced map convenient to use, we register `applyPoint` as a `CoeFun` instance. This lets us apply a homomorphism `f` directly to a point, writing `f x` instead of `f.applyPoint x`:

```
instance : CoeFun (FormalGroupHom F G)
    (fun _ ↦ {σ : Type*} → F.Point σ → G.Point σ) where
  coe f := f.applyPoint
```

The homomorphism condition is precisely the statement that this induced map respects the formal group addition. We therefore prove that it preserves addition, and then bundle the map together with the proofs that it preserves zero and addition into an `AddMonoidHom`:

```
lemma FormalGroupHom.map_add (f : FormalGroupHom F G) {x y : F.Point σ} :
    f (x + y) = f x + f y := ...

def FormalGroupHom.toAddMonoidHom (f : FormalGroupHom F G) :
    F.Point σ →+ G.Point σ where
  toFun := f.applyPoint
  map_zero' := ...
  map_add' _ _ := f.map_add
```

In this way, every formal group homomorphism gives rise to a genuine homomorphism of additive monoids between the associated groups of points. This allows us to reuse the entire Mathlib API for `AddMonoidHom` when reasoning about the map induced by a formal group homomorphism.

A formal group isomorphism is a formal group homomorphism which has an inverse formal group homomorphism. In Lean, this is packaged as the following structure:

```
@[ext]
structure FormalGroupIso (F G : FormalGroup R) where
  /-- The underlying formal group homomorphism of a formal group isomorphism. -/
  toHom : FormalGroupHom F G
  /-- The inverse homomorphism of underlying formal group homomorphism. -/
  invHom : FormalGroupHom G F
  /-- `toHom ∘ invHom = id`. -/
  left_inv : toHom.toPowerSeries.subst ∘ (PowerSeries.subst invHom.toPowerSeries) = id
  /-- `invHom ∘ toHom = id`. -/
  right_inv : invHom.toPowerSeries.subst ∘ (PowerSeries.subst toHom.toPowerSeries) =
    id
```

Thus `FormalGroupIso F G` bundles together two homomorphisms: `toHom : FormalGroupHom F G` and `invHom : FormalGroupHom G F`. The fields `left_inv` and `right_inv` state that these two homomorphisms are inverse to each other under composition of power series. More precisely, `left_inv` says that composing `invHom` and then `toHom` gives the identity, while `right_inv` says that composing `toHom` and then `invHom` gives the identity.

Therefore, a term of type `FormalGroupIso F G` is not merely a map from `F` to `G`. It contains a homomorphism, an inverse homomorphism, and proofs that the two compositions are the identity.

Just as a homomorphism induces a map on points, an isomorphism induces an isomorphism between the groups of points. Using the maps `applyPoint` of `toHom` and `invHom` together with the two inverse conditions, we package a formal group isomorphism into an `AddEquiv` between `F.Point σ` and `G.Point σ`:

```
def FormalGroupIso.toEquiv (α : FormalGroupIso F G) :
    F.Point σ ≃+ G.Point σ where
  toFun := α.toHom.applyPoint
  invFun := α.invHom.applyPoint
  left_inv x := ...
  right_inv y := ...
  map_add' _ _ := α.toHom.map_add
```

Here the fields `toFun` and `invFun` are the maps on points induced by `toHom` and `invHom`, while `left_inv` and `right_inv` follow from the `left_inv` and `right_inv` fields of the isomorphism. The field `map_add'` reuses `FormalGroupHom.map_add` for `toHom`. Thus a formal group isomorphism yields an isomorphism of additive groups `F.Point σ ≃+ G.Point σ`, and we may again invoke the Mathlib API for `AddEquiv`.

Recall that a homomorphism f from F to G is an isomorphism if and only if the coefficient of X in f is a unit in R. This result is formalized as:

```
theorem isIso_of_isUnit_coeff_one {F G : FormalGroup R} (f : FormalGroupHom F G)
    (h : IsUnit (f.toPowerSeries.coeff 1)) :
    ∃ (g : FormalGroupIso F G), g.toHom = f := ...

theorem isUnit_coeff_one_of_isIso {F G : FormalGroup R} (f : FormalGroupIso F G) :
    IsUnit (f.toHom.toPowerSeries.coeff 1) := ...
```

There are two ways to express an element `a` of a ring `R` to be invertible. The first one is `IsUnit a`, while the second one is `Invertible a`. The type `IsUnit a` is defined as a `Prop`. On the other hand, `Invertible a` is a typeclass (residing in `Type`) that explicitly packages the element a with its concrete inverse.

