

# Setoids in Intensional Type Theory

Andrew Pitts



UNIVERSITY OF  
CAMBRIDGE

Topos Institute, 22 October 2026

# Setoids in Intensional Type Theory

Andrew Pitts



UNIVERSITY OF  
CAMBRIDGE

Topos Institute, 22 October 2026

Will try not to be too technical.  
Feel free to interrupt with questions.

# Motivations

Why I'm interested in dependent type theory:

- ▶ *extensional* type theory (ETT) and topos theory
- ▶ *intensional* type theory (ITT) and programming language theory

# Motivations

Why I'm interested in dependent type theory:

- ▶ *extensional* type theory (ETT) and topos theory  
toposes have a beautifully simple category-theoretic definition whose deep consequences are often most easily explored via an *internal language* combining intuitionistic higher-order logic and ETT (for which there's no convenient machine-assistance, unfortunately)
- ▶ *intensional* type theory (ITT) and programming language theory

# Motivations

Why I'm interested in dependent type theory:

- ▶ *extensional* type theory (ETT) and topos theory  
toposes have a beautifully simple category-theoretic definition whose deep consequences are often most easily explored via an *internal language* combining intuitionistic higher-order logic and ETT (for which there's no convenient machine-assistance, unfortunately)
- ▶ *intensional* type theory (ITT) and programming language theory  
ITT's inductively defined datatypes are great for defining and proving properties of syntax and operational semantics and there is excellent machine-assistance for ITT (I use Agda)

# Motivations

Why I'm interested in dependent type theory:

- ▶ *extensional* type theory (ETT) and topos theory
- ▶ *intensional* type theory (ITT) and programming language theory

**This talk:** a newish\* take on “**displayed setoids**” in ITT which gives a semantics for ETT entirely within ITT.

\* there's a precursor in Martin Hofmann's PhD thesis (1995);  
re-discovered by me and independently in recent work of Loïc Pujet (TYPES 2025)

# Motivations

Why I'm interested in dependent type theory:

- ▶ *extensional* type theory (ETT) and topos theory
- ▶ *intensional* type theory (ITT) and programming language theory

**This talk:** a newish take on “**displayed setoids**” in ITT which gives a semantics for ETT entirely within ITT.

Paper and Agda code accompanying this talk at:

<https://amp12.github.io/IntensionalSetoids/>

# Intensional Martin-Löf Type Theory (ITT)

*Judgements* (forms of assertion):

|              |                                |
|--------------|--------------------------------|
| $a : A$      | things $a$ have types $A$      |
| $a = a' : A$ | equality of things is per type |

# Intensional Martin-Löf Type Theory (ITT)

*Judgements* (forms of assertion):

$a : A$

things  $a$  have types  $A$

$a = a' : A$

equality of things is per type

$A : \mathcal{U}_n$

types  $A$  are just things (in a universe  $\mathcal{U}_n$ )

$\mathcal{U}_n : \mathcal{U}_{n+1}$

universes are also things

$x : A \vdash b x : B x$

things and their types can depend on things

# Intensional Martin-Löf Type Theory (ITT)

*Judgements* (forms of assertion):

|              |                                |
|--------------|--------------------------------|
| $a : A$      | things $a$ have types $A$      |
| $a = a' : A$ | equality of things is per type |

*Provable judgements* are inductively generated

by rules for  **$\Pi$ -types** (including  $\eta$ )  
and for **dependent record types** (including  $\eta$ ), e.g.  $\Sigma$ -types  
and for **inductive datatypes** of various kinds (including  
inductive-recursive types)

# Intensional Martin-Löf Type Theory (ITT)

*Logic* emerges

(contrast classical foundations: first FOL, then ZFC axioms)

it is neutrally constructive

(classical laws can be postulated without inconsistency, cf. Lean)

it is proof relevant

(can regard some types as **propositions** and some things as **proofs**)

it is decidable which judgements are provable [good!]

but functions are not extensional with respect to  
propositional equality (**Id**-types) [bad!]

