# An Inductive Universe for Setoids

Loïc Pujet

13 june 2025

# The Setoid Model

Hofmann's PhD thesis: two translations from CC to CC

$$\Gamma \vdash t : A \quad \rightsquigarrow \quad [\![\Gamma]\!] \vdash [\![t]\!] : [\![A]\!]$$
$$\Gamma \vdash t \equiv u : A \quad \rightsquigarrow \quad [\![\Gamma]\!] \vdash [\![t]\!] \equiv [\![u]\!] : [\![A]\!]$$

They validate

- function extensionality
- proposition extensionality
- quotient types

# The Setoid Model

Hofmann's PhD thesis: two translations from CC to CC

$$\Gamma \vdash t : A \quad \rightsquigarrow \quad [\![\Gamma]\!] \vdash [\![t]\!] : [\![A]\!]$$
$$\Gamma \vdash t \equiv u : A \quad \rightsquigarrow \quad [\![\Gamma]\!] \vdash [\![t]\!] \equiv [\![u]\!] : [\![A]\!]$$

They validate

- function extensionality
- proposition extensionality
- quotient types

BUT:

- First model: no true dependent types
- Second model: missing definitional equations

# The Setoid Model, again

Altenkirch '99 adds definitional proof irrelevance

$$
\begin{aligned}
&\mathsf{Setoid} :\equiv \{ \\
&\quad A && : \mathsf{Type} \\
&\quad \_ \sim_A \_ && : A \to A \to \mathsf{SProp} \\
&\quad \mathsf{refl} && : x \sim_A x \\
&\quad \mathsf{sym} && : x \sim_A y \to y \sim_A x \\
&\quad \mathsf{trans} && : x \sim_A y \to y \sim_A z \to x \sim_A z \\
&\}
\end{aligned}
$$

→ true dependent types

# The Setoid Model, again

Altenkirch '99 adds definitional proof irrelevance

$$
\begin{aligned}
&\mathrm{Setoid} :\equiv \{ \\
&\quad A && : \mathrm{Type} \\
&\quad \_ \sim_A \_ && : A \to A \to \mathrm{SProp} \\
&\quad \mathrm{refl} && : x \sim_A x \\
&\quad \mathrm{sym} && : x \sim_A y \to y \sim_A x \\
&\quad \mathrm{trans} && : x \sim_A y \to y \sim_A z \to x \sim_A z \\
&\}
\end{aligned}
$$

→ true dependent types

+ Universe of non-dependent types

# An Inductive-Recursive Universe

$$
\begin{aligned}
&\text{Inductive } U :\equiv \\
&\quad |\ N \ : U \\
&\quad |\ \Pi \ : (A : U)\,(P : \mathsf{El}\,A \to U)\,(P_e : a \ {}_{A}{\sim}_{A}\ a' \to P\,a \sim_U P\,a') \to U
\end{aligned}
$$

$$
\begin{aligned}
\mathsf{El}\,N &\equiv \mathbb{N} \\
\mathsf{El}\,(\Pi\,A\,P\,P_e) &\equiv (f : (a : \mathsf{El}\,A) \to \mathsf{El}\,(P\,a)) \\
&\quad \times (f_e : a \ {}_{A}{\sim}_{A}\ a' \to f\,a \ {}_{P\,a}{\sim}_{P\,a'}\ f\,a')
\end{aligned}
$$

$$
\begin{aligned}
N \sim_U N &\equiv \top \\
\Pi\,A\,P\,P_e \sim_U \Pi\,B\,Q\,Q_e &\equiv (A \sim_U B) \times (a \ {}_{A}{\sim}_{B}\ b \to P\,a \sim_U Q\,b) \\
\_ \sim_U \_ &\equiv \bot
\end{aligned}
$$

$$
\begin{aligned}
n \ {}_{N}{\sim}_{N}\ m &\equiv \text{(* inductive def of equality *)} \\
\langle f, f_e \rangle \ {}_{\Pi\,A\,P\,P_e}{\sim}_{\Pi\,B\,Q\,Q_e}\ \langle g, g_e \rangle &\equiv a \ {}_{A}{\sim}_{B}\ b \to f\,a \ {}_{P\,a}{\sim}_{Q\,b}\ g\,b \\
x \ {}_{\_}{\sim}_{\_}\ y &\equiv \bot
\end{aligned}
$$

