Revisiting Hofmann's setoid model

Loïc Pujet

1 july 2026

I. Background

Axioms in type theory

Type theory is an "axiom-free" foundation for mathematics

Axioms in type theory

Type theory is an "axiom-free" foundation for mathematics

(the job of axioms is offloaded to derivation rules)

Axioms in type theory

Type theory is an "axiom-free" foundation for mathematics

(the job of axioms is offloaded to derivation rules)

$$\frac{}{\mathbb{N} : \mathit{Type}} \qquad \frac{}{0 : \mathbb{N}} \qquad \frac{n : \mathbb{N}}{S\,n : \mathbb{N}}$$

$$\frac{P : \mathbb{N} \to \mathit{Type} \qquad p_z : P\,0 \qquad p_s : \Pi\,(n : \mathbb{N}).\,P\,n \to P\,(S\,n)}{\mathit{natrec}(P, p_z, p_s) : \Pi\,(n : \mathbb{N}).\,P\,n}$$

The Hurry Coward correspondance

These derivation rules interact via the conversion relation

See also: cut elimination, program evaluation

$$
\begin{aligned}
(\lambda\, x\,.\, f)\; t \quad &\equiv \quad f[x :\equiv t] \\
\mathit{natrec}(P, p_s, p_z)\; 3 \quad &\equiv \quad p_z\; (p_z\; (p_z\; p_s))
\end{aligned}
$$

The Hurry Coward correspondance

These derivation rules interact via the conversion relation

See also: cut elimination, program evaluation

$$(\lambda\, x\,.\, f)\; t \quad \equiv \quad f[x :\equiv t]$$
$$natrec(P, p_s, p_z)\; 3 \quad \equiv \quad p_z\; (p_z\; (p_z\; p_s))$$

Every judgment in type theory is stable under conversion

$$\_ : Vector\; \mathbb{N}\; (2 + 2)$$

The Hurry Coward correspondance

These derivation rules interact via the conversion relation

See also: cut elimination, program evaluation

$$\begin{aligned} (\lambda\, x\,.\, f)\; t &\quad\equiv\quad f[x :\equiv t] \\ \mathit{natrec}(P, p_s, p_z)\; 3 &\quad\equiv\quad p_z\; (p_z\; (p_z\; p_s)) \end{aligned}$$

Every judgment in type theory is stable under conversion

$$\_ : \mathit{Vector}\; \mathbb{N}\; 4$$

The Hurry Coward correspondance

These derivation rules interact via the conversion relation

See also: cut elimination, program evaluation

$$(\lambda\, x\,.\, f)\, t \quad \equiv \quad f[x :\equiv t]$$
$$\mathit{natrec}(P, p_s, p_z)\, 3 \quad \equiv \quad p_z\, (p_z\, (p_z\, p_s))$$

Every judgment in type theory is stable under conversion

$$[0, 1, 2, 3] : \mathit{Vector}\ \mathbb{N}\ 4$$

The Hurry Coward correspondance

These derivation rules interact via the conversion relation

See also: cut elimination, program evaluation

$$\begin{aligned} (\lambda\, x\,.\, f)\, t &\equiv f[x :\equiv t] \\ natrec(P, p_s, p_z)\, 3 &\equiv p_z\, (p_z\, (p_z\, p_s)) \end{aligned}$$

Every judgment in type theory is stable under conversion

$$[0, 1, 2, 3] : Vector\ \mathbb{N}\ 4$$

→ conversion should be decidable

Some terms are more equal than others

Conversion is decidable and automatically handled by software

→ we need an other equality relation to use in our proofs

Some terms are more equal than others

Conversion is decidable and automatically handled by software

→ we need an other equality relation to use in our proofs

Propositional equality

$x =_A y$ is defined inductively as the smallest reflexive relation

Some terms are more equal than others

Conversion is decidable and automatically handled by software

$\rightarrow$ we need an other equality relation to use in our proofs

Propositional equality

$x =_A y$ is defined inductively as the smallest reflexive relation

Now it becomes possible to state and prove e.g.