# ITT identity types $\text{Id}_A a a'$

are just (indexed) inductive datatypes of a particular shape:

$$\text{formation: } \frac{\vdash A : \mathcal{U} \quad \vdash a : A \quad \vdash a' : A}{\vdash \text{Id}_A a a' : \mathcal{U}}$$

$$\text{constructor: } \frac{\vdash A : \mathcal{U} \quad \vdash a : A}{\vdash \text{refl}_A a : \text{Id}_A a a}$$

other rules (*elimination* and *computation*) are automatic from those

Simple and consequential (HoTT)  
but can be very tedious

(e.g. identity proofs (things in  $\text{Id}_A a a'$ ) give rise to annoying explicit transport functions  $B a \rightarrow B a'$ , rather than  $\vdash B a = B a' : \mathcal{U}$ )

# Extensional Martin-Löf Type Theory (ETT)

ETT identity types are no longer an instance of inductive datatypes:  
same formation and constructor rules as for `Id`  
but elimination and computation rules are replaced by

$$\text{reflection of identity proofs: } \frac{\vdash e : \text{Id}_A a a'}{\vdash a = a' : A}$$

$$\text{uniqueness of identity proofs: } \frac{\vdash e : \text{Id}_A a a'}{\vdash e = \text{refl}_A a : \text{Id}_A a a}$$

In ETT there is really only one notion of equality [good!],  
function extensionality holds [good!],  
and transport along equality is invisible [good!]  
but provability of judgements is in general undecidable [bad!]

# Conventional wisdom

The lack of function extensionality in ITT seems like a real handicap for using  
ITT-based theorem-provers for formalizing *denotational* semantics of PLs  
(or indeed, most mathematics)

But if one wants extensionality within intensional type theory,  
conventional wisdom says to use  
“the setoid model of type theory”  
(but be warned, it's hell)

# Conventional wisdom

The lack of function extensionality in ITT seems like a real handicap for using ITT-based theorem-provers for formalizing *denotational* semantics of PLs (or indeed, most mathematics)

But if one wants extensionality within intensional type theory, conventional wisdom says to use “**the** setoid model of type theory”  
(but be warned, it’s hell)

Classical notion of “setoid = set + equivalence relation” can be transferred from set theory to type theory in several different ways. The equivalence relation could be valued in a universe of types (Palmgren [3]), an impredicative universe of propositions (Hofmann [2]), or a predicative universe of proof irrelevant (“strict”) propositions (Altenkirch *et al* [1]).

[1] Thorsten Altenkirch, Simon Boulier, Ambrus Kaposi, Christian Sattler and Filippo Sestini, *Constructing a universe for the setoid model*, FoSSaCS 2021.

[2] Martin Hofmann, *Extensional Concepts in Intensional Type Theory*, PhD Thesis, Edinburgh, 1995.

[3] Erik Palmgren, *From type theory to setoids and back*, MSCS 32(2022)1283–1312.

# Conventional wisdom

The lack of function extensionality in ITT seems like a real handicap for using ITT-based theorem-provers for formalizing *denotational* semantics of PLs (or indeed, most mathematics)

But if one wants extensionality within intensional type theory, conventional wisdom says to use “the setoid model of **type theory**”  
(but be warned, it’s hell)

Which type theory?  
Ans: here it is ETT  
but for any, modelling universes has proved to be problematic [1, 3].

[1] Thorsten Altenkirch, Simon Boulier, Ambrus Kaposi, Christian Sattler and Filippo Sestini, *Constructing a universe for the setoid model*, FoSSaCS 2021.

[2] Martin Hofmann, *Extensional Concepts in Intensional Type Theory*, PhD Thesis, Edinburgh, 1995.

[3] Erik Palmgren, *From type theory to setoids and back*, MSCS 32(2022)1283–1312.

# Conventional wisdom

The lack of function extensionality in ITT seems like a real handicap for using ITT-based theorem-provers for formalizing *denotational* semantics of PLs (or indeed, most mathematics)

But if one wants extensionality within intensional type theory, conventional wisdom says to use “the setoid **model** of type theory”  
(but be warned, it’s hell)

It’s common to describe the various *categorical structures* built from setoids that could be used to give a semantics of (some) type theory, but no-one defines within ITT *the actual semantics functions* (even Palmgren [3])

[1] Thorsten Altenkirch, Simon Boulier, Ambrus Kaposi, Christian Sattler and Filippo Sestini, *Constructing a universe for the setoid model*, FoSSaCS 2021.