# Is that actually a definition?

# Is that actually a definition?

Agda accepts it ¯_(ツ)_/¯

# Is that actually a definition?

Agda accepts it ¯_(ツ)_/¯

...actually, this is **double** ind-rec. No general theory for those AFAIK

# Is that actually a definition?

Agda accepts it ¯_(ツ)_/¯

...actually, this is double ind-rec. No general theory for those AFAIK

Altenkirch, Boulier, Kaposi, Sattler and Sestini '21: we can do better

# Is that actually a definition?

Agda accepts it ¯_(ツ)_/¯

...actually, this is double ind-rec. No general theory for those AFAIK

Altenkirch, Boulier, Kaposi, Sattler and Sestini '21: we can do better

- encoding as an inductive-inductive family

# Is that actually a definition?

Agda accepts it ¯_(ツ)_/¯

...actually, this is **double** ind-rec. No general theory for those AFAIK

Altenkirch, Boulier, Kaposi, Sattler and Sestini '21: we can do better

- encoding as an inductive-inductive family
- encoding as an inductive family in a theory with a SProp-valued equality with large elimination

# I.

# An Inductive Universe

# Let's just hack our way through it

$$
\begin{aligned}
&\text{Inductive } U :\equiv \\
&\quad|\ N \ : U \\
&\quad|\ \Pi \ : (A : U) \\
&\qquad\quad (P : \mathsf{El}\, A \to U) \\
&\qquad\quad (P_e : a \ {}_{A}{\sim}_{A}\ a' \to P\, a \ \sim_U \ P\, a') \to U
\end{aligned}
$$

$$
\begin{aligned}
&\mathsf{El} : U \to \text{Type} \\
&\mathsf{El}\, N \equiv \mathbb{N} \\
&\mathsf{El}\, (\Pi\, A\, P\, P_e) \equiv (f : (a : \mathsf{El}\, A) \to \mathsf{El}\, (P\, a)) \\
&\qquad\qquad \times (f_e : a \ {}_{A}{\sim}_{A}\ a' \to f\, a \ {}_{P\,a}{\sim}_{P\,a'}\ f\, a')
\end{aligned}
$$

(* Definition of equalities *)

# Let's just hack our way through it

Inductive $U$ :≡
| $N$ : $U$
| $\Pi$ : $(A : U)$
  $(P : \mathrm{El}\, A \to U) \to U$

$$\mathrm{El} : U \to \mathrm{Type}$$
$$\mathrm{El}\, N \equiv \mathbb{N}$$
$$\mathrm{El}\, (\Pi\, A\, P) \equiv (f : (a : \mathrm{El}\, A) \to \mathrm{El}\, (P\, a)) \times (f_e : a \;{}_{A}\!\sim_{A}\; a' \to f\, a \;{}_{P\,a}\!\sim_{P\,a'}\; f\, a')$$

(* Definition of equalities *)

# Let's just hack our way through it

Inductive $U$ :≡

| $N$ : $U$
| $\Pi$ : $(A : U)\ (A_= : A \to A \to \mathrm{SProp})$
  $(P : \mathrm{El}\ A \to U)\ (P_= : P\,a \to P\,a' \to \mathrm{SProp}) \to U$

$$\mathrm{El} : U \to \mathrm{Type}$$
$$\mathrm{El}\ N \equiv \mathbb{N}$$
$$\mathrm{El}\ (\Pi\ A\ A_=\ P\ P_=) \equiv (f : (a : \mathrm{El}\ A) \to \mathrm{El}\ (P\ a)) \times (f_e : A_=\ a\ a' \to P_=\ (f\ a)\ (f\ a'))$$

(* Definition of equalities *)