- $\Pi\,(n : \mathbb{N})\,(m : \mathbb{N}).\, n + m =_{\mathbb{N}} m + n$
- $\Pi\,(n : \mathbb{N}).\,(n = 0) + (\Sigma\,(m : \mathbb{N}).\, n =_{\mathbb{N}} S\ m)$

The joy of intensionality

This propositional equality is **under-determined** at higher types

→ the following are **not** provable:

The joy of intensionality

This propositional equality is **under-determined** at higher types

→ the following are **not** provable:

Function extensionality (funext)

$f =_{A \to B} g$ if and only if $\Pi\,(x : A).\, f\ x =_B g\ x$

The joy of intensionality

This propositional equality is **under-determined** at higher types

→ the following are **not** provable:

Function extensionality (funext)

$f =_{A \to B} g$ if and only if $\Pi\,(x : A).\, f\ x =_B g\ x$

Proposition extensionality (propext)

$P =_{Prop} Q$ if and only if $P \leftrightarrow Q$

The joy of intensionality

This propositional equality is **under-determined** at higher types

→ the following are **not** provable:

Function extensionality (funext)

$f =_{A \to B} g$ if and only if $\Pi\,(x : A).\, f\ x =_B g\ x$

Proposition extensionality (propext)

$P =_{Prop} Q$ if and only if $P \leftrightarrow Q$

Uniqueness of Identity Proofs (UIP)

$p =_{(x=y)} q$ is always true

The joy of intensionality

This propositional equality is **under-determined** at higher types

→ the following are **not** provable:

Function extensionality (funext)

$f =_{A \to B} g$ if and only if $\Pi\,(x : A).\, f\, x =_B g\, x$

Proposition extensionality (propext)

$P =_{Prop} Q$ if and only if $P \leftrightarrow Q$

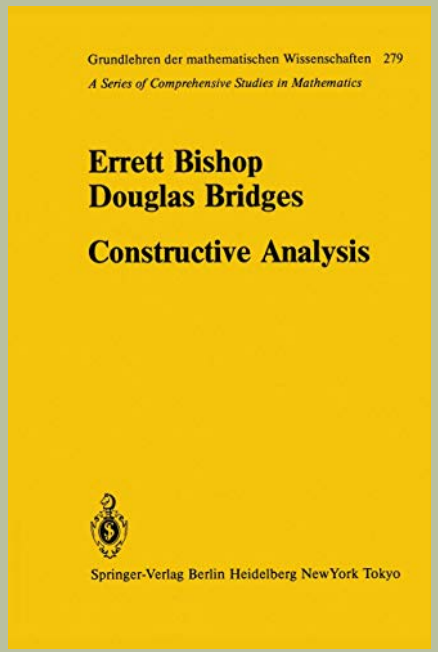
Uniqueness of Identity Proofs (UIP)

$p =_{(x=y)} q$ is always true

MLTT does not support **quotient types** either

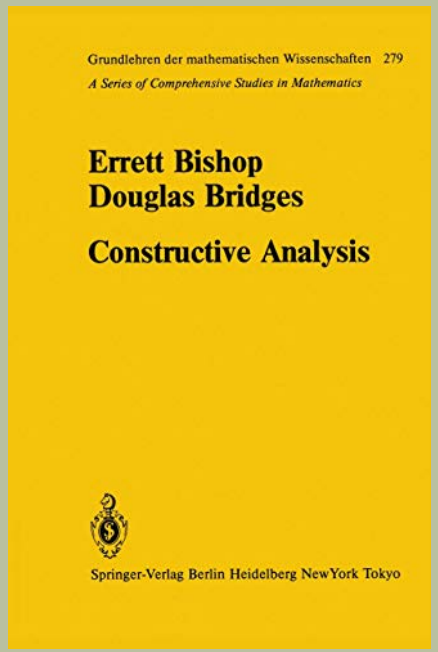
It is not a bug, it is a feature

Per Martin-Löf intended his type theory as an intensional and predicative foundation for Bishop-style constructive mathematics