At the end of this section, we discuss the formalization of substitution inverse of a power series. Let R be a commutative ring. Recall that given a power series f over R, suppose that the coefficient of X in f is invertible in R, then there is a power series g over R such that $f(g(X)) = g(f(X)) = X$. This g is called substitution inverse of f.

In mathlib, it is defined as `substInv` and `substInvOfIsUnit` inside the `namespace PowerSeries`. `substInvOfIsUnit` is a variant of `substInv` using assumption `IsUnit` rather than `Invertible`.

4.3 The functional equation lemma

In this section, we discuss the formalization of the functional equation lemma. Throughout this section, all Lean declarations are made inside the `namespace FormalGroup`. To simplify the presentation, we omit the namespace keyword from the displayed code. We begin by formalizing the data appearing in the statement.

Let K be a commutative ring, let R be a subring of K, and let I be an ideal of R. Let p, q, and t be natural numbers, where p is prime and q is a power of p, say $q = p^t$, with $t \neq 0$. Let σ be a ring endomorphism of K, and let s be a sequence of elements of K.

In Lean, these assumptions are introduced as follows:

```
variable {K : Type*} [CommRing K] {R : Subring K} {I : Ideal R}
  {p t q : ℕ} [hp : Fact (Nat.Prime p)] (ht : t ≠ 0) (hq : q = p ^ t)
  (σ : K →+* K) (s : ℕ → K) {g : PowerSeries R}
```

Recall that $f_g(X)$ is defined to be the unique power series satisfying equation (1.2.1.1). We formalize $f_g(X)$ using the recursive formula of coefficient in equation (1.2.2.1) as

```
def RecurFunAux (hg : g.constantCoeff = 0) : ℕ → K
  | 0 => 0
  | k + 1 =>
    g.coeff (k + 1) + ∑ j ∈ (Icc 1 (multiplicity q (k + 1))).attach,
      have : (k + 1) / (q ^ (j : ℕ)) < k + 1 := ...
      (s j) * σ^[j] (RecurFunAux hg ((k + 1) / (q ^ (j : ℕ))))

def RecurFun := PowerSeries.mk (RecurFunAux ht hq σ s hg)
```

To define `RecurFunAux` recursively, Lean must check that every recursive call is made on a strictly smaller natural number. In the recursive step for `k + 1`, the recursive call is made at `(k + 1) / (q ^ (j : ℕ))`, so we need a proof that `((k + 1) / (q ^ (j : ℕ))) < k + 1` for every index `j` appearing in the sum. This is the reason for using `Finset.attach` . The summation is taken over `(Icc 1 (multiplicity q (k + 1))).attach`. An element of this attached finset is not just a natural number `j`, but a natural number together with a proof that it belongs to the interval `(Icc 1 (multiplicity q (k + 1)))` In other words, from such a `j` Lean can use the facts `1 ≤ j` and `j ≤ multiplicity q (k + 1)`.

These bounds are needed to prove that division by `q ^ j` strictly decreases `k + 1`. Once Lean has the proof `(k + 1) / (q ^ (j : ℕ)) < k + 1` the recursive call `RecurFunAux hg ((k + 1) / (q ^ (j : ℕ)))` is accepted as a structurally smaller call. Without `Finset.attach`, the

membership information about `j` would not be directly available inside the summation, and Lean would not have enough information to justify the recursive definition.

Recall that functional equation (1.2.1.1) is

$$f_g(X) = g(X) + \sum_{i=1}^{\infty} s_i \sigma_*^i f_g(X^{q^i}).$$

And it is formalized as

```
theorem FunEq_of_RecurFun [TopologicalSpace K] [T2Space K] (hs₀ : s 0 = 0) :
    let f := RecurFun ht hq σ s hg
    f = g.map R.subtype +
      ∑' (i : ℕ), s i · (f.expand (q ^ i) (q_pow_neZero hq)).map (σ^i) := ...
```

The assumptions `[TopologicalSpace K]` and `[T2Space K]` are needed because the right-hand side contains the infinite sum `∑' (i : ℕ), ...`. The topology on `K` induces the corresponding topology on `PowerSeries K`, so that Lean can interpret this expression as a limit of finite partial sums. The Hausdorff condition `[T2Space K]` ensures that such limits are unique, which is important when proving equality of power series defined by an infinite summation.

