An Authoritarian Approach to Presheaves

Pierre-Marie Pédrot

INRIA

Birmingham CS Seminar
5th June 2020

It's Time to CIC Ass

CIC, the Calculus of Inductive Constructions.

It's Time to CIC Ass

CIC, the Calculus of Inductive Constructions.

CIC, a very fancy **intuitionistic** **logical system**.

- Not just higher-order logic, not just first-order logic
- First class notion of computation and crazy inductive types

It's Time to CIC Ass

CIC, the Calculus of Inductive Constructions.

CIC, a very fancy **intuitionistic** **logical system**.

- Not just higher-order logic, not just first-order logic
- First class notion of computation and crazy inductive types

CIC, a very powerful **functional** **programming language**.

- Finest types to describe your programs
- No clear phase separation between runtime and compile time

It's Time to CIC Ass

CIC, the Calculus of Inductive Constructions.

CIC, a very fancy **intuitionistic** **logical system**.

- Not just higher-order logic, not just first-order logic
- First class notion of computation and crazy inductive types

CIC, a very powerful **functional** **programming language**.

- Finest types to describe your programs
- No clear phase separation between runtime and compile time

The Pinnacle of the Curry-Howard correspondence

It's Time to CIC Ass

CIC, the Calculus of Inductive Constructions.

CIC, a very fancy **intuitionistic** **logical system**.

- Not just higher-order logic, not just first-order logic
- First class notion of computation and crazy inductive types

CIC, a very powerful **functional** **programming language**.

- Finest types to describe your programs
- No clear phase separation between runtime and compile time

The Pinnacle of the Curry-Howard correspondence

Good Properties We Love

Consistency There is no proof of `False`.

Implementability Type-checking is decidable.

Canonicity Closed integers are indeed integers, i.e

$$\vdash M : \mathbb{N} \quad \text{implies} \quad M \equiv \mathtt{S} \ldots \mathtt{S}\,\mathtt{O}$$

Assuming we have a notion of reduction compatible with conversion:

Normalization Reduction is normalizing

Subject reduction Reduction is compatible with typing

Good Properties We Love

Consistency There is no proof of `False`.

Implementability Type-checking is decidable.

Canonicity Closed integers are indeed integers, i.e

$$\vdash M : \mathbb{N} \quad \text{implies} \quad M \equiv \mathtt{S} \ldots \mathtt{S}\,\mathtt{O}$$

Assuming we have a notion of reduction compatible with conversion:

Normalization Reduction is normalizing

Subject reduction Reduction is compatible with typing

Some of these properties are interdependent

Extending Coq

Our mission: to boldly extend type theory with new principles

Extending Coq

Our mission: to boldly extend type theory with new principles

$\rightsquigarrow$ we need to design models for that.

$\rightsquigarrow$ and ensure they satisfy the good properties.

Today we will focus on a specific family of models...

Extending Coq

Our mission: to boldly extend type theory with new principles

$\rightsquigarrow$ we need to design models for that.

$\rightsquigarrow$ and ensure they satisfy the good properties.

Today we will focus on a specific family of models...

PRESHEAVES!

- Bread and Butter of Model Construction
- Proof-relevant Kripke semantics
- a.k.a. Intuitionistic Forcing

A Bit of Categorical Nonsense

Definition

Let $\mathbb{P}$ be a category. A presheaf over $\mathbb{P}$ is just a functor $\mathbb{P}^{\mathsf{op}} \to \mathbf{Set}$.

(In what follows we will fix the base category $\mathbb{P}$ once and for all.)

A Bit of Categorical Nonsense

Definition

Let $\mathbb{P}$ be a category. A presheaf over $\mathbb{P}$ is just a functor $\mathbb{P}^{\mathsf{op}} \to \mathbf{Set}$.

(In what follows we will fix the base category $\mathbb{P}$ once and for all.)

Theorem

Presheaves with nat. transformations as morphisms form a category $\mathsf{Psh}(\mathbb{P})$.

A Bit of Categorical Nonsense

Definition

Let $\mathbb{P}$ be a category. A presheaf over $\mathbb{P}$ is just a functor $\mathbb{P}^{\mathsf{op}} \rightarrow \mathbf{Set}$.