# Let's just hack our way through it

$$
\begin{aligned}
&\text{Inductive } U :\equiv \\
&\mid N \ : U \\
&\mid \Pi \ : (A : U)\,(A_= : A \to A \to \text{SProp}) \\
&\qquad (P : \mathsf{El}\,A \to U)\,(P_= : P\,a \to P\,a' \to \text{SProp}) \to U
\end{aligned}
$$

$$
\begin{aligned}
&\mathsf{El} : U \to \text{Type} \\
&\mathsf{El}\,N \equiv \mathbb{N} \\
&\mathsf{El}\,(\Pi\,A\,A_=\,P\,P_=) \equiv (f : (a : \mathsf{El}\,A) \to \mathsf{El}\,(P\,a)) \\
&\qquad \times (f_e : A_=\,a\,a' \to P_=\,(f\,a)\,(f\,a'))
\end{aligned}
$$

small IR

(* Definition of equalities *)

# Let's just hack our way through it

Inductive $U$ :≡

$$
\begin{array}{l}
\mid N : U \\
\mid \Pi : (A : U)\,(A_= : A \to A \to \mathrm{SProp}) \\
\qquad (P : \mathrm{El}\,A \to U)\,(P_= : P\,a \to P\,a' \to \mathrm{SProp}) \to U
\end{array}
$$

$$
\begin{array}{lcl}
\mathrm{El} : U \to \mathrm{Type} & & \\
\mathrm{El}\,N & \equiv & \mathbb{N} \\
\mathrm{El}\,(\Pi\,A\,A_=\,P\,P_=) & \equiv & (f : (a : \mathrm{El}\,A) \to \mathrm{El}\,(P\,a)) \\
& & \times\,(f_e : A_=\,a\,a' \to P_=\,(f\,a)\,(f\,a'))
\end{array}
$$

(* Definition of equalities without using $A_=$ or $P_=$ *)

# Let's just hack our way through it

Inductive $U_e : U \to \text{Type} :\equiv$

$$
\begin{aligned}
| \; & N_e : U_e \, N \\
| \; & \Pi_e : (A : U)\,(A_e : U_e\,A) \\
& \quad (P : \mathsf{El}\,A \to U)\,(P_e : (a : A) \to U_e\,(P\,a)) \\
& \quad (P_{ext} : a \;{}_{A}{\sim}_{A}\; a' \to P\,a \sim_U P\,a') \\
& \quad \to U_e\,(\Pi\,A\,(\_ \;{}_{A}{\sim}_{A}\; \_)\,P\,(\lambda\,a\,a'\,.\,\_ \;{}_{P\,a}{\sim}_{P\,a'}\; \_))
\end{aligned}
$$

# Let's just hack our way through it

$$
\begin{aligned}
&\text{Inductive } U_e : U \to \text{Type} :\equiv \\
&\mid N_e : U_e\, N \\
&\mid \Pi_e : (A : U)\,(A_e : U_e\, A) \\
&\qquad (P : \mathrm{El}\, A \to U)\,(P_e : (a : A) \to U_e\,(P\, a)) \\
&\qquad (P_{ext} : a \;{}_{A}{\sim}_{A}\; a' \to P\, a \sim_U P\, a') \\
&\qquad \to U_e\,(\Pi\, A\,(\_ \;{}_{A}{\sim}_{A}\; \_)\; P\,(\lambda\, a\, a' .\, \_ \;{}_{P\,a}{\sim}_{P\,a'}\; \_))
\end{aligned}
$$

$$U' = (A : U) \times (U_e\, A)$$

# Well, that was easy

Surprisingly enough, it works.

We can get an inductive-recursive universe hierarchy with

# Well, that was easy

Surprisingly enough, it works.

We can get an inductive-recursive universe hierarchy with

- A type coercion operator,

# Well, that was easy

Surprisingly enough, it works.

We can get an inductive-recursive universe hierarchy with

- A type coercion operator,
- $\Sigma$-types, W-types, Id-types,

# Well, that was easy

Surprisingly enough, it works.

We can get an inductive-recursive universe hierarchy with

- A type coercion operator,
- Σ-types, W-types, Id-types,
- quotient types,

# Well, that was easy

Surprisingly enough, it works.

We can get an inductive-recursive universe hierarchy with

- A type coercion operator,
- $\Sigma$-types, W-types, Id-types,
- quotient types,
- a universe of propositions with Propext,

# Well, that was easy

Surprisingly enough, it works.