It is not a bug, it is a feature

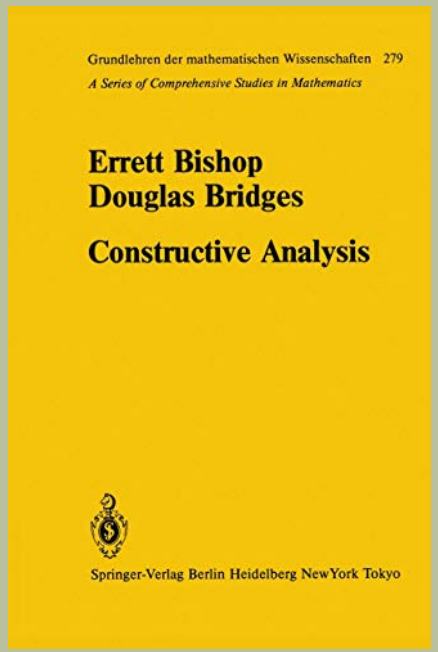
Per Martin-Löf intended his type theory as an intensional and predicative foundation for Bishop-style constructive mathematics



Bishop does not work directly with types, but with setoids

It is not a bug, it is a feature

Per Martin-Löf intended his type theory as an intensional and predicative foundation for Bishop-style constructive mathematics

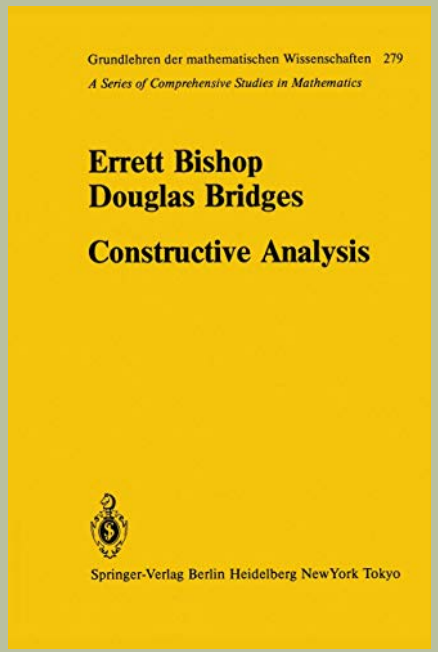


Bishop does not work directly with types, but with setoids

A setoid is a type equipped with an equivalence relation

It is not a bug, it is a feature

Per Martin-Löf intended his type theory as an intensional and predicative foundation for Bishop-style constructive mathematics



Bishop does not work directly with types, but with setoids

A setoid is a type equipped with an equivalence relation

A setoid morphism is a function which preserves the equivalence relation

Setoid Hell

In practice though, setoids are a bit clunky:

- Lots of boilerplate
  (slightly mitigated by typeclasses and tactics, but still!)
- No type dependency

Setoid Hell

In practice though, setoids are a bit clunky:

- Lots of boilerplate
  (slightly mitigated by typeclasses and tactics, but still!)
- No type dependency

In practice, most users of proof assistants prefer to postulate extensionality axioms and work with dependent types

Setoid Hell

In practice though, setoids are a bit clunky:

- Lots of boilerplate
  (slightly mitigated by typeclasses and tactics, but still!)
- No type dependency

In practice, most users of proof assistants prefer to postulate extensionality axioms and work with dependent types

...so much for axiom-free mathematics!

II. Setoid models

Compilation 101

In his PhD thesis, Martin Hofmann envisioned a compiler for these extensionality axioms

MLTT + extensional axioms $\longrightarrow$ MLTT

Compilation 101

In his PhD thesis, Martin Hofmann envisioned a compiler for these extensionality axioms

MLTT + extensional axioms $\longrightarrow$ MLTT

He developed a notion of dependent setoids and used them to build a model of MLTT inside MLTT

Compilation 101

In his PhD thesis, Martin Hofmann envisioned a compiler for these extensionality axioms

MLTT + extensional axioms $\longrightarrow$ MLTT

He developed a notion of dependent setoids and used them to build a model of MLTT inside MLTT

Contexts $\rightarrow$ Setoids
Substitutions $\rightarrow$ Setoid morphisms
Types $\rightarrow$ Dependent setoids
Terms $\rightarrow$ Sections of dependent setoids