[2] Martin Hofmann, *Extensional Concepts in Intensional Type Theory*, PhD Thesis, Edinburgh, 1995.

[3] Erik Palmgren, *From type theory to setoids and back*, MSCS 32(2022)1283–1312.

# Conventional wisdom

The lack of function extensionality in ITT seems like a real handicap for using ITT-based theorem-provers for formalizing *denotational* semantics of PLs (or indeed, most mathematics)

But if one wants extensionality within intensional type theory, conventional wisdom says to use “the setoid model of type theory” (but be warned, it’s hell)

Here:

- ▶ new notion of family of type-valued setoids in ITT
- ▶ setoid semantics for ETT within ITT
- ▶ use safe Agda to work with ITT

# Agda theorem-prover

I will use Agda with its `--safe` option for machine-assistance with ITT.

Some Agda notation:

$\Pi$ -types             $(x : A) \rightarrow B\ x$

function types     $A \rightarrow B$

universe type    `Set`

identity types     $a \equiv a'$     (common type of  $a$  and  $a'$  is implicit)

# Setoids in safe Agda

A *setoid* is a type  $A : \text{Set}$  equipped with  $(A \ni \_ \sim \_) : A \rightarrow A \rightarrow \text{Set}$

think of  $(A \ni x \sim y) : \text{Set}$  as the type of proofs that  $A$  considers  $x$  and  $y$  to be equal

and functions

$$\forall x \rightarrow (A \ni x \sim x)$$

$$\forall x y \rightarrow (A \ni x \sim y) \rightarrow (A \ni y \sim x)$$

$$\forall x y z \rightarrow (A \ni x \sim y) \rightarrow (A \ni y \sim z) \rightarrow (A \ni x \sim z)$$

# Setoids in safe Agda

A *setoid* is a type  $A : \text{Set}$  equipped with  $(A \ni \_ \sim \_) : A \rightarrow A \rightarrow \text{Set}$

think of  $(A \ni x \sim y) : \text{Set}$  as the type of proofs that  $A$  considers  $x$  and  $y$  to be equal

and functions

$$\forall x \rightarrow (A \ni x \sim x)$$

$$\forall x y \rightarrow (A \ni x \sim y) \rightarrow (A \ni y \sim x)$$

$$\forall x y z \rightarrow (A \ni x \sim y) \rightarrow (A \ni y \sim z) \rightarrow (A \ni x \sim z)$$

$f : A \rightarrow A'$  is a *morphism of setoids* if it comes with a function

$$\forall x y \rightarrow (A \ni x \sim y) \rightarrow (A' \ni f x \sim f y)$$

# Setoids in safe Agda

A *setoid* is a type  $A : \text{Set}$  equipped with  $(A \ni \_ \sim \_) : A \rightarrow A \rightarrow \text{Set}$

think of  $(A \ni x \sim y) : \text{Set}$  as the type of proofs that  $A$  considers  $x$  and  $y$  to be equal

and functions

$$\forall x \rightarrow (A \ni x \sim x)$$

$$\forall x y \rightarrow (A \ni x \sim y) \rightarrow (A \ni y \sim x)$$

$$\forall x y z \rightarrow (A \ni x \sim y) \rightarrow (A \ni y \sim z) \rightarrow (A \ni x \sim z)$$

$f : A \rightarrow A'$  is a *morphism of setoids* if it comes with a function

$$\forall x y \rightarrow (A \ni x \sim y) \rightarrow (A' \ni f x \sim f y)$$

No groupoid laws or coherence equalities!

