Demystifying Proof Assistants

An Introduction to Interactive Theorem Proving

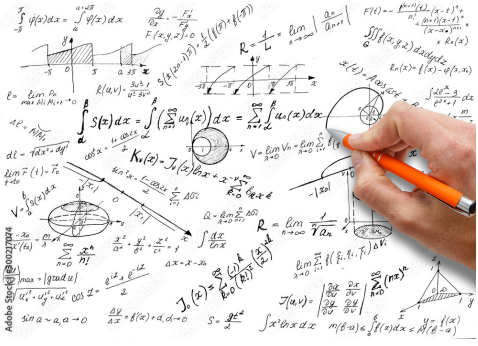
Fabian Grubmüller

Stockholm University / KTH Royal Institute of Technology

April 23, 2024

Outline

Motivation




FROM THE MAKERS OF WOLFRAM LANGUAGE AND MATHEMATICA
WolframAlpha
Enter what you want to calculate or know about
NATURAL LANGUAGE
MATH INPUT
EXTENDED KEYBOARD
EXAMPLES
UPLOAD
RANDOM

Level 1

Level 1

Thm: In a monoid M, left inverses are equal to right inverses.
Proof: Let $x, l, r \in M$ with $l \cdot x = 1 = x \cdot r$. Then

$$l = l \cdot 1 = l \cdot (x \cdot r) = (l \cdot x) \cdot r = 1 \cdot r = r$$

This concludes the proof. $\square$

Level 1

Thm: In a monoid M, left inverses are equal to right inverses.
Proof: Let $x, l, r \in M$ with $l \cdot x = 1 = x \cdot r$. Then

$$l = l \cdot 1 = l \cdot (x \cdot r) = (l \cdot x) \cdot r = 1 \cdot r = r$$

This concludes the proof. □

Level 2

Level 2

Thm: All sheep have the same colour.

Level 2

Thm: All sheep have the same colour.
Proof: By induction of the statement $A(n) \Leftrightarrow$ in a collection of n sheep, all have the same colour.

Level 2

Thm: All sheep have the same colour.
Proof: By induction of the statement $A(n) \Leftrightarrow$ in a collection of n sheep, all have the same colour.
Base: $n = 1$ trivial.

Level 2

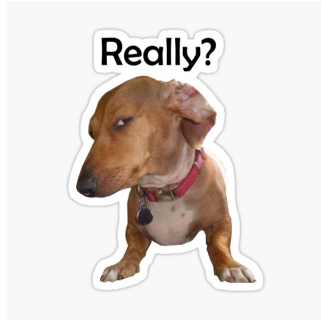
Thm: All sheep have the same colour.
Proof: By induction of the statement $A(n) \Leftrightarrow$ in a collection of n sheep, all have the same colour.
Base: $n = 1$ trivial.
Step: Assume $A(n)$ holds for some $n \in \mathbb{N}$. Take a collection of $n+1$ sheep and order them as $s_0, \dots, s_n$. Consider the collections of sheep $C_1 := \{s_0, \dots, s_{n-1}\}$ and $C_2 := \{s_1, \dots, s_n\}$. C_1 and C_2 are collections of n sheep each, so they are unicoloured by IH. $s_1, \dots, s_{n-1}$, it holds that s_0, s_n as well as all of $s_1, \dots, s_{n-1}$ have the same colour.

Level 2

Thm: All sheep have the same colour.
Proof: By induction of the statement $A(n) \Leftrightarrow$ in a collection of n sheep, all have the same colour.
Base: $n = 1$ trivial.
Step: Assume $A(n)$ holds for some $n \in \mathbb{N}$. Take a collection of $n+1$ sheep and order them as $s_0, \dots, s_n$. Consider the collections of sheep $C_1 := \{s_0, \dots, s_{n-1}\}$ and $C_2 := \{s_1, \dots, s_n\}$. C_1 and C_2 are collections of n sheep each, so they are unicoloured by IH. $s_1, \dots, s_{n-1}$, it holds that s_0, s_n as well as all of $s_1, \dots, s_{n-1}$ have the same colour.
The statement follows by induction. $\square$