In the formula, `R.subtype` is the canonical ring homomorphism from the subring `R` to the ambient ring `K`. Thus `g.map R.subtype` regards the power series `g` over `R` as a power series over `K`. The term `(f.expand (q ^ i) (q_pow_neZero hq)).map (σ^i)` corresponds to the expression $\sigma_*^i f_g(X^{q^i})$ in the functional equation. A discussion of `MvPowerSeries.expand` and its one-variable version for power series can be found in Section 4.1.

Since the coefficient of X in $f_g(X)$ is invertible in K. Then we could the definition `substInv` to define its substitution inverse as

```
def inv_RecurFun := (RecurFun ht hq σ s hg).substInv
```

The following definition constructs the bivariate power series corresponding to $f_g^{-1}(f_g(X) + f_g(Y))$:

```
def inv_add_RecurFun :=
    (inv_RecurFun ht hq σ s hg hg_unit).subst
      ((RecurFun ht hq σ s hg).toMvPowerSeries (0 : Fin 2) +
        (RecurFun ht hq σ s hg).toMvPowerSeries 1)
```

Here `inv_RecurFun ht hq σ s hg hg_unit` denotes the compositional inverse of the power series `RecurFun ht hq σ s hg`. The expression `(RecurFun ht hq σ s hg).toMvPowerSeries (0 : Fin 2)` regards this one-variable power series as a bivariate power series in the first variable, so it represents $f_g(X)$. Similarly, `(RecurFun ht hq σ s hg).toMvPowerSeries 1` represents $f_g(Y)$, where the second variable is used instead.

Therefore, the whole expression defines the bivariate power series

$$F_g(X, Y) = f_g^{-1}(f_g(X) + f_g(Y)).$$

The first part of the functional equation lemma 1.2.3 states that $F_g(X, Y)$ actually has coefficients in the subring R. In Lean, this is formalized by proving that every coefficient of `inv_add_RecurFun` belongs to `R`:

```
theorem coeff_inv_add_mem_Subring [UniformSpace K] [T2Space K] [DiscreteUniformity K]
    (hs₀ : s 0 = 0) :
    ∀ n, (inv_add_RecurFun ht hq σ s hg hg_unit).coeff n ∈ R := ...
```

After this theorem, the power series $F_g(X,Y)$ gives the desired commutative formal group law over R.

```
def inv_add_RecurFun :=
    (inv_RecurFun ht hq σ s hg hg_unit).subst
      ((RecurFun ht hq σ s hg).toMvPowerSeries (0 : Fin 2) +
        (RecurFun ht hq σ s hg).toMvPowerSeries 1)
```

The `def inv_add_RecurFun` denotes the multivariate power series $F_g(X,Y) = f_g^{-1}(f_g(X) + f_g(Y))$. Also, `(RecurFun ht hq σ s hg).toMvPowerSeries (0 : Fin 2)` represents the multivariate power series $f_g(X)$, where X is the first indeterminate variable in a bivariate power series. Similarly, `(RecurFun ht hq σ s hg).toMvPowerSeries 1` represents $f_g(Y)$.

Recall that the first part of the functional equation lemma 1.2.3 says that $F_g(X,Y)$ has coefficients in R. Moreover, it is a commutative formal group law by the definition of $F_g(X,Y)$. The first part of the functional equation lemma is formalized as follows.

```
theorem coeff_inv_add_mem_Subring [UniformSpace K] [T2Space K] [DiscreteUniformity K]
    (hs₀ : s 0 = 0) : ∀ n, (inv_add_RecurFun ht hq σ s hg hg_unit).coeff n ∈ R := ...
```

Next, we prove that $F_g(X,Y)$ is a formal group law over K. The associativity condition follows from

$$F_g(F_g(X,Y),Z) = f_g^{-1}(f_g(X) + f_g(Y) + f_g(Z)) = F_g(X,F_g(Y,Z))$$

Using this coefficient result, we can descend the formal group law from K to R. This is done by applying `FormalGroup.toSubring` and the theorem `coeff_inv_add_mem_Subring`.

```
def invAdd_RecurFun [UniformSpace K] [T2Space K]
    [DiscreteUniformity K] (hs₀ : s 0 = 0) : FormalGroup R :=
  (FormalGroup.invAdd_RecurFun_Aux ht hq σ s hg hg_unit).toSubring _
    (coeff_inv_add_mem_Subring ...)
```

Moreover, this formal group law is commutative. This follows from the definition of $F_g(X,Y)$, since the expression $f_g(X) + f_g(Y)$ is symmetric in X and Y.