We can get an inductive-recursive universe hierarchy with

- A type coercion operator,
- Σ-types, W-types, Id-types,
- quotient types,
- a universe of propositions with Propext,
- universe embeddings.

# Well, that was easy

Surprisingly enough, it works.

We can get an inductive-recursive universe hierarchy with

- A type coercion operator,
- Σ-types, W-types, Id-types,
- quotient types,
- a universe of propositions with Propext,
- universe embeddings.

Syntactic translation of MLTT + funext + propext + UIP + quotients into MLTT + SProp which preserves conversion.

# II.

# Proof-relevant setoids

# Choice issues

When working with setoids, we encounter choice issues

Computational data and equality proofs live in different worlds
→ difference between Σ and ∃

# Choice issues

When working with setoids, we encounter choice issues

Computational data and equality proofs live in different worlds
→ difference between Σ and ∃

$$(x : A) \to \Sigma(y : B).R\,a\,b \quad \to \quad \Sigma(f : A \to B).(x : A) \to R\,x\,f(x)$$

$$(x : A) \to \exists(y : B).R\,a\,b \quad \not\to \quad \exists(f : A \to B).(x : A) \to R\,x\,f(x)$$

# Review of universes

# Review of universes

**SProp**
Impredicative, definitionally proof-irrelevant, large elim only allowed for $\bot$

# Review of universes

**SProp**

Impredicative, definitionally proof-irrelevant, large elim only allowed for $\bot$ ★★★☆☆

# Review of universes

**SProp**
Impredicative, definitionally proof-irrelevant, large elim only allowed for ⊥ ★★★☆☆

**Prop**
Impredicative, morally proof-irrelevant, large elim only allowed for subsingletons (in particular, ⊥, Id, Acc)

# Review of universes

**SProp**
Impredicative, definitionally proof-irrelevant, large elim only allowed for $\bot$ ★★★☆☆

**Prop**
Impredicative, morally proof-irrelevant, large elim only allowed for subsingletons (in particular, $\bot$, Id, Acc) ★★★☆☆

# Review of universes

**SProp**
Impredicative, definitionally proof-irrelevant, large elim only allowed for $\bot$ ★★★☆☆

**Prop**
Impredicative, morally proof-irrelevant, large elim only allowed for subsingletons (in particular, $\bot$, Id, Acc) ★★★☆☆

**Impredicative Set**
Impredicative, proof-relevant, weird, large elim only allowed for small inductives

# Review of universes

**SProp**
Impredicative, definitionally proof-irrelevant, large elim only allowed for ⊥ ★★★☆☆

**Prop**
Impredicative, morally proof-irrelevant, large elim only allowed for subsingletons (in particular, ⊥, Id, Acc) ★★★☆☆

**Impredicative Set**
Impredicative, proof-relevant, weird, large elim only allowed for small inductives ★★☆☆☆

# Review of universes

**SProp**
Impredicative, definitionally proof-irrelevant, large elim only allowed for ⊥ ★★★☆☆

**Prop**
Impredicative, morally proof-irrelevant, large elim only allowed for subsingletons (in particular, ⊥, Id, Acc) ★★★☆☆

**Impredicative Set**
Impredicative, proof-relevant, weird, large elim only allowed for small inductives ★★☆☆☆

**Type**
Predicative hierarchy, proof-relevant, large elim allowed

# Review of universes

**SProp**
Impredicative, definitionally proof-irrelevant, large elim only allowed for ⊥ ★★★☆☆

**Prop**
Impredicative, morally proof-irrelevant, large elim only allowed for subsingletons (in particular, ⊥, Id, Acc) ★★★☆☆

**Impredicative Set**
Impredicative, proof-relevant, weird, large elim only allowed for small inductives ★★☆☆☆

**Type**
Predicative hierarchy, proof-relevant, large elim allowed ★★★★★

# Varieties of setoids

# Varieties of setoids

**SProp setoids**

Once you truncate something, it's lost for good → no choice at all

# Varieties of setoids

**SProp setoids**
Once you truncate something, it's lost for good → no choice at all

**Prop setoids**
Large elimination of accessibility → $\Sigma^0_1$-choice

# Type-valued setoids

Using `Type` means that the truncation barely does anything: we simply change the equality relation to the trivial one.