Level 2

Thm: All sheep have the same colour.
Proof: By induction of the statement $A(n) \Leftrightarrow$ in a collection of n sheep, all have the same colour.
Base: $n = 1$ trivial.
Step: Assume $A(n)$ holds for some $n \in \mathbb{N}$. Take a collection of $n+1$ sheep and order them as $s_0, \dots, s_n$. Consider the collections of sheep $C_1 := \{s_0, \dots, s_{n-1}\}$ and $C_2 := \{s_1, \dots, s_n\}$. C_1 and C_2 are collections of n sheep each, so they are unicoloured by IH. $s_1, \dots, s_{n-1}$, it holds that s_0, s_n as well as all of $s_1, \dots, s_{n-1}$ have the same colour.
The statement follows by induction. $\square$



Level ∞

Level ∞

INTER-UNIVERSAL TEICHMÜLLER THEORY I: CONSTRUCTION OF HODGE THEATERS

Shinichi Mochizuki

May 2020

Abstract. The present paper is the first in a series of four papers, the goal of which is to establish an *arithmetic* version of *Teichmüller theory* for **number fields** equipped with an **elliptic curve** — which we refer to as **"inter-universal Teichmüller theory"** — by applying the theory of *semi-graphs of anabelioids*, *Frobenioids*, the *étale theta function*, and *log-shells* developed in earlier papers by the author. We begin by fixing what we call *"initial* Θ*-data"*, which consists of an *elliptic curve* E_F over a *number field* F, and a *prime number* $l \geq 5$, as well as some other technical data satisfying certain technical properties. This data determines various *hyperbolic orbicurves* that are related via finite étale coverings to the once-punctured elliptic curve X_F determined by E_F. These finite étale coverings admit various *symmetry properties* arising from the **additive** and **multiplicative** structures on the ring $\mathbb{F}_l = \mathbb{Z}/l\mathbb{Z}$ acting on the *l-torsion points* of the elliptic curve. We then construct *"*$\Theta^{\pm\mathrm{ell}}$*NF-Hodge theaters"* associated to the given Θ-data. These $\Theta^{\pm\mathrm{ell}}$NF-Hodge theaters may be thought of as *miniature models of* **conventional scheme theory** in which the **two underlying combinatorial dimensions** of a number field — which may be thought of as corresponding to the **additive** and **multiplicative** structures of a ring or, alternatively, to the **group of units** and **value group** of a local field associated to the number field — are, in some sense, **"dismantled"** or **"disentangled"** from one another. All $\Theta^{\pm\mathrm{ell}}$NF-Hodge theaters are isomorphic to one another, but may also be related to one another by means of a "Θ-link", which relates certain *Frobenioid-theoretic* portions of one $\Theta^{\pm\mathrm{ell}}$NF-Hodge