Thus the first part of the functional equation lemma 1.2.3 gives a commutative formal group law over R, constructed from the recursive power series f_g.

The second part of the functional equation lemma 1.2.3 concerns the comparison between two recursively defined power series. Suppose that g and g' are power series over R with zero constant coefficient. The lemma says that the relevant composition involving the inverse of one recursive series and the other recursive series again has coefficients in R.

In Lean, this is formalized as follows:

```
lemma coeff_inv_RecurFun_g'_mem_Subring [UniformSpace K] [T2Space K]
    [DiscreteUniformity K] (hs0 : s 0 = 0) :
    let f_g' := (RecurFun ht hq σ s hg')
    let G := (inv_RecurFun ht hq σ s hg hg_unit).subst f_g'
    ∀ n, PowerSeries.coeff n G ∈ R
```

Here `f_g'` denotes the recursive power series associated to `g'`. The power series `G` is obtained by substituting `f_g'` into the inverse recursive power series. The conclusion states that every coefficient of `G` lies in the subring `R`.

The third part of the functional equation lemma 1.2.3 gives a stability result under substitution. Let h be a power series over R with zero constant coefficient. The goal is to show that there exists a power series $\hat{h}$ over R such that

$$f_g(h(X)) = f_{\hat{h}}(X).$$

Using the functional equation (1.2.1.1), this is equivalent to showing that the power series

$$f_g(h(X)) - \sum_{n=1}^{\infty} s_n \cdot \sigma^n f_g(h(X^{q^n}))$$

has coefficients in R. This power series is then the desired $\hat{h}$.

The corresponding Lean statement is:

```
lemma coeff_inv_RecurFun_g'_mem_Subring' [UniformSpace K] [T2Space K]
    [DiscreteUniformity K] (hs₀ : s 0 = 0) {h : PowerSeries R}
    (h₀ : h.constantCoeff = 0):
    let f := (RecurFun ht hq σ s hg)
    let f₁ : PowerSeries K := f.subst (h.map R.subtype)
    ∀ n, (f₁ - ∑' i, (s i) • (f₁.expand (q ^ i) (q_pow_neZero hq)).map (σ ^ i)).coeff
    n ∈ R
```

In this statement, `f₁` denotes the composition $f_g(h(X))$, regarded as a power series over K. The infinite sum is the correction term appearing in the functional equation. The conclusion says that the coefficients of the resulting difference all lie in R.

The fourth part of the functional equation lemma 1.2.3 proves a congruence statement. It shows that congruence modulo a power of the ideal I is preserved after substituting into the recursive power series f_g.

```
theorem congr_equiv [UniformSpace K] [T2Space K] [DiscreteUniformity K]
    (hs₀ : s 0 = 0) {α : PowerSeries R} {β : PowerSeries K}
    (hα : α.constantCoeff = 0) (hβ : β.constantCoeff = 0)
    {r : ℕ} (hr : 1 ≤ r) :
    let f := RecurFun ht hq σ s hg
    (∀ n, α.coeff n - β.coeff n ∈ R.subtype '' ↑(I ^ r)) ↔
      (∀ n, PowerSeries.coeff n (f.subst (α.map R.subtype)) -
        PowerSeries.coeff n (f.subst β) ∈ R.subtype '' ↑(I ^ r))
```

There is a small technical point in this statement. The power series `α` is defined over R, while `β` is defined over K. Therefore, after comparing their coefficients, the difference `α.coeff n - β.coeff n` is regarded as an element of K. For this reason, the congruence condition is expressed as membership in a subset of K, rather than directly as membership in the ideal `I ^ r` of R.

The subset `R.subtype '' ↑(I ^ r)` is the image of the ideal `I ^ r` under the natural inclusion `R.subtype : R →+* K`. Thus it consists of those elements of K which come from elements of `I ^ r`. The symbol `↑` is needed because Lean coerces the ideal `I ^ r : Ideal R` to its underlying set `I ^ r : Set R` before taking its image under `R.subtype`.

