OTT

Observational Type Theory

meets

The Calculus of Inductive Constructions

CIC

Observational Type Theory

OTT is an extension of Martin-Löf Type Theory

It swaps the inductive equality of MLTT for the observational equality: a propositional equality defined on a type by type basis

$$\frac{A : \mathrm{Type} \qquad t, u : A}{t \sim_A u : \mathrm{Prop}}$$

This equality recovers extensionality principles for MLTT (function extensionality, proposition extensionality...) without sacrificing computational properties.

Altenkirch, McBride, Swierstra '07. Observational Equality, Now!

Calculus of Inductive Constructions

CIC is the type theory behind Coq and Lean

On top of Martin-Löf Type Theory, it adds

- A comprehensive class of indexed inductive types

- Two impredicative universes of propositions

$$\frac{\Gamma \vdash A : \mathrm{Type} \qquad \Gamma, x : A \vdash B : \mathrm{Prop}}{\Gamma \vdash \Pi\,(x : A)\,.\,B : \mathrm{Prop}}$$

Prop is proof-relevant

SProp is proof-irrelevant

OTT + CIC = <3

We want an observational Coq!

OTT + CIC = <3

We want an observational Coq!

– Most of mathematics relies on quotients and extensionality principles, which are not available in Coq

OTT + CIC = <3

We want an observational Coq!

- Most of mathematics relies on quotients and extensionality principles, which are not available in Coq

- Cubical type theories are too complex for ordinary, set-truncated math : we do not want to prove that everything is a hset by hand.

OTT + CIC = <3

We want an observational Coq!

- Most of mathematics relies on quotients and extensionality principles, which are not available in Coq

- Cubical type theories are too complex for ordinary, set-truncated math : we do not want to prove that everything is a hset by hand.

- Extensionality principles + impredicativity = a proof assistant for 1-toposes

OTT + CIC = <3

We want an observational Coq!

- Most of mathematics relies on quotients and extensionality principles, which are not available in Coq

- Cubical type theories are too complex for ordinary, set-truncated math : we do not want to prove that everything is a hset by hand.

- Extensionality principles + impredicativity = a proof assistant for 1-toposes

Thus we need to make sure that OTT plays nicely with the features of the Coq proof assistant

OTT + CIC = <3

A programme unfolded in several steps:

TT^{obs} : MLTT with SProp and an observational equality

CC^{obs} : Adds support for an impredicative SProp

CIC^{obs} : Adds support for cast-on-reflexivity, adds support for general inductive types

The observational equality

We equip every type with a propositional relation $\sim$

$$\frac{A : \mathrm{Type} \qquad t, u : A}{t \sim_A u : \mathrm{SProp}} \qquad \frac{A : \mathrm{Type} \qquad t : A}{\mathrm{refl}(t) : t \sim_A t}$$

This is a strict proposition → any two proofs of equality are convertible

The observational equality

We equip every type with a propositional relation $\sim$

$$\frac{A : \mathrm{Type} \qquad\qquad t, u : A}{t \sim_A u : \mathrm{SProp}} \qquad\qquad \frac{A : \mathrm{Type} \qquad\qquad t : A}{\mathrm{refl}(t) : t \sim_A t}$$

This is a strict proposition → any two proofs of equality are convertible

Since strict propositions contains no computational information, we can postulate all the axioms we want – they won't block computation!

$\mathrm{funext} : (\prod (x : A)\,.\, f\,x \sim_B g\,x) \to f \sim_{A \to B} g$

$\mathrm{propext} : (P \to Q) \times (Q \to P) \to P \sim_{\mathrm{SProp}} Q$

$\mathrm{transp} : \prod\, (P : A \to \mathrm{SProp})\,(t : A)\,(x : P\,t)\,(u : A)\,(e : t \sim_A u)\,.\, P\,u$

The observational equality

Dependent funext and transp are enough to characterize the equality on inductive types and dependent products.

We also need to define the observational equality for the universe.
Since it cannot be univalent, we ask for the injectivity of type constructors:

$\pi_1^{\varepsilon} : (A \to B) \sim_{\text{Type}} (A' \to B') \to A' \sim_{\text{Type}} A$

$\pi_2^{\varepsilon} : (A \to B) \sim_{\text{Type}} (A' \to B') \to B \sim_{\text{Type}} B'$

antidiag : $A \sim_{\text{Type}} B \to \bot$ if A and B have different head constructors

etc.

The observational equality

Since the observational equality contains no computational info, how do we eliminate it?