(In what follows we will fix the base category $\mathbb{P}$ once and for all.)

Theorem

Presheaves with nat. transformations as morphisms form a category $\mathsf{Psh}(\mathbb{P})$.

Actually $\mathsf{Psh}(\mathbb{P})$ is even a **topos**!

A Bit of Categorical Nonsense

Definition

Let $\mathbb{P}$ be a category. A presheaf over $\mathbb{P}$ is just a functor $\mathbb{P}^{\mathsf{op}} \to \mathbf{Set}$.

(In what follows we will fix the base category $\mathbb{P}$ once and for all.)

Theorem

Presheaves with nat. transformations as morphisms form a category $\mathsf{Psh}(\mathbb{P})$.

Actually $\mathsf{Psh}(\mathbb{P})$ is even a **topos**!

Bear with me, we will handwave through this in the next slides.

All Your Base Category Are Belong to Us

What is $\mathsf{Psh}(\mathbb{P})$?

What is $\mathsf{Psh}(\mathbb{P})$?

Objects: A presheaf $(\mathbf{A}, \theta_{\mathbf{A}})$ is given by

- A family of $\mathbb{P}$-indexed sets $\mathbf{A}_p : \mathbf{Set}$
- A family of "restriction morphisms"

$$\theta_{\mathbf{A}} : \Pi\{p, q \in \mathbb{P}\}\,(\alpha \in \mathbb{P}(q, p)).\ \mathbf{A}_p \to \mathbf{A}_q$$

All Your Base Category Are Belong to Us

What is Psh($\mathbb{P}$)?

Objects: A presheaf $(\mathbf{A}, \theta_{\mathbf{A}})$ is given by

- A family of $\mathbb{P}$-indexed sets $\mathbf{A}_p : \mathbf{Set}$
- A family of "restriction morphisms"

$$\theta_{\mathbf{A}} : \Pi\{p, q \in \mathbb{P}\}\,(\alpha \in \mathbb{P}(q, p)).\ \mathbf{A}_p \to \mathbf{A}_q$$

"$\theta_{\mathbf{A}}\,\alpha\,x$ lowers its argument x along $\alpha \in \mathbb{P}(q, p)$"

All Your Base Category Are Belong to Us

What is Psh($\mathbb{P}$)?

Objects: A presheaf $(\mathbf{A}, \theta_{\mathbf{A}})$ is given by

- A family of $\mathbb{P}$-indexed sets $\mathbf{A}_p : \mathbf{Set}$
- A family of "restriction morphisms"

$$\theta_{\mathbf{A}} : \Pi\{p, q \in \mathbb{P}\}\,(\alpha \in \mathbb{P}(q, p)).\ \mathbf{A}_p \to \mathbf{A}_q$$

"$\theta_{\mathbf{A}}\,\alpha\,x$ lowers its argument x along $\alpha \in \mathbb{P}(q, p)$"

s.t. given $x \in \mathbf{A}_p$, $\alpha \in \mathbb{P}(q, p)$ and $\beta \in \mathbb{P}(r, q)$:

$$\theta_{\mathbf{A}}\,\mathtt{id}_p\,x \equiv x \qquad\qquad \theta_{\mathbf{A}}\,(\beta \circ \alpha)\,x \equiv \theta_{\mathbf{A}}\,\beta\,(\theta_{\mathbf{A}}\,\alpha\,x)$$

All Your Base Category Are Belong to Us

What is Psh($\mathbb{P}$)?

Objects: A presheaf $(\mathbf{A}, \theta_\mathbf{A})$ is given by

- A family of $\mathbb{P}$-indexed sets $\mathbf{A}_p : \mathbf{Set}$
- A family of "restriction morphisms"

$$\theta_\mathbf{A} : \Pi\{p, q \in \mathbb{P}\}\,(\alpha \in \mathbb{P}(q, p)).\ \mathbf{A}_p \to \mathbf{A}_q$$

"$\theta_\mathbf{A}\,\alpha\,x$ lowers its argument x along $\alpha \in \mathbb{P}(q, p)$"

s.t. given $x \in \mathbf{A}_p$, $\alpha \in \mathbb{P}(q, p)$ and $\beta \in \mathbb{P}(r, q)$:

$$\theta_\mathbf{A}\,\mathtt{id}_p\,x \equiv x \qquad\qquad \theta_\mathbf{A}\,(\beta \circ \alpha)\,x \equiv \theta_\mathbf{A}\,\beta\,(\theta_\mathbf{A}\,\alpha\,x)$$

"Lowering is compatible with the structure of $\mathbb{P}$"

All Your Base Category Are Belong to Us

What is $\mathsf{Psh}(\mathbb{P})$?