# Setoids in safe Agda

A *setoid* is a type  $A : \text{Set}$  equipped with  $(A \ni \_ \sim \_) : A \rightarrow A \rightarrow \text{Set}$

think of  $(A \ni x \sim y) : \text{Set}$  as the type of proofs that  $A$  considers  $x$  and  $y$  to be equal

and functions

$$\forall x \rightarrow (A \ni x \sim x)$$

$$\forall x y \rightarrow (A \ni x \sim y) \rightarrow (A \ni y \sim x)$$

$$\forall x y z \rightarrow (A \ni x \sim y) \rightarrow (A \ni y \sim z) \rightarrow (A \ni x \sim z)$$

$f : A \rightarrow A'$  is a *morphism of setoids* if it comes with a function

$$\forall x y \rightarrow (A \ni x \sim y) \rightarrow (A' \ni f x \sim f y)$$

**Guiding Principle:** types that are used as propositions (like  $A \ni x \sim y$ ) should never have their elements compared for intensional identity

# Displayed setoid = ?

The crucial question (for modelling dependent types):

Given a setoid  $A$ , what structure is needed to make  $B : A \rightarrow \text{Set}$  into a well-behaved displayed setoid (*aka* family of setoids)?

Answers in the literature:

Palmgren does not follow our Guiding Principle (and is more complicated).

Altenkirch *et al* avoid the problem by valuing  $\sim$  in `Prop`, but that takes one out of safe Agda.

Pujet (TYPES 2025) takes essentially the same approach as here.

# Displayed setoid = ?

The crucial question (for modelling dependent types):

Given a setoid  $A$ , what structure is needed to make  $B : A \rightarrow \mathbf{Set}$  into a well-behaved displayed setoid (*aka* family of setoids)?

Certainly need setoid structure on  $\Sigma_{x:A} B x$  making  $\mathsf{fst}$  a morphism:

$$\begin{array}{ccc} \Sigma_{x:A} B x & \Sigma_{x:A} B x \ni (x, y) & \overset{?}{\sim} (x', y') \\ \mathsf{fst} \downarrow & & \downarrow \\ A & A \ni x & \sim x' \end{array}$$

# Displayed setoid = ?

The crucial question (for modelling dependent types):

Given a setoid  $A$ , what structure is needed to make  $B : A \rightarrow \text{Set}$  into a well-behaved displayed setoid (*aka* family of setoids)?

Certainly need setoid structure on  $\sum_{x:A} Bx$  making  $\text{fst}$  a morphism:

$$\begin{array}{ccc} \sum_{x:A} Bx & (A \ni x \sim x') & \times (B \ni x, y \approx x', y') \\ \text{fst} \downarrow & \downarrow & \\ A & A \ni x \sim x' & \end{array}$$

cartesian product, not dependent product, so  
that proofs of  $B \ni x, y \approx x', y'$  do not depend  
upon proofs of  $A \ni x \sim x'$

# Heterogeneous equivalence relation

$$(B \ni \_, \_ \approx \_, \_) : (x : A)(y : B x)(x' : A)(y' : B x') \rightarrow \text{Set}$$

structure ensuring  $(A \ni x \sim x') \times (B \ni x, y \approx x', y')$  is an equiv. rel. for  $\Sigma A B$ :

structure ensuring  $B x$  has congruence property (transport) for any  $A \ni x \sim x'$ :

# Heterogeneous equivalence relation

$$(B \ni \_, \_ \approx \_, \_) : (x : A)(y : B x)(x' : A)(y' : B x') \rightarrow \text{Set}$$

structure ensuring  $(A \ni x \sim x') \times (B \ni x, y \approx x', y')$  is an equiv. rel. for  $\Sigma A B$ :

$$\forall x y \rightarrow (B \ni x, y \approx x, y)$$

$$\forall x y x' y' \rightarrow (A \ni x \sim x') \rightarrow (B \ni x, y \approx x', y') \rightarrow (B \ni x', y' \approx x, y)$$

$$\begin{aligned} &\forall x y x' y' x'' y'' \rightarrow (A \ni x \sim x') \rightarrow (A \ni x' \sim x'') \rightarrow \\ &(B \ni x, y \approx x', y') \rightarrow (B \ni x', y' \approx x'', y'') \rightarrow (B \ni x, y \approx x'', y'') \end{aligned}$$

structure ensuring  $Bx$  has congruence property (transport) for any  $A \ni x \sim x'$ :

$$\text{coe} : \forall \{x x'\} \rightarrow (A \ni x \sim x') \rightarrow Bx \rightarrow Bx'$$

$$\forall x y x' \rightarrow (e : A \ni x \sim x') \rightarrow (B \ni x, y \approx x', \text{coe } e y)$$

# Displayed setoids

*Displayed setoid* over a setoid  $A$  is  
 $B : A \rightarrow \text{Set}$  + heterogeneous equiv. rel.