This allows us to include some cool choice principles:

# Type-valued setoids

Using `Type` means that the truncation barely does anything: we simply change the equality relation to the trivial one.

This allows us to include some cool choice principles:

**Function comprehension / Unique choice**
Functional relations are the same thing as functions

# Type-valued setoids

Using $\mathsf{Type}$ means that the truncation barely does anything: we simply change the equality relation to the trivial one.

This allows us to include some cool choice principles:

**Function comprehension / Unique choice**
Functional relations are the same thing as functions

**Countable and dependent choice**
Since the setoid equality on $\mathbb{N}$ coincides with the meta-equality, every function out of $\mathbb{N}$ is automatically a setoid morphism.

# Type-valued setoids

Using Type means that the truncation barely does anything: we simply change the equality relation to the trivial one.

This allows us to include some cool choice principles:

**Choice for higher order types**
For the setoid equality on $\mathbb{N} \to \mathbb{N}$ to coincide with the meta-equality, we need function extensionality in the meta...

# Type-valued setoids

Using Type means that the truncation barely does anything: we simply change the equality relation to the trivial one.

This allows us to include some cool choice principles:

**Choice for higher order types**
For the setoid equality on $\mathbb{N} \to \mathbb{N}$ to coincide with the meta-equality, we need function extensionality in the meta...

What a coincidence! We have a translation that does just that ☺
Type-valued setoids inside the SProp-valued setoid model have choice for all Martin-Löf types

# Type-valued setoids

Using Type means that the truncation barely does anything: we simply change the equality relation to the trivial one.

This allows us to include some cool choice principles:

**Choice for higher order types**
For the setoid equality on $\mathbb{N} \to \mathbb{N}$ to coincide with the meta-equality, we need function extensionality in the meta...

What a coincidence! We have a translation that does just that ☺
Type-valued setoids **inside the SProp-valued setoid model** have choice for all Martin-Löf types

(cf. Rathjen, Choice principles in constructive and classical set theories)

# What about Impredicative Set?

idk, Impredicative-Set-valued setoids seem to sit somewhere inbetween Prop-valued and Type-valued setoids

# Universes for proof-relevant setoids

Take the universe construction from earlier, and substitute SProp for your favourite universe. It just works!

# A syntactic model?

Can we get the ultimate setoid translation out of this?

$$\Gamma \vdash t : A \quad \rightsquigarrow \quad [\![\Gamma]\!] \vdash [\![t]\!] : [\![A]\!]$$
$$\Gamma \vdash t \equiv u : A \quad \rightsquigarrow \quad [\![\Gamma]\!] \vdash [\![t]\!] \equiv [\![u]\!] : [\![A]\!]$$

# A syntactic model?

Can we get the ultimate setoid translation out of this?

$$\Gamma \vdash t : A \quad \rightsquigarrow \quad [\![\Gamma]\!] \vdash [\![t]\!] : [\![A]\!]$$
$$\Gamma \vdash t \equiv u : A \quad \rightsquigarrow \quad [\![\Gamma]\!] \vdash [\![t]\!] \equiv [\![u]\!] : [\![A]\!]$$

Unfortunately, no ☹

# A syntactic model?

Can we get the ultimate setoid translation out of this?

$$\Gamma \vdash t : A \quad \rightsquigarrow \quad [\![\Gamma]\!] \vdash [\![t]\!] : [\![A]\!]$$
$$\Gamma \vdash t \equiv u : A \quad \rightsquigarrow \quad [\![\Gamma]\!] \vdash [\![t]\!] \equiv [\![u]\!] : [\![A]\!]$$

Unfortunately, no ☹

Substitution don't go under binders: $(\lambda x.\, t)[\sigma] \not\equiv \lambda x.\, t[\sigma^{\uparrow}]$
Barras, Coquand, Huber, "A Generalization of Takeuti-Gandy Interpretation"

# Questions

- Can we derive a systematic encoding of double induction-recursion from this hack?
- Can we find a nice-ish syntax for the "proof-relevant observational type theory" of this weak model?


Thank you!