What is $\mathsf{Psh}(\mathbb{P})$?

Morphisms: A morphism from $(\mathbf{A}, \theta_{\mathbf{A}})$ to $(\mathbf{B}, \theta_{\mathbf{B}})$ is given by

- A family of $\mathbb{P}$-index functions $f_p : \mathbf{A}_p \to \mathbf{B}_p$
- which is natural, i.e. given $x \in \mathbf{A}_p$ and $\alpha \in \mathbb{P}(q, p)$

$$\theta_{\mathbf{B}}\,\alpha\,(f_p\,x) \equiv f_q\,(\theta_{\mathbf{A}}\,\alpha\,x)$$

All Your Base Category Are Belong to Us

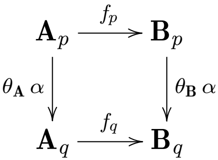
What is Psh($\mathbb{P}$)?

Morphisms: A morphism from $(\mathbf{A}, \theta_\mathbf{A})$ to $(\mathbf{B}, \theta_\mathbf{B})$ is given by

- A family of $\mathbb{P}$-index functions $f_p : \mathbf{A}_p \to \mathbf{B}_p$
- which is natural, i.e. given $x \in \mathbf{A}_p$ and $\alpha \in \mathbb{P}(q, p)$

$$\theta_\mathbf{B}\, \alpha\, (f_p\, x) \equiv f_q\, (\theta_\mathbf{A}\, \alpha\, x)$$

"f is compatible with restriction"

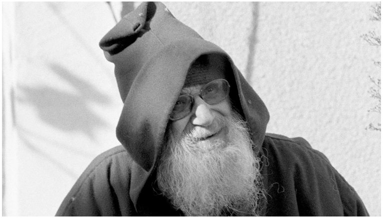


The Wise Speak Only of What They Know

$\mathsf{Psh}(\mathbb{P})$ is a **topos**.

The Wise Speak Only of What They Know

$\mathrm{Psh}(\mathbb{P})$ is a **topos**.

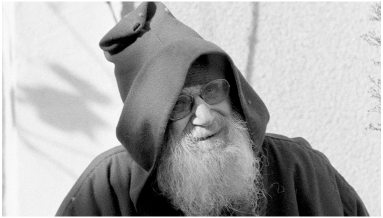


"Speak, friend, and pullback."

Merely a categorical curse word

The Wise Speak Only of What They Know

$\mathsf{Psh}(\mathbb{P})$ is a **topos**.



"Speak, friend, and pullback."

Merely a categorical curse word

For our purposes, that means that

- $\mathsf{Psh}(\mathbb{P})$ is some kind of type theory
- ... in particular, it contains the simply-typed λ-calculus

Who cares about topoi?

Presheaves, Presheaves Everywhere

Who cares about topoi?

Presheaves actually form a model of CIC.

Presheaves, Presheaves Everywhere

Who cares about topoi?

Presheaves actually form a model of CIC.

As usual:

$$\vdash A : \Box \quad \rightsquigarrow \quad [\![A]\!] \in \mathsf{Psh}(\mathbb{P})$$

$$\vdash M : A \quad \rightsquigarrow \quad [M] \in \mathsf{Nat}(1, [\![A]\!])$$

I won't give further details here. One remark though.

Presheaves, Presheaves Everywhere

Who cares about topoi?

Presheaves actually form a model of CIC.

As usual:

$$\vdash A : \square \quad \rightsquigarrow \quad [\![A]\!] \in \mathsf{Psh}(\mathbb{P})$$

$$\vdash M : A \quad \rightsquigarrow \quad [M] \in \mathsf{Nat}(1, [\![A]\!])$$

I won't give further details here. One remark though.

Yet another set-theoretical model!

Cantor's Hell

Let's have a look at the good properties we long for.

Cantor's Hell

Let's have a look at the good properties we long for.