*Element* of a displayed setoid  $B : A \rightarrow \text{Set}$  is  
a dependent function  $b : (x : A) \rightarrow B\,x$  equipped with  
 $\forall x\,x' \rightarrow (A \ni x \sim x') \rightarrow (B \ni x, b\,x \approx x', b\,x')$

# Displayed setoids

*Displayed setoid* over a setoid  $A$  is  
 $B : A \rightarrow \text{Set}$  + heterogeneous equiv. rel.

*Element* of a displayed setoid  $B : A \rightarrow \text{Set}$  is  
a dependent function  $b : (x : A) \rightarrow B\,x$  equipped with  
 $\forall x\,x' \rightarrow (A \ni x \sim x') \rightarrow (B \ni x, b\,x \approx x', b\,x')$

Setoids in intensional type theory, their morphisms, displays and elements  
can be organised into a **setoid-enriched** category-with-families [definition omitted]  
with lots of good properties for modelling type theories;  
but there is a problem...

# Setoid universes

**The problem:** the collection of all setoids (of some size) can be made into a setoid in several ways (intensional identity, isomorphism, ...), but none, it seems, can be endowed with a generic displayed setoid giving rise to a full model of type-theory-with-universes.

# Setoid universes

**The problem:** the collection of all setoids (of some size) can be made into a setoid in several ways (intensional identity, isomorphism, ...), but none, it seems, can be endowed with a generic displayed setoid giving rise to a full model of type-theory-with-universes.

**The solution:** decide in advance what constructs the universe should contain and then make an *inductive* definition of **codes for setoids** and simultaneously *recursive* definition of the **setoid denoted by a code**.

Altenkirch *et al* and Palmgren both do this, but in very different ways. I adapt and extend Altenkirch's approach.

# Setoid universes of types in safe Agda

inductive

$\mathcal{U}_n : \text{Set}$

recursive

$\mathcal{E}_n : \mathcal{U}_n \rightarrow \text{Set}$

$\mathcal{U}_n \ni \_ \sim \_ : \mathcal{U}_n \rightarrow \mathcal{U}_n \rightarrow \text{Set}$

$\mathcal{E}_n \ni \_, \_ \approx \_, \_ : (X : \mathcal{U}_n)(x : \mathcal{E}_n X)(X' : \mathcal{U}_n)(x' : \mathcal{E}_n X') \rightarrow \text{Set}$

(where  $n : \mathbb{N}$ , the type of natural numbers)

# Setoid universes of types in safe Agda

Universe types:  $U_n : \mathcal{U}_{n+1} \quad \mathcal{E}_{n+1}(U_n) = \mathcal{U}_n$

$$\mathcal{U}_{n+1} \ni U_n \sim U_n = 1 \qquad \mathcal{E}_{n+1} \ni U_n, X \approx U_n, X' = (\mathcal{U}_n \ni X \sim X')$$

$$\mathcal{U}_{n+1} \ni U_n \sim \_ = \emptyset \qquad \mathcal{E}_{n+1} \ni U_n, \_ \approx \_, \_ = \emptyset$$

$$\mathcal{U}_{n+1} \ni \_ \sim U_n = \emptyset \qquad \mathcal{E}_{n+1} \ni \_, \_ \approx U_n, \_ = \emptyset$$

# Setoid universes of types in safe Agda

Universe types:  $U_n : \mathcal{U}_{n+1} \quad \mathcal{E}_{n+1}(U_n) = \mathcal{U}_n$

$$\mathcal{U}_{n+1} \ni U_n \sim U_n = 1 \quad \mathcal{E}_{n+1} \ni U_n, X \approx U_n, X' = (\mathcal{U}_n \ni X \sim X')$$

$$\mathcal{U}_{n+1} \ni U_n \sim \_ = \emptyset \quad \mathcal{E}_{n+1} \ni U_n, \_ \approx \_, \_ = \emptyset$$

$$\mathcal{U}_{n+1} \ni \_ \sim U_n = \emptyset \quad \mathcal{E}_{n+1} \ni \_, \_ \approx U_n, \_ = \emptyset$$