Hence the theorem says that two power series are coefficientwise congruent modulo `I ^ r` before applying `f_g` if and only if their images under substitution into `f_g` are still coefficientwise congruent modulo `I ^ r`.

The full proofs of these statements can be found in the GitHub repository.

4.4 Universal formal group laws

In this section, we discuss the formalization of universal formal group laws. We first fix a commutative ring `R` and a formal group law `U` over `R`.

```
variable {R : Type u} [CommRing R] (U : FormalGroup R)
```

The collection of formal group laws can be viewed as a set-valued functor on the category of commutative rings. More precisely, to each commutative ring `R` it assigns the set of formal group laws over `R`, and to each ring homomorphism it assigns the corresponding pushforward of formal group laws along that homomorphism.

In Lean, this functor is defined as follows:

```
def FGL : CommRingCat.{u} ⇒ Type u where
  obj R := FormalGroup R
  map f := FormalGroup.map f.hom
  map_id R := by ext F : 1; simp [FormalGroup.map]
  map_comp f g := by ext F : 1; simp [FormalGroup.map]
```

Here ⇒ denotes the type of functors between two categories. Although the target category is written as `Type u` in Lean, mathematically this should be understood as the category of sets, or more precisely as the category of types in a fixed universe. Thus `FGL` is the functor

$$\mathrm{FGL} : \mathrm{CommRing} \to \mathrm{Set}.$$

The object part of the functor is given by `obj R := FormalGroup R`, so a commutative ring is sent to the type of formal group laws over that ring. If `f : X ⟶ Y` is a morphism in `CommRingCat`, then `f.hom` is the underlying ring homomorphism from `X` to `Y`. The map part `map f := FormalGroup.map f.hom` sends a formal group law over `X` to its coefficientwise pushforward over `Y`.

The fields `map_id` and `map_comp` prove the functorial properties: pushing forward along the identity homomorphism gives back the same formal group law, and pushing forward along a composite homomorphism is the same as pushing forward in two steps.

Recall that a one-dimensional formal group law `U` over a ring `R` is universal if every formal group law over any commutative ring is obtained uniquely from `U` by applying a ring homomorphism from `R`. In other words, for every commutative ring `T` and every formal group law `F` over `T`, there exists a unique ring homomorphism $\phi : R \to T$ such that the pushforward of `U` along φ is `F`.

This is formalized as follows:

```
def IsUniversal : Prop :=
    ∀ {T : Type u} [CommRing T], ∀ F, ∃! (φ : R →+* T), U.map φ = F
```

The definition `IsUniversal` is a proposition attached to the formal group law `U`. The variables `R` and `T` are implicit: `R` can be inferred from `U : FormalGroup R`, and `T` can be inferred from `F : FormalGroup T`. Since this definition is placed in the namespace `FormalGroup`, the statement that `U` is universal can be written as `U.IsUniversal`.

The universal property can also be expressed categorically. Under the assumption that `U` is universal, the functor `FGL` is corepresented by the ring `R` together with the universal element `U`.

```
theorem FGL_representable (h : U.IsUniversal) :
    FGL.IsCorepresentedBy U (X := of R) := ...
```

Here `of R` denotes the object of `CommRingCat` associated to the commutative ring `R`. The theorem says that `U` represents the universal element of the functor `FGL`. More explicitly, for every commutative ring `S`, there is a bijection

$$\mathrm{Hom}_{\mathrm{CommRing}}(R, S) \cong \mathrm{FGL}(S),$$

sending a ring homomorphism φ `: R →+* S` to the pushforward formal group law `U.map φ`.

The structure `Functor.IsCorepresentedBy` is the dual version of `Functor.IsRepresentedBy` in mathlib. Informally, a set-valued functor `F` is corepresented by an object `X` with universal element `x : F.obj X` if every element of `F.obj Y` is obtained uniquely by applying `F` to a morphism `X ⟶ Y`. Equivalently, for every object `Y`, the induced map $(X \to Y) \to F(Y)$ is a bijection.

This is the form used in our theorem: the universal formal group law `U` over `R` corepresents the functor `FGL`, because for every commutative ring `S`, ring homomorphisms from `R` to `S` are in bijection with formal group laws over `S`.