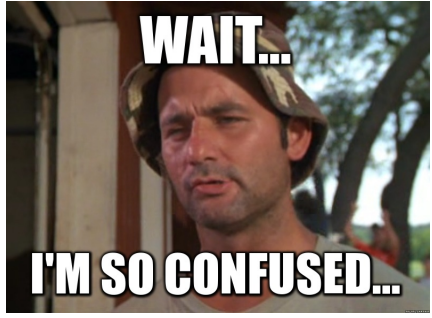
Level ∞

INTER-UNIVERSAL TEICHMÜLLER THEORY I: CONSTRUCTION OF HODGE THEATERS

Shinichi Mochizuki

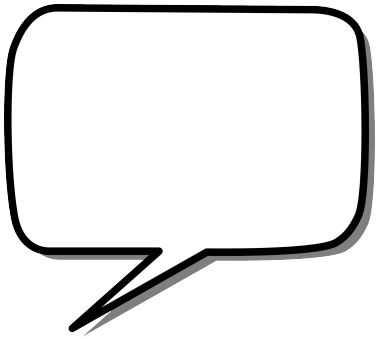
May 2020

Abstract. The present paper is the first in a series of four papers, the goal of which is to establish an *arithmetic* version of *Teichmüller theory* for **number fields** equipped with an **elliptic curve** — which we refer to as **"inter-universal Teichmüller theory"** — by applying the theory of *semi-graphs of anabelioids, Frobenioids*, the *étale theta function*, and *log-shells* developed in earlier papers by the author. We begin by fixing what we call *"initial Θ-data"*, which consists of an *elliptic curve* E_F over a *number field* F, and a *prime number* $l \geq 5$, as well as some other technical data satisfying certain technical properties. This data determines various *hyperbolic orbicurves* that are related via finite étale coverings to the once-punctured elliptic curve X_F determined by E_F. These finite étale coverings admit various *symmetry properties* arising from the **additive** and **multiplicative** structures on the ring $\mathbb{F}_l = \mathbb{Z}/l\mathbb{Z}$ acting on the *l-torsion points* of the elliptic curve. We then construct *"$\Theta^{\pm\mathrm{ell}}$NF-Hodge theaters"* associated to the given Θ-data. These $\Theta^{\pm\mathrm{ell}}$NF-Hodge theaters may be thought of as *miniature models of* **conventional scheme theory** in which the **two underlying combinatorial dimensions** of a number field — which may be thought of as corresponding to the **additive** and **multiplicative** structures of a ring or, alternatively, to the **group of units** and **value group** of a local field associated to the number field — are, in some sense, **"dismantled"** or **"disentangled"** from one another. All $\Theta^{\pm\mathrm{ell}}$NF-Hodge theaters are isomorphic to one another, but may also be related to one another by means of a **"Θ-link"**, which relates certain *Frobenioid-theoretic* portions of one $\Theta^{\pm\mathrm{ell}}$NF-Hodge



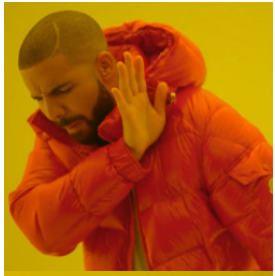
Reality Check

Reality Check

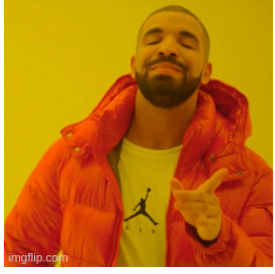


Trust Issues?!

Trust Issues?!



Some random mathematician on the other side of the world



Automatic Theorem Prover

Benefits

Process

Process

- REPL or LSP

Process

- REPL or LSP
- highlighting, type checking

Process

- REPL or LSP
- highlighting, type checking
- tactics vs term-style

Process

- REPL or LSP
- highlighting, type checking
- tactics vs term-style
- (optional) automatic proof insertion

Process

- REPL or LSP
- highlighting, type checking
- tactics vs term-style
- (optional) automatic proof insertion
- (optional) program compilation

Process

- REPL or LSP
- highlighting, type checking
- tactics vs term-style
- (optional) automatic proof insertion
- (optional) program compilation
- Under the hood: (higher order) unification, term rewriting

Crash Course Type Theory

Keywords:

- Propositions as types

(What are you looking at? Please focus on the talk :P)

Crash Course Type Theory

Keywords:

- Propositions as types
- Higher order types

(What are you looking at? Please focus on the talk :P)

Crash Course Type Theory

Keywords:

- Propositions as types
- Higher order types
- λ calculus

(What are you looking at? Please focus on the talk :P)

Crash Course Type Theory

Keywords:

- Propositions as types
- Higher order types
- λ calculus
- Curry-Howard isomorphism

(What are you looking at? Please focus on the talk :P)

Crash Course Type Theory

Keywords:

- Propositions as types
- Higher order types
- λ calculus
- Curry-Howard isomorphism
- Proof (ir-)relevance, ex-/intensionality

(What are you looking at? Please focus on the talk :P)

Crash Course Type Theory

Keywords:

- Propositions as types
- Higher order types
- λ calculus
- Curry-Howard isomorphism
- Proof (ir-)relevance, ex-/intensionality
- Currying (higher order functions)

(What are you looking at? Please focus on the talk :P)

Popular Tools (Overview)

- Prolog
- Coq
- Agda
- Isabelle
- Lean

Prolog (1972)

- one of the oldest logic programming languages
- uses Horn clauses
- widely used in computer science, business and AI

```
mortal(X) :- human(X).

human(socrates).

mortal(X) :- human(X).

?- mortal(socrates).
% Yes

?- mortal(X).
% X = socrates
```

Coq (Calculus of Constructions) (1989)

Coq (Calculus of Constructions) (1989)

- Uses *Calculus of Constructions* (CoC)

Coq (Calculus of Constructions) (1989)

- Uses *Calculus of Constructions* (CoC)
- Mature and big ecosystem

Coq (Calculus of Constructions) (1989)

- Uses *Calculus of Constructions* (CoC)
- Mature and big ecosystem
- archaic syntax (e. g. no Unicode support)

Coq (Calculus of Constructions) (1989)