Consistency There is no proof of `False`.

Cantor's Hell

Let's have a look at the good properties we long for.

Consistency There is no proof of `False`. ☺

Cantor's Hell

Let's have a look at the good properties we long for.

Consistency There is no proof of `False`. ☺

Canonicity Closed integers are integers... are they?

$$\vdash M : \mathbb{N} \quad \text{“(C)ZF-implies”} \quad M \equiv \mathtt{S} \ldots \mathtt{S}\,\mathtt{0}$$

Cantor's Hell

Let's have a look at the good properties we long for.

Consistency There is no proof of `False`. ☺

Canonicity Closed integers are integers... are they?

$$\vdash M : \mathbb{N} \quad \text{"(C)ZF-implies"} \quad M \equiv \mathtt{S} \ldots \mathtt{S}\,\mathtt{0}$$ 😕

Implementability Type-checking is **not** decidable.

Cantor's Hell

Let's have a look at the good properties we long for.

Consistency There is no proof of `False`. ☺

Canonicity Closed integers are integers... are they?

$$\vdash M : \mathbb{N} \quad \text{“(C)ZF-implies”} \quad M \equiv \mathtt{S} \ldots \mathtt{S}\,\mathtt{0}$$

Implementability Type-checking is **not** decidable. ☹

Cantor's Hell

Let's have a look at the good properties we long for.

Consistency There is no proof of `False`. ☺

Canonicity Closed integers are integers... are they?

$$\vdash M : \mathbb{N} \quad \text{“(C)ZF-implies”} \quad M \equiv \mathtt{S} \ldots \mathtt{S}\,\mathtt{0}$$

Implementability Type-checking is **not** decidable. ☹

Reduction Never heard of that. What's syntax already?

Cantor's Hell

Let's have a look at the good properties we long for.

Consistency There is no proof of `False`. ☺

Canonicity Closed integers are integers... are they?

$$\vdash M : \mathbb{N} \quad \text{"(C)ZF-implies"} \quad M \equiv \mathtt{S} \ldots \mathtt{S}\,\mathtt{O}$$ 😕

Implementability Type-checking is **not** decidable. ☹

Reduction Never heard of that. What's syntax already? 😱

Cantor's Hell

Let's have a look at the good properties we long for.

Consistency There is no proof of `False`. ☺

Canonicity Closed integers are integers... are they?

$$\vdash M : \mathbb{N} \quad \text{“(C)ZF-implies”} \quad M \equiv \mathtt{S} \ldots \mathtt{S}\,\mathtt{0}$$

Implementability Type-checking is **not** decidable. ☹

Reduction Never heard of that. What's syntax already? 😱

⇝ Exeunt **Normalization** and **Subject reduction**.

Cantor's Hell

Let's have a look at the good properties we long for.

Consistency There is no proof of `False`. ☺

Canonicity Closed integers are integers... are they?

$$\vdash M : \mathbb{N} \quad \text{“(C)ZF-implies”} \quad M \equiv \mathtt{S} \ldots \mathtt{S}\,\mathtt{0}$$

Implementability Type-checking is **not** decidable. ☹

Reduction Never heard of that. What's syntax already?

$\rightsquigarrow$ Exeunt **Normalization** and **Subject reduction**.

Phenomenological Law

Set-theoretical models suck.

Syntactic Models

What is a model?

- Takes syntax as input.
- Interprets it into some low-level language.
- Must preserve the meaning of the source.
- Refines the behaviour of under-specified structures.

What is a model?

- Takes syntax as input.
- Interprets it into some low-level language.
- Must preserve the meaning of the source.
- Refines the behaviour of under-specified structures.

This looks suspiciously familiar...

What is a model?

- Takes syntax as input.
- Interprets it into some low-level language.
- Must preserve the meaning of the source.
- Refines the behaviour of under-specified structures.

This looks suspiciously familiar...

"By Jove, this is a *compiler*!"

What is a model?

- Takes syntax as input.
- Interprets it into some low-level language.
- Must preserve the meaning of the source.
- Refines the behaviour of under-specified structures.

This looks suspiciously familiar...

“By Jove, this is a *compiler*!”

This is a folklore in the Curry-Howard community.