Compilation 101

In his PhD thesis, Martin Hofmann envisioned a compiler for these extensionality axioms

MLTT + extensional axioms $\longrightarrow$ MLTT

He developed a notion of dependent setoids and used them to build a model of MLTT inside MLTT

$$\Gamma \vdash t : A \quad \longrightarrow \quad [\![\Gamma]\!] \vdash [\![t]\!] : [\![A]\!]$$

Compilation 101

In his PhD thesis, Martin Hofmann envisioned a compiler for these extensionality axioms

MLTT + extensional axioms $\longrightarrow$ MLTT

He developed a notion of dependent setoids and used them to build a model of MLTT inside MLTT

$\Gamma \vdash t : A \quad \longrightarrow \quad [\![\Gamma]\!] \vdash [\![t]\!] : [\![A]\!]$

$[\![funext]\!] \quad \equiv \quad$ (* some proof-term in MLTT *)

Fine print

Hofmann's construction comes short of being a model of MLTT

Fine print

Hofmann's construction comes short of being a model of MLTT

- No universe of types
  Without these, no way to even prove $0 \neq 1$

Fine print

Hofmann's construction comes short of being a model of MLTT

- No universe of types
  Without these, no way to even prove $0 \neq 1$
- Some conversion rules do not hold in the model

$$A \equiv B \quad \longrightarrow \quad [\![A]\!] \not\equiv [\![B]\!]$$

Fine print

Hofmann's construction comes short of being a model of MLTT

- No universe of types
  Without these, no way to even prove $0 \neq 1$
- Some conversion rules do not hold in the model

$$A \equiv B \quad \longrightarrow \quad [\![A]\!] \not\equiv [\![B]\!]$$

Hofmann's model only handles a weak version of type theory

The source of the problem

The category of setoids is not cartesian closed

The source of the problem

The category of setoids is not cartesian closed

$Hom(A \times B, C) \dashv Hom(A, B \to C)$ only holds up to setoid equality

The source of the problem

The category of setoids is not cartesian closed

$Hom(A \times B, C) \dashv Hom(A, B \to C)$ only holds up to setoid equality

Setoid should really be understood as a *Setoid*-enriched category!
Unfortunately, the rules of type theory are strict

A non-solution

Category theorists will define the ex/reg completion by quotienting the function spaces $A \to B$ by setoid equality

A non-solution

Category theorists will define the ex/reg completion by quotienting the function spaces $A \to B$ by setoid equality

Constructively, this construction behaves very poorly

A better non-solution

Altenkirch (1999) identifies a subcategory of *Setoid* which is sufficiently strict to build a model of MLTT

A better non-solution

Altenkirch (1999) identifies a subcategory of *Setoid* which is sufficiently strict to build a model of MLTT

Extend MLTT with a universe of strict propositions

$$\frac{A : SProp \qquad x : A \qquad y : A}{x \equiv y}$$

Then define a strict setoid as a type with a *SProp*-valued relation

A better non-solution

Altenkirch (1999) identifies a subcategory of *Setoid* which is sufficiently strict to build a model of MLTT

Extend MLTT with a universe of strict propositions

$$\frac{A : \mathit{SProp} \qquad x : A \qquad y : A}{x \equiv y}$$

Then define a strict setoid as a type with a *SProp*-valued relation

We can now interpret types as (dependent) strict setoids

MLTT + extensionality − universes $\longrightarrow$ MLTT + SProp

A better non-solution

A few years later, Altenkirch et al. add a universe to their model

Altenkirch, Boulier, Kaposi, Sattler, Sestini 2021

A better non-solution

A few years later, Altenkirch et al. add a universe to their model

Altenkirch, Boulier, Kaposi, Sattler, Sestini 2021

To do so, they need their propositional equality to live in SProp
Let's call this requirement strict equality

$$\underbrace{\text{MLTT + extensionality}}_{\text{What we wanted}} \longrightarrow \underbrace{\text{MLTT + SProp + strict eq}}_{\text{"Reasonable" extension}}$$

Everything comes at a cost

Are we happy now?

Everything comes at a cost

Are we happy now? Not really!