Natural number type:  $\text{Nat} : \mathcal{U}_0 \quad \mathcal{E}_0(\text{Nat}) = \mathbb{N}$

$$\mathcal{U}_0 \ni \text{Nat} \sim \text{Nat} = 1 \quad \mathcal{E}_0 \ni \text{Nat}, n \approx \text{Nat}, n' = (n \equiv n')$$

$$\mathcal{U}_0 \ni \text{Nat} \sim \_ = \emptyset \quad \mathcal{E}_0 \ni \text{Nat}, \_ \approx \_, \_ = \emptyset$$

$$\mathcal{U}_0 \ni \_ \sim \text{Nat} = \emptyset \quad \mathcal{E}_0 \ni \_, \_ \approx \text{Nat}, \_ = \emptyset$$

# Setoid universes of types in safe Agda

Universe types:  $U_n : \mathcal{U}_{n+1} \quad \mathcal{E}_{n+1}(U_n) = \mathcal{U}_n$

$$\mathcal{U}_{n+1} \ni U_n \sim U_n = 1 \qquad \mathcal{E}_{n+1} \ni U_n, X \approx U_n, X' = (\mathcal{U}_n \ni X \sim X')$$

$$\mathcal{U}_{n+1} \ni U_n \sim \_ = \emptyset \qquad \mathcal{E}_{n+1} \ni U_n, \_ \approx \_, \_ = \emptyset$$

$$\mathcal{U}_{n+1} \ni \_ \sim U_n = \emptyset \qquad \mathcal{E}_{n+1} \ni \_, \_ \approx U_n, \_ = \emptyset$$

Natural number type:  $\text{Nat} : \mathcal{U}_0 \quad \mathcal{E}_0(\text{Nat}) = \mathbb{N}$

$$\mathcal{U}_0 \ni \text{Nat} \sim \text{Nat} = 1 \qquad \mathcal{E}_0 \ni \text{Nat}, n \approx \text{Nat}, n' = (n \equiv n')$$

$$\mathcal{U}_0 \ni \text{Nat} \sim \_ = \emptyset \qquad \mathcal{E}_0 \ni \text{Nat}, \_ \approx \_, \_ = \emptyset$$

$$\mathcal{U}_0 \ni \_ \sim \text{Nat} = \emptyset \qquad \mathcal{E}_0 \ni \_, \_ \approx \text{Nat}, \_ = \emptyset$$

Empty type: [description omitted here]

$\Pi$ -types: [description omitted here]

# Setoid universes of types in safe Agda

Identity types:  $\text{Id}_n : (X : \mathcal{U}_n) \quad \mathcal{E}_n(\text{Id}_n X x x') = (\mathcal{E}_n \ni X, x \approx X, x')$

$$\frac{(x x' : \mathcal{E}_n X)}{\mathcal{U}_n}$$

$$\mathcal{U}_n \ni (\text{Id}_n X x x') \sim (\text{Id}_n Y y y') = (\mathcal{U}_n \ni X \sim Y) \times (\mathcal{E}_n \ni X, x \approx Y, y) \times (\mathcal{E}_n \ni X, x' \approx Y, y')$$

$$\mathcal{U}_n \ni (\text{Id}_n X x x') \sim \_ = \emptyset$$

$$\mathcal{U}_n \ni \_ \sim (\text{Id}_n X x x') = \emptyset$$

$$\mathcal{E}_n \ni (\text{Id}_n \_ \_ \_), \_ \approx (\text{Id}_n \_ \_ \_), \_ = 1$$

$$\mathcal{E}_n \ni (\text{Id}_n \_ \_ \_), \_ \approx \_, \_ = \emptyset$$

$$\mathcal{E}_n \ni \_, \_ \approx (\text{Id}_n \_ \_ \_), \_ = \emptyset$$

# Setoid universes of types in safe Agda

Identity types:  $\text{Id}_n : (X : \mathcal{U}_n)$   
 $\frac{(x\ x' : \mathcal{E}_n X)}{\mathcal{U}_n}$