4.5 Heights

We now introduce the height of a commutative formal group law. In this section, we fix a field K of characteristic p, and we consider a commutative formal group law `F` over K.

The first step is to define the multiplication-by-`n` series of the formal group law. Since we have already constructed an `AddCommGroup` instance on `F.Point` σ under the assumption `F.IsComm`, we can directly use the scalar multiplication notation `n · x` for repeated addition of a point.

```
def series (n : ℕ) : FormalGroupHom F F where
  toPowerSeries := (n · ⟨PowerSeries.X, PowerSeries.HasSubst.X _⟩ : F.Point Unit).val
  zero_constantCoeff := ...
  hom := ...
```

Here ⟨`PowerSeries.X, ...`⟩ is the point of `F.Point Unit` given by the coordinate power series `PowerSeries.X`, and `n · ⟨PowerSeries.X, ...⟩` is its `n`-fold sum under the addition induced by `F`; taking `.val` extracts the underlying power series. Thus `F.series n` is the formal group homomorphism for multiplication by `n`, usually denoted $[n]_F(X)$. The field `zero_constantCoeff` shows it has constant coefficient zero, and `hom` proves compatibility with the formal group law structure; we omit these proofs.

The definition of height focuses on the multiplication-by-`p` series `F.series p`. The following theorem shows that, if it is nonzero, then it has a nonzero coefficient in some degree of the form `p ^ n`.

```
theorem exist_coeff_pow_ne_zero_iff_ne_zero :
    (∃ n, (F.series p).toPowerSeries.coeff (p ^ n) ≠ 0) ↔
      (F.series p).toPowerSeries ≠ 0 := ...
```

Using the alternative definition in Remark 1.7.4, when `F.series p` is nonzero, it yields a smallest `n` with a nonzero coefficient in degree `p ^ n`. The height of `F` is then defined as follows:

```
def height : ℕ∞ :=
  letI := Classical.decEq K
  letI := Classical.decEq (PowerSeries K)
  if h : (F.series p).toPowerSeries = 0 then ⊤ else
    Nat.find ((F.exist_coeff_pow_ne_zero_iff_ne_zero).mpr h)
```

The value of `height` lies in $\mathbb{N}\infty$, the natural numbers together with a top element $\top$. If `F.series p` is zero, the height is $\top$; otherwise `Nat.find` chooses the smallest `n` with a nonzero coefficient in degree `p ^ n`.

In ordinary notation, the height records the first degree of the form `p ^ n` in which `[p]_F(X)` has a nonzero coefficient: `F` has height `n` if

$$[p]_F(X) = aX^{p^n} + \text{higher order terms}$$

with `a` nonzero, and infinite height if `[p]_F(X) = 0`.

Conclusion and future work

Conclusion

In this thesis, we developed a Mathlib-compatible framework for formal group laws in Lean. Our main results can be summarized as follows.

First, we formalized formal group laws over a commutative ring together with their commutativity, homomorphisms, and isomorphisms, providing a foundational API for working with formal group laws in Lean.

Second, we proved that a formal group law induces a group structure on its points, which is commutative whenever the law is. This shows that the algebraic structure expected of formal group laws can be recovered and used formally.

Third, we formalized the functional equation lemma, including both the construction satisfying the functional equation and the descent statement that the resulting formal group law has coefficients in the required subring.

Along the way, we established a number of supporting results about multivariate power series, which provide reusable infrastructure for future work involving power series in Mathlib. Finally, we formalized the height of a formal group law, bridging the basic algebraic theory to deeper classification results and arithmetic applications.

Future Work

The framework developed in this thesis opens up several natural directions for further work.

The most immediate is Lubin–Tate theory, which uses formal group laws to construct highly structured abelian extensions of local fields. Formalizing it would combine our formal group law API with Mathlib's existing theory of local fields, valuations, and ramification, and would be a significant step toward explicit local class field theory.

A second direction concerns the formal group laws of elliptic curves. Every elliptic curve has an associated formal group law, obtained by completing the curve at the identity. Over a field of positive characteristic, the height of this law records important arithmetic information: it equals 1 in the ordinary case and 2 in the supersingular case. Formalizing this link would connect our theory to Mathlib's existing formalization of elliptic curves.