Everything comes at a cost

Are we happy now? Not really!

Unlike setoids, strict setoids do not support effective quotients

Everything comes at a cost

Are we happy now? Not really!

Unlike setoids, strict setoids do not support effective quotients

Let A be a type, and $R : A \to A \to Type$ an equivalence relation
Forming the quotient A/R destroys the computational content of R

$\Rightarrow$ from $\pi(x) =_{A/R} \pi(y)$, we can never recover $R\ x\ y$

III. Revisiting Hofmann's model

Back to the basics

Let's go back to regular setoids

Back to the basics

Let's go back to regular setoids

Hofmann noticed that the reason *Setoid* is not cartesian closed boils down to morphisms not preserving reflexivity

Back to the basics

Let's go back to regular setoids

Hofmann noticed that the reason *Setoid* is not cartesian closed boils down to morphisms not preserving reflexivity

Define a category *ReflSetoid* whose morphisms preserve reflexivity

Extensionally, *ReflSetoid* does provide a model of MLTT, but

- some equations are propositional instead of definitional
- effectivity of quotients is non-constructive

More categorical jargon

Taking the POV of cubical type theories, *Setoid* and *ReflSetoid* can be understood as categories of fibrant presheaves

More categorical jargon

Taking the POV of cubical type theories, *Setoid* and *ReflSetoid* can be understood as categories of fibrant presheaves

Setoid = reflexive fibrant presheaves on the truncated semi-cube category

ReflSetoid = fibrant presheaves on the truncated cube category

Perverse sheaves, but even worse

The obstacles in *Setoid* and *ReflSetoid* are of a different nature:

Perverse sheaves, but even worse

The obstacles in *Setoid* and *ReflSetoid* are of a different nature:

- *Setoid* is not cartesian closed
  exponentials of reflexive presheaves are not reflexive

Perverse sheaves, but even worse

The obstacles in *Setoid* and *ReflSetoid* are of a different nature:

- *Setoid* is not cartesian closed
  exponentials of reflexive presheaves are not reflexive
- *ReflSetoid* is cartesian closed
  however the naturality equations are not sufficiently strict

Perverse sheaves, but even worse

The obstacles in *Setoid* and *ReflSetoid* are of a different nature:

- *Setoid* is not cartesian closed
  exponentials of reflexive presheaves are not reflexive
- *ReflSetoid* is cartesian closed
  however the naturality equations are not sufficiently strict

Pédrot (2020) found an alternative presentation of presheaf models which features strict naturality equations

Putting everything together

We must now incorporate fibrancy to Pédrot's construction

Putting everything together

We must now incorporate fibrancy to Pédrot's construction

- Define an inductive-recursive universe of fibrant presheaves

Putting everything together

We must now incorporate fibrancy to Pédrot's construction

- Define an inductive-recursive universe of fibrant presheaves
- Define the equality and type coercion on the universe

Putting everything together

We must now incorporate fibrancy to Pédrot's construction

- Define an inductive-recursive universe of fibrant presheaves
- Define the equality and type coercion on the universe
- This turns the universe into a fibrant presheaf

Putting everything together

We must now incorporate fibrancy to Pédrot's construction

- Define an inductive-recursive universe of fibrant presheaves
- Define the equality and type coercion on the universe
- This turns the universe into a fibrant presheaf

Extra step: eliminate induction-recursion from the construction

What do we gain?

We got back to the result of Altenkirch et al.

$$\text{MLTT + extensionality} \longrightarrow \text{MLTT + SProp + strict eq}$$

What do we gain?

We got back to the result of Altenkirch et al.

$$\text{MLTT + extensionality} \longrightarrow \text{MLTT + SProp + strict eq}$$

Additionally, we can recover some computational content from equalities using e.g. the accessibility predicate

What about effective quotients?

What about effective quotients?

Effectivity of quotients amounts to the following principle:

- Let A be a setoid whose elements are all equal (hProp)
- Let $\|A\|$ be the setoid whose equality relation is replaced by the trivial one
- Then there is a setoid morphism $\|A\| \to A$

What about effective quotients?