The observational equality

Since the observational equality contains no computational info, how do we eliminate it? We add a primitive cast operator!

$$\frac{A, B : \mathsf{Type} \qquad e : A \sim_{\mathsf{Type}} B \qquad t : A}{\mathsf{cast}(A,B,e,t) : B}$$

$$\frac{A : \mathsf{Type} \qquad t : A}{\mathsf{cast}(A,A,\mathsf{refl}(A),t) \equiv t}$$

The observational equality

Since the observational equality contains no computational info, how do we eliminate it? We add a primitive cast operator!

$$\frac{A, B : \mathrm{Type} \quad e : A \sim_{\mathrm{Type}} B \quad t : A}{\mathrm{cast}(A,B,e,t) : B} \qquad \frac{A : \mathrm{Type} \quad t : A}{\mathrm{cast}(A,A,\mathrm{refl}(A),t) \equiv t}$$

The cast operator computes according to the head constructors of A and B

$$\mathrm{cast}(A \to B, A' \to B', e, f) \equiv \lambda(x : A').\ \mathrm{cast}(B, B', \pi_1^{\varepsilon} e, \mathrm{cast}(A', A, \pi_2^{\varepsilon} e, x))$$

$$\mathrm{cast}(A \to B, \mathbb{N}, e, f) \equiv \mathrm{exfalso}(\mathbb{N}, \mathrm{antidiag}(e))$$

The observational equality

Since the observational equality contains no computational info, how do we eliminate it? We add a primitive cast operator!

$$\frac{A, B : \mathrm{Type} \qquad e : A \sim_{\mathrm{Type}} B \qquad t : A}{\mathrm{cast}(A,B,e,t) : B} \qquad\qquad \frac{A : \mathrm{Type} \qquad t : A}{\mathrm{cast}(A,A,\mathrm{refl}(A),t) \equiv t}$$

The cast operator computes according to the head constructors of A and B

$$\mathrm{cast}(A \to B, A' \to B', e, f) \equiv \lambda(x : A').\ \mathrm{cast}(B, B', \pi_1^{\varepsilon} e, \mathrm{cast}(A', A, \pi_2^{\varepsilon} e, x))$$

$$\mathrm{cast}(A \to B, \mathbb{N}, e, f) \equiv \mathrm{exfalso}(\mathbb{N}, \mathrm{antidiag}(e))$$

With the cast, we can derive the J eliminator for Type-valued predicates:

$$\frac{t : A \qquad P : \prod (x : A).\ t \sim_A x \to \mathrm{Type} \qquad u : A \qquad e : t \sim_A u \qquad a : P\ t}{\mathrm{cast}(P\ t\ \mathrm{refl}(t), P\ u\ e, \mathrm{ap}\ P\ e, e), a) : P\ y\ e}$$

Indexed Inductive Types

First observation : we need to add new normal forms

Inductive eq (A : Type) (a : A) : A → Type :=
| eq_refl : eq A a a

Using the induction principle for eq, we can show that

$$\text{eq A t u} \longleftrightarrow t \sim_A u$$

Thus function extensionality is provable for eq, which implies that not every closed proof of eq reduces to eq_refl

Indexed Inductive Types

First observation : we need to add new normal forms

Inductive eq (A : Type) (a : A) (b : A) : Type :=
| eq_refl : $a \sim_A b \rightarrow$ eq A a b

Using the induction principle for eq, we can show that

$$\text{eq } A\ t\ u \longleftrightarrow t \sim_A u$$

Thus function extensionality is provable for eq, which implies that not every closed proof of eq reduces to eq_refl

We can recover canonicity by translating indices to parameters

Indexed Inductive Types

Second observation : equality between inductive types should not imply equality of the indices. Consider the following type:

Inductive Empty (A : Type) : Type := $\emptyset$

If Empty A $\sim_{Type}$ Empty B implies A $\sim_{Type}$ B, then we have a retract of Type inside Type, which is inconsistent

Instead, we use the equality of the constructor arguments

Indexed Inductive Types

```
Inductive vect (A : Type) : ℕ → Type :=
| vnil : vect A 0
| vcons : Π (m : ℕ) . A → vect A m → vect A (S m)
```

Indexed Inductive Types

```
Inductive vect (A : Type) (n : ℕ) : Type :=
| vnil : n ~ℕ 0 → vect A n
| vcons : Π (m : ℕ) . A → vect A m → n ~ℕ S m → vect A n
```

Indexed Inductive Types

Inductive vect (A : Type) (n : ℕ) : Type :=

| vnil : $n \sim_{\mathbb{N}} 0 \rightarrow$ vect A n

| vcons : Π (m : ℕ) . A $\rightarrow$ vect A m $\rightarrow$ $n \sim_{\mathbb{N}} S\ m \rightarrow$ vect A n

from e : vect A n $\sim_{Type}$ vect A' n', we obtain

$vnil_1^{\varepsilon} : (n \sim_{\mathbb{N}} 0) \sim_{SProp} (n' \sim_{\mathbb{N}} 0)$

$vcons_1^{\varepsilon} : A \sim_{Type} A'$

$vcons_2^{\varepsilon} : \Pi\ (m : \mathbb{N})\ .\ \text{vect}\ A\ m \sim_{Type} \text{vect}\ A'\ m$

$vcons_3^{\varepsilon} : \Pi\ (m : \mathbb{N})\ .\ (n \sim_{\mathbb{N}} S\ m) \sim_{SProp} (n' \sim_{\mathbb{N}} S\ m)$

Impredicativity

Our sort of strict propositions is impredicative, and supports large elimination for the observational equality

Thus we need to be careful in our implementation: the algorithm used by Lean in a similar setting is non-terminating.