A third goal is Lazard's theorem, which asserts the existence of a universal formal group law and identifies its coefficient ring as a polynomial ring. Fundamental in the algebraic theory of formal group laws and central to complex cobordism in algebraic topology, its formalization would require a careful development of the Lazard ring and the accompanying universal constructions.

A fourth direction is the classification of formal group laws by height, the central invariant in positive characteristic. A natural long-term goal is to formalize the classification of formal group laws of a given height, including the construction of isomorphisms after a suitable base change, thereby extending the basic theory of this thesis to a more structural understanding.

Taken together, these directions build on the reusable infrastructure developed in this thesis as a foundation for further work in arithmetic geometry.

Bibliography

[1] Frank W. Anderson and Kent R. Fuller. *Rings and Categories of Modules*, volume 13 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, 2 edition, 1992.

[2] Dagur Asgeirsson, Riccardo Brasca, Nikolas Kuhn, Filippo A. E. Nuccio Mortarino Majno Di Capriglio, and Adam Topaz. Categorical foundations of formalized condensed mathematics. *Journal of Symbolic Logic*, 2024. Advance online publication.

[3] Lars Becker, María Inés de Frutos-Fernández, Leo Diedering, Floris van Doorn, Sébastien Gouëzel, Asgar Jamneshan, Evgenia Karunus, Edward van de Meent, Pietro Monticone, Jasper Mulder-Sohn, Jim Portegies, Joris Roos, Michael Rothgang, Rajula Srivastava, James Sundstrom, Jeremy Tan, and Christoph Thiele. A blueprint for the formalization of Carleson's theorem on convergence of Fourier series, 2025.

[4] Alex Best, Christopher Birkbeck, Riccardo Brasca, Eric Rodriguez Boidi, Ruben van De Velde, and Andrew Yang. A complete formalization of Fermat's last theorem for regular primes in Lean, 2025.

[5] J. W. S. Cassels and A. Fröhlich, editors. *Algebraic Number Theory*. Academic Press, London, 1967.

[6] Antoine Chambert-Loir and María Inés de Frutos-Fernández. A formalization of divided powers in Lean, 2025.

[7] Johan Commelin. Liquid Tensor Experiment. *Mitteilungen der Deutschen Mathematiker-Vereinigung*, 30(3):166–170, 2022.

[8] Thierry Coquand. An analysis of Girard's paradox. Research Report RR-0531, Inria, 1986.

[9] Leonardo de Moura, Soonho Kong, Jeremy Avigad, Floris van Doorn, and Jakob von Raumer. The Lean theorem prover (system description). In *Automated Deduction – CADE-25*, pages 378–388. Springer International Publishing, 2015.

[10] Leonardo de Moura and Sebastian Ullrich. The Lean 4 theorem prover and programming language. In *Automated Deduction – CADE 28*, pages 625–635. Springer International Publishing, 2021.

[11] Sidharth Hariharan, Christopher Birkbeck, Seewoo Lee, Ho Kiu Gareth Ma, Bhavik Mehta, Auguste Poiroux, and Maryna Viazovska. A milestone in formalization: The sphere packing problem in dimension 8, 2026.

[12] Michiel Hazewinkel. *Formal Groups and Applications*, volume 78 of *Pure and Applied Mathematics*. Academic Press, New York, 1978.

[13] Michel Lazard. Sur les groupes de Lie formels à un paramètre. *Bulletin de la Société Mathématique de France*, 83:251–274, 1955.

[14] Lean FRO. Lean Focused Research Organization: Progress and roadmap, 2024.

[15] Jonathan Lubin and John Tate. Formal complex multiplication in local fields. *Annals of Mathematics*, 81(2):380–387, 1965.

[16] Eben Matlis. Injective modules over Noetherian rings. *Pacific Journal of Mathematics*, 8(3):511–528, 1958.

[17] Joseph H. Silverman. *The Arithmetic of Elliptic Curves*, volume 106 of *Graduate Texts in Mathematics*. Springer, New York, 2nd edition, 2009.

[18] Maryna S. Viazovska. The sphere packing problem in dimension 8. *Annals of Mathematics*, 185(3):991–1015, 2017.

[19] Eric Wieser. Multiple-inheritance hazards in dependently-typed algebraic hierarchies. In *Intelligent Computer Mathematics (CICM 2023)*, volume 14101 of *Lecture Notes in Computer Science*, pages 222–236. Springer, 2023.