On Curry-Howard Poetry

Usual models are more like **interpreters**.

No separation between $\left\{\begin{array}{l}\text{implementation}\\\text{meta}\end{array}\right\}$ vs. $\left\{\begin{array}{l}\text{host}\\\text{target}\end{array}\right\}$ languages

$$\vdash_{\mathcal{S}} A \quad \xrightarrow{\text{meta}} \quad \vDash_{\mathcal{M}} A$$

Notably, $\vDash_{\mathcal{M}}$ lives in the semantical world.

Example: NbE, external realizability.

On Curry-Howard Poetry

Syntactic models are proper **compilers**.

Target and meta languages are **clearly distinct**.

$$\vdash_{\mathcal{S}} A \quad \xrightarrow{\text{meta}} \quad \vdash_{\mathcal{T}} [\![A]\!]$$

Now $\vdash_{\mathcal{T}}$ is pure syntax, only soundness lives in the meta!

Example: CPS translation, internal realizability.

On Curry-Howard Poetry

Syntactic models are proper **compilers**.

Target and meta languages are **clearly distinct**.

$$\vdash_{\mathcal{S}} A \quad \xrightarrow{\text{meta}} \quad \vdash_{\mathcal{T}} [\![A]\!]$$

Now $\vdash_{\mathcal{T}}$ is pure syntax, only soundness lives in the meta!

Example: CPS translation, internal realizability.

We will be interested in instances where $\mathcal{S}, \mathcal{T}$ are type theories.

Syntactic Models, Details

Step 0: Pick two type theories, a source $\mathcal{S}$ and a target $\mathcal{T}$.

Typically both theories are CIC.

Syntactic Models, Details

Step 0: Pick two type theories, a source $\mathcal{S}$ and a target $\mathcal{T}$.

Typically both theories are CIC.

Step 1: Define $[\cdot]$ on the syntax of $\mathcal{S}$ and derive $[\![\cdot]\!]$ from it s.t.

$$\vdash_{\mathcal{S}} M : A \qquad \text{implies} \qquad \vdash_{\mathcal{T}} [M] : [\![A]\!]$$

Proving this is the one appeal to a (weak) meta.

Syntactic Models, Details

Step 0: Pick two type theories, a source $\mathcal{S}$ and a target $\mathcal{T}$.

Typically both theories are CIC.

Step 1: Define $[\cdot]$ on the syntax of $\mathcal{S}$ and derive $[\![\cdot]\!]$ from it s.t.

$$\vdash_{\mathcal{S}} M : A \qquad \text{implies} \qquad \vdash_{\mathcal{T}} [M] : [\![A]\!]$$

Proving this is the one appeal to a (weak) meta.

Step 2: Flip views and actually pose

$$\vdash_{\mathcal{S}} M : A \qquad := \qquad \vdash_{\mathcal{T}} [M] : [\![A]\!]$$

Syntactic Models, Details

Step 0: Pick two type theories, a source $\mathcal{S}$ and a target $\mathcal{T}$.

Typically both theories are CIC.

Step 1: Define $[\cdot]$ on the syntax of $\mathcal{S}$ and derive $[\![\cdot]\!]$ from it s.t.

$$\vdash_{\mathcal{S}} M : A \qquad \text{implies} \qquad \vdash_{\mathcal{T}} [M] : [\![A]\!]$$

Proving this is the one appeal to a (weak) meta.

Step 2: Flip views and actually pose

$$\vdash_{\mathcal{S}} M : A \qquad := \qquad \vdash_{\mathcal{T}} [M] : [\![A]\!]$$

Step 3: Expand $\mathcal{S}$ by going down to the $\mathcal{T}$ *assembly language*, implementing new terms through the $[\cdot]$ translation.

Why Syntactic Models?

Obviously, that's subtle.

- The translation $[\cdot]$ must preserve typing (not easy)
- In particular, it must preserve conversion (even worse)

Why Syntactic Models?

Obviously, that's subtle.

- The translation $[\cdot]$ must preserve typing (not easy)
- In particular, it must preserve conversion (even worse)

Yet, a lot of nice consequences.

- Does not require non-type-theoretical foundations (*monism*)
- Can be implemented in Coq (*software monism*)
- Easy to show (relative) consistency, look at $[\![\texttt{False}]\!]$
- Inherit properties from CIC: computationality, decidability, **implementation**...