Effectivity of quotients amounts to the following principle:

- Let A be a setoid whose elements are all equal (hProp)
- Let $\|A\|$ be the setoid whose equality relation is replaced by the trivial one
- Then there is a setoid morphism $\|A\| \to A$

There is an obvious solution in $Setoid$: the identity function sends equal terms of $\|A\|$ to equal terms of A

What about effective quotients?

Effectivity of quotients amounts to the following principle:

- Let A be a setoid whose elements are all equal (hProp)
- Let $\|A\|$ be the setoid whose equality relation is replaced by the trivial one
- Then there is a setoid morphism $\|A\| \to A$

There is an obvious solution in $Setoid$: the identity function sends equal terms of $\|A\|$ to equal terms of A

...but it does not preserve reflexivity

What about effective quotients?

If we assume excluded middle for equalities (ELEM)

$$\Pi\,(x : A)(y : A).(x =_A y) + (x =_A y \to \bot)$$

we can solve this problem in *ReflSetoid*.

What about effective quotients?

If we assume excluded middle for equalities (ELEM)

$$\Pi\,(x : A)(y : A).(x =_A y) + (x =_A y \to \bot)$$

we can solve this problem in $RaSetoid$.

Define a morphism from $\|A\| \to A$ as follows:

- take the identity on the underlying type

What about effective quotients?

If we assume excluded middle for equalities (ELEM)

$$\Pi\,(x : A)(y : A).(x =_A y) + (x =_A y \to \bot)$$

we can solve this problem in $RflSetoid$.

Define a morphism from $\|A\| \to A$ as follows:

- take the identity on the underlying type
- Let $x\ y : \|A\|$, we must show that they are equal in A.
  If $x = y$, then they are equal by reflexivity.
  If $x \neq y$, then use the fact that all elements in A are equal.

What about effective quotients?

If we assume excluded middle for equalities (ELEM)

$$\Pi\,(x : A)(y : A).(x =_A y) + (x =_A y \to \bot)$$

we can solve this problem in *ReflSetoid*.

Define a morphism from $\|A\| \to A$ as follows:

- take the identity on the underlying type
- Let $x\ y : \|A\|$, we must show that they are equal in A.
  If $x = y$, then they are equal by reflexivity.
  If $x \neq y$, then use the fact that all elements in A are equal.
- Reflexivity is preserved!

What about effective quotients?

Well sure. But ELEM is obviously non-constructive!

What about effective quotients?

Well sure. But ELEM is obviously non-constructive! ...or is it?

What about effective quotients?

Well sure. But ELEM is obviously non-constructive! ...or is it?

Claim
We can give a presentation of MLTT + SProp + strict eq + ELEM which is strongly normalising and has canonicity

Computing with ELEM

Let's enrich type theory with an operator which decides equality

$$eq : (A : Type)\,(x\ y : A) \to (x =_A y) + (x =_A y \to \bot)$$

Computing with ELEM

Let's enrich type theory with an operator which decides equality

$$eq : (A : Type)\,(x\ y : A) \to (x =_A y) + (x =_A y \to \bot)$$

Note that this is not a propositional disjunction. This operator really decides equality between x and y

Computing with ELEM

Let's enrich type theory with an operator which decides equality

$$eq : (A : Type)\,(x\ y : A) \to (x =_A y) + (x =_A y \to \bot)$$

Note that this is not a propositional disjunction. This operator really decides equality between x and y

$\Rightarrow$ This operator is strongly intensional. In presence of function extensionality, equality is not decidable at higher types!

Computing with ELEM

So how does *eq* compute?

Computing with ELEM

So how does *eq* compute? It only computes on closed terms

Computing with ELEM

So how does *eq* compute? It only computes on closed terms

$eq\ A\ x\ y \rightsquigarrow left$ if x and y are equal closed terms
$eq\ A\ x\ y \rightsquigarrow right$ if x and y are distinct closed terms
$eq\ A\ x\ y$ is neutral otherwise

Final Result

MLTT + ext. + eff. quotients $\longrightarrow$ MLTT + SProp + strict eq + ELEM

Question:
Can we turn this model into a reasonable type theory?

Thank you!