Abel, Coquand '19. Failure of Normalization in Impredicative Type Theory with Proof-Irrelevant Propositional Equality

Impredicativity

Our sort of strict propositions is impredicative, and supports large elimination for the observational equality

Thus we need to be careful in our implementation: the algorithm used by Lean in a similar setting is non-terminating.

$$\bot := \Pi\ (X : \mathrm{SProp})\ .\ X$$

$$\delta := \lambda\ (x : \bot)\ .\ x\ (\bot \to \bot)\ x$$

$$\Omega := \delta\ (\lambda X.\ \mathrm{cast}(\bot \to \bot, X)\ \delta)$$

$$\Omega \rightsquigarrow \Omega \rightsquigarrow \Omega \rightsquigarrow \ldots$$

Abel, Coquand '19. Failure of Normalization in Impredicative Type Theory with Proof-Irrelevant Propositional Equality

Impredicativity

Our sort of strict propositions is impredicative, and supports large elimination for the observational equality

Thus we need to be careful in our implementation: the algorithm used by Lean in a similar setting is non-terminating.

$\bot := \Pi\ (X : SProp)\ .\ X$

$\delta := \lambda\ (x : \bot)\ .\ x\ (\bot \to \bot)\ x$

$\Omega := \delta\ (\lambda X.\ \mathrm{cast}(\bot \to \bot, X)\ \delta)$

$\Omega \rightsquigarrow \Omega \rightsquigarrow \Omega \rightsquigarrow \dots$

However, this is not a problem if we don't reduce irrelevant proofs

Abel, Coquand '19. Failure of Normalization in Impredicative Type Theory with Proof-Irrelevant Propositional Equality

Models in Grothendieck Toposes

Theorem : $\mathrm{CIC}^{\mathrm{obs}}$ has a model in any Grothendieck 1-topos, where the interpretation of the universe hierarchy contains codes for every object of the topos.

Gratzer '22, An inductive-recursive universe generic for small families

Models in Grothendieck Toposes

Theorem : CIC^{obs} has a model in any Grothendieck 1-topos, where the interpretation of the universe hierarchy contains codes for every object of the topos.

Proof sketch : given a hierarchy of strict universes $U_0 < U_1 < U_2 \ldots$ for the topos, we use small induction to build a new hierarchy of universes of codes

$V_i : U_i \to U_{i+1} :=$

| embed : $\prod (X : U_i) . V_i\, X$

| code$\prod$: $\prod (X : U_i)\, (X_\varepsilon : V_i\, X)\, (Y : X \to U_i)\, (Y_\varepsilon : (x : X) \to V_i\, (Y\, x))\, .\, V_i\, (\prod X\, Y)$

| ....

Gratzer '22, An inductive-recursive universe generic for small families

Models in Grothendieck Toposes

Corollary : CIC^{obs} is consistent.

Models in Grothendieck Toposes

Corollary : CIC^{obs} is consistent.

Is CIC^{obs} a reasonable language for 1-toposes?

– It is much more powerful than the theory of elementary toposes: we get not only a natural number object, but also some limited amount of replacement (enough to define $\mathbb{N} + P(\mathbb{N}) + P(P(\mathbb{N})) + \ldots$)

Models in Grothendieck Toposes

Corollary : CIC^{obs} is consistent.

Is CIC^{obs} a reasonable language for 1-toposes?

– It is much more powerful than the theory of elementary toposes: we get not only a natural number object, but also some limited amount of replacement (enough to define $\mathbb{N} + P(\mathbb{N}) + P(P(\mathbb{N})) + \ldots$)

– And yet, we don't get the principle of unique choice:
$(R : A \to B \to \mathrm{SProp}) \times (\Pi\ (a : A)\ .\ \exists!\ (b : B)\ .\ R\ a\ b)$
$\to \Sigma\ (f : A \to B)\ .\ (\Pi\ (a : A)\ .\ R\ a\ (f\ a))$

Normalization Models

Theorem : every well-typed term of CIC^{obs} is normalizing.

Corollary : the typing relation for CIC^{obs} is decidable.

Proof sketch : we build a normalization model in MLTT (formalized in Agda), using Abel et al.'s framework.

The cast operator is fundamentally non-parametric, which implies that we need a proof-irrelevant reducibility predicate.

Unsurprisingly, this prevents us from supporting Prop in our model.

But with a simple trick, we can have SProp!

Normalization Models

Corollary : every integer function that can be defined as a closed term of type $\mathbb{N} \to \mathbb{N}$ in $\mathrm{CIC}^{\mathrm{obs}}$ can also be defined in bare MLTT.

This is connected to the lack of unique choice:
even though we can use impredicativity to show that there exist functional relations that cannot be defined in MLTT, we cannot extract them to terms of type $\mathbb{N} \to \mathbb{N}$

Normalization Models

Corollary : every integer function that can be defined as a closed term of type $\mathbb{N} \to \mathbb{N}$ in MLTT+Univalence can also be defined in bare MLTT.

Proof sketch : we can use the cubical model of Cohen et al. to embed MLTT+Univalence in CIC^{obs}