$$\mathcal{E}_n(\text{Id}_n X\ x\ x') = (\mathcal{E}_n \ni X, x \approx X, x')$$

$$\mathcal{U}_n \ni (\text{Id}_n X\ x\ x') \sim (\text{Id}_n Y\ y\ y') = (\mathcal{U}_n \ni X \sim Y) \times (\mathcal{E}_n \ni X, x \approx Y, y) \times (\mathcal{E}_n \ni X, x' \approx Y, y')$$

$$\mathcal{U}_n \ni (\text{Id}_n X\ x\ x') \sim \_ = \emptyset$$

$$\mathcal{U}_n \ni \_ \sim (\text{Id}_n X\ x\ x') = \emptyset$$

$$\mathcal{E}_n \ni (\text{Id}_n \_ \_), \_ \approx (\text{Id}_n \_ \_), \_ = 1$$

$$\mathcal{E}_n \ni (\text{Id}_n \_ \_), \_ \approx \_, \_ = \emptyset$$

$$\mathcal{E}_n \ni \_, \_ \approx (\text{Id}_n \_ \_), \_ = \emptyset$$

Models *extensional* identity types.

**Theorem.** These setoid universes in ITT support a semantics for ETT with a countable hierarchy of universes closed under  $\Pi$ -types, natural number type, empty type and identity types.

**Theorem.** These setoid universes in ITT support a semantics for ETT with a countable hierarchy of universes closed under  $\Pi$ -types, natural number type, empty type and identity types.

**Proof.** Everything is done in safe Agda

(see <https://amp12.github.io/IntensionalSetoids/>)

- inductive definition of the ETT syntax and type system ( $\vdash_{\text{ETT}}$ )  
using a library for well-scoped locally nameless representation of syntax  
(<https://github.com/amp12/WSLN>)
- inductive-recursive setoid universe structure  
(possible to augment with further typing constructs)
- setoid semantics of ETT types and terms and proof of soundness

□

# ITT setoid semantics of ETT types and terms

Hofmann-Streicher approach:

- ▶ first give semantic *partial* functions on raw types and terms
- ▶ then prove the partial functions are total for types and terms for which there exist proofs of well-formedness

ITT lacks suitable types of partial function

so instead we define the (type-valued) *graphs* of the semantics functions for raw types and terms and then prove them single-valued and total for well-formed ones.

What we end up with is...

- ▶  $\llbracket A \rrbracket_p : \mathcal{U}_n$ , if  $p$  is a proof that  $\vdash_{\text{ETT}} A : U_n$
- ▶  $\llbracket a \rrbracket_{p,q} : \mathcal{E}_n(\llbracket A \rrbracket_p)$ , if  $p$  is a proof that  $\vdash_{\text{ETT}} A : U_n$  and  $q$  is a proof that  $\vdash_{\text{ETT}} a : A$

- ▶  $\llbracket A \rrbracket_p : \mathcal{U}_n$ , if  $p$  is a proof that  $\vdash_{\text{ETT}} A : U_n$
- ▶  $\llbracket a \rrbracket_{p,q} : \mathcal{E}_n(\llbracket A \rrbracket_p)$ , if  $p$  is a proof that  $\vdash_{\text{ETT}} A : U_n$  and  $q$  is a proof that  $\vdash_{\text{ETT}} a : A$

**Proof irrelevance:**  $\mathcal{E}_n \ni \llbracket A \rrbracket_p, \llbracket a \rrbracket_{p,q} \approx \llbracket A \rrbracket_{p'}, \llbracket a \rrbracket_{p',q'}$   
 where  $p, p' : (\vdash_{\text{ETT}} A : U_n)$  and  $q, q' : (\vdash_{\text{ETT}} a : A)$