- Uses *Calculus of Constructions* (CoC)
- Mature and big ecosystem
- archaic syntax (e. g. no Unicode support)

```
plus_comm =
fun n m : nat =>
nat_ind (fun n0 : nat => n0 + m = m + n0)
  (plus_n_0 m)
  (fun (y : nat) (H : y + m = m + y) =>
   eq_ind (S (m + y))
     (fun n0 : nat => S (y + m) = n0)
     (f_equal S H)
     (m + S y)
     (plus_n_Sm m y)) n
     : forall n m : nat, n + m = m + n
```

Agda (v2) (2007)

- essentially an extension of Haskell

Agda (v2) (2007)

- essentially an extension of Haskell
- based on Unified Theory of Dependent Types (~MLTT)

Agda (v2) (2007)

- essentially an extension of Haskell
- based on Unified Theory of Dependent Types (~MLTT)
- extensions for *cubical type theory* (HoTT)

Agda (v2) (2007)

- essentially an extension of Haskell
- based on Unified Theory of Dependent Types (~MLTT)
- extensions for *cubical type theory* (HoTT)
- no built-in tactics language (dependent types)

Agda (v2) (2007)

- essentially an extension of Haskell
- based on Unified Theory of Dependent Types (~MLTT)
- extensions for *cubical type theory* (HoTT)
- no built-in tactics language (dependent types)
- extensive use of Unicode characters

Agda vs Haskell

Agda:

```
_+_ : ℕ→ℕ→ℕ
m +  zero    = m
m + (succ n) = succ (m + n)
```

Haskell:

```
(+) :: Nat -> Nat -> Nat
m +  zero    = m
m + (succ n) = succ (m + n)
```

Isabelle (1986)

Isabelle (1986)

- small kernel ($\rightarrow$ more trustworthiness)

Isabelle (1986)

- small kernel ($\rightarrow$ more trustworthiness)
- many extensions

Isabelle (1986)

- small kernel ($\rightarrow$ more trustworthiness)
- many extensions
- separation of definition and tactics language

Isabelle (1986)

- small kernel ($\rightarrow$ more trustworthiness)
- many extensions
- separation of definition and tactics language
- *Archive of Formal Proofs*

Isabelle

```
theorem sqrt2_not_rational:
  "sqrt 2 ∉ℚ "
proof
  let ?x = "sqrt 2"
  assume "?x ∈ℚ "
  then obtain m n :: nat where
    sqrt_rat: "¦?x¦ = m / n" and lowest_terms: "coprime m n"
    by (rule Rats_abs_nat_div_natE)
  hence "m^2 = ?x^2 * n^2" by (auto simp add: power2_eq_square)
  hence eq: "m^2 = 2 * n^2" using of_nat_eq_iff power2_eq_square by fastforce
  hence "2 dvd m^2" by simp
  hence "2 dvd m" by simp
  have "2 dvd n" proof -
    from ‹2 dvd ›m obtain k where "m = 2 * k" ..
    with eq have "2 * n^2 = 2^2 * k^2" by simp
    hence "2 dvd n^2" by simp
    thus "2 dvd n" by simp
  qed
  with ‹2 dvd ›m have "2 dvd gcd m n" by (rule gcd_greatest)
  with lowest_terms have "2 dvd 1" by simp
  thus False using odd_one by blast
qed
```

Lean (v4) (2013)

Lean (v4) (2013)

- based on Calculus of Constructions (same as Coq)

Lean (v4) (2013)

- based on Calculus of Constructions (same as Coq)
- modern syntax (Unicode support)

Lean (v4) (2013)

- based on Calculus of Constructions (same as Coq)
- modern syntax (Unicode support)
- Very extensive library Mathlib

Lean (v4) (2013)

- based on Calculus of Constructions (same as Coq)
- modern syntax (Unicode support)
- Very extensive library Mathlib
- A lot of recent high-impact development

Lean (v4) (2013)

- based on Calculus of Constructions (same as Coq)
- modern syntax (Unicode support)
- Very extensive library Mathlib
- A lot of recent high-impact development
- Functional style; macros for tactics

Lean (v4)

```
theorem and_swap (p q : Prop) : p ∧ q → q ∧ p := by
  intro h
  apply And.intro
  · exact h.right
  · exact h.left
```

Practical part with Lean 4 (feel free to join in yourself!)

Lean online editor

Files