**Soundness:** if  $\vdash_{\text{ETT}} a = a' : A$ , then  $\mathcal{E}_n \ni \llbracket A \rrbracket_p, \llbracket a \rrbracket_{p,q} \approx \llbracket A \rrbracket_p, \llbracket a' \rrbracket_{p,q'}$   
 where  $p : (\vdash_{\text{ETT}} A : U_n)$ ,  $q : (\vdash_{\text{ETT}} a : A)$  and  $q' : (\vdash_{\text{ETT}} a' : A)$

- ▶  $\llbracket A \rrbracket_p : \mathcal{U}_n$ , if  $p$  is a proof that  $\vdash_{\text{ETT}} A : U_n$
- ▶  $\llbracket a \rrbracket_{p,q} : \mathcal{E}_n(\llbracket A \rrbracket_p)$ , if  $p$  is a proof that  $\vdash_{\text{ETT}} A : U_n$  and  $q$  is a proof that  $\vdash_{\text{ETT}} a : A$

**Proof irrelevance:**  $\mathcal{E}_n \ni \llbracket A \rrbracket_p, \llbracket a \rrbracket_{p,q} \approx \llbracket A \rrbracket_{p'}, \llbracket a \rrbracket_{p',q'}$   
 where  $p, p' : (\vdash_{\text{ETT}} A : U_n)$  and  $q, q' : (\vdash_{\text{ETT}} a : A)$

**Soundness:** if  $\vdash_{\text{ETT}} a = a' : A$ , then  $\mathcal{E}_n \ni \llbracket A \rrbracket_p, \llbracket a \rrbracket_{p,q} \approx \llbracket A \rrbracket_p, \llbracket a' \rrbracket_{p,q'}$   
 where  $p : (\vdash_{\text{ETT}} A : U_n)$ ,  $q : (\vdash_{\text{ETT}} a : A)$  and  $q' : (\vdash_{\text{ETT}} a' : A)$

**Corollary.** (Consistency) In safe Agda there is a function

$$(t : \text{Tm}) \rightarrow (\vdash_{\text{ETT}} t : \text{Emp}) \rightarrow \emptyset$$

( $\text{Tm}$  is the inductive ITT type of ETT terms,  $\text{Emp}$  is the empty type in ETT and  $\emptyset$  is the empty type in ITT)

**Proof.** If  $t : \text{Tm}$  and  $p : (\vdash_{\text{ETT}} t : \text{Emp})$  then soundness of the semantics gives  $\llbracket t \rrbracket_{-,p} : \llbracket \text{Emp} \rrbracket_-$ . But by definition  $\llbracket \text{Emp} \rrbracket_- \equiv \emptyset$ , so  $\lambda t p \rightarrow \llbracket t \rrbracket_{-,p}$  is the desired function.  $\square$

# Other applications of this notion of displayed setoid in ITT?

ITT is a proof-theoretically weak system, adequate for formalizing formal languages, e.g. ITT and ETT.

What can be proved about ITT and ETT purely within ITT?

E.g. decidability of provable judgements in ITT

There is Andreas Abel et al, POPL 2017

but can one use setoids in ITT to give an intensional version of Thierry Coquand's cute *Canonicity and normalization for dependent type theory* (TCS 777, 2019)?

(What about undecidability of provable judgements in ETT?

See Martin Hofmann's 1995 PhD thesis, 1995 — but is there a “synthetic” proof in ITT?)

The Agda development described in this talk and a paper are at  
<https://amp12.github.io/IntensionalSetoids/>

## References

Andreas Abel, Joakim Öhman, and Andrea Vezzosi, *Decidability of conversion for type theory in type theory*, POPL 2017.

Thorsten Altenkirch, Simon Boulier, Ambrus Kaposi, Christian Sattler and Filippo Sestini, *Constructing a universe for the setoid model*, FoSSaCS 2021.

Thierry Coquand, *Canonicity and normalization for dependent type theory*, TCS 777(2019)184–191.

Martin Hofmann, *Extensional Concepts in Intensional Type Theory*, PhD Thesis, Edinburgh, 1995.

Erik Palmgren, *From type theory to setoids and back*, MSCS 32(2022)1283–1312.

Loïc Pujet, *An inductive universe for setoids*, TYPES 2025.

Thank you for your attention!