On Syntactic Models

They were first introduced by Martin Hofmann in his PhD (1997).

... then somewhat neglected.

On Syntactic Models

They were first introduced by Martin Hofmann in his PhD (1997).

... then somewhat neglected.

At Gallinette, we have been using them successfully in the recent years

- For effectful type theories mostly
- But the one model that **originally** sparked our interest was...

On Syntactic Models

They were first introduced by Martin Hofmann in his PhD (1997).

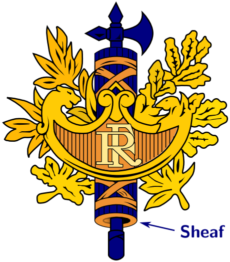
... then somewhat neglected.

At Gallinette, we have been using them successfully in the recent years

- For effectful type theories mostly
- But the one model that **originally** sparked our interest was...

PRESHEAVES!

“Is it possible to see the presheaf construction as a syntactic model?”

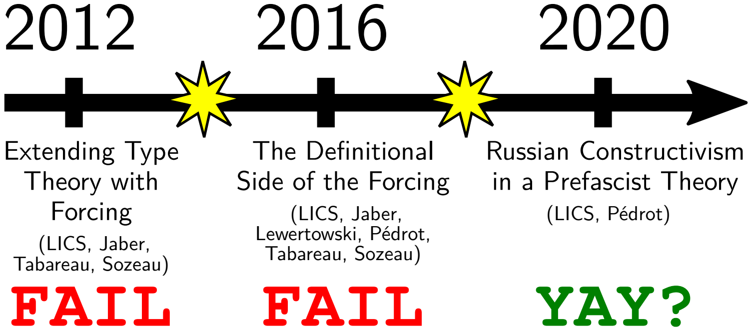


French Coat of Arms

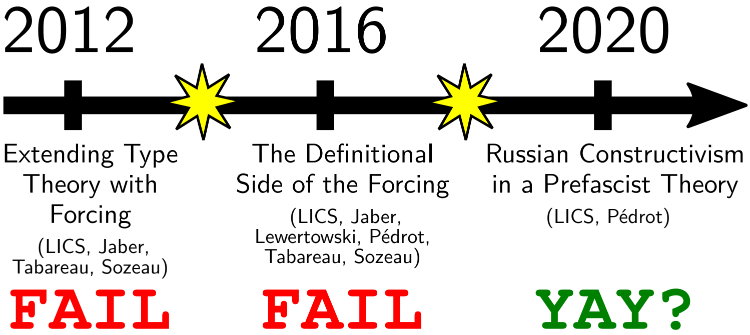
Persevere Diabolicum

Why the hell am I talking about syntactic presheaves today?

Why the hell am I talking about syntactic presheaves today?



Why the hell am I talking about syntactic presheaves today?



It is the journey, not the destination

2012

(We were warned.)

Syntactic Presheaves, 2012 Edition

"A presheaf is just a functor $\mathbb{P}^{\mathsf{op}} \rightarrow \mathbf{Set}$."

Syntactic Presheaves, 2012 Edition

"A presheaf is just a functor $\mathbb{P}^{\mathsf{op}} \rightarrow \mathbf{Set}$."

"Hold my beer!"

Syntactic Presheaves, 2012 Edition

"A presheaf is just a functor $\mathbb{P}^{\mathrm{op}} \to \mathbf{Set}$."

"Hold my beer!"

Replace $\mathbf{Set}$ everywhere with CIC.

Syntactic Presheaves, 2012 Edition

"A presheaf is just a functor $\mathbb{P}^{\mathrm{op}} \to \mathbf{Set}$."

"Hold my beer!"

Replace $\mathbf{Set}$ everywhere with CIC.

What could possibly go wrong?

Close Encounters of the Third Type

Replace $\mathbf{Set}$ everywhere with CIC.

Close Encounters of the Third Type

Replace **Set** everywhere with CIC.

$$\texttt{Cat} : \square \quad := \quad \left\{ \begin{array}{l} \mathbb{P} : \square \\ \leq : \mathbb{P} \rightarrow \mathbb{P} \rightarrow \square \\ \texttt{id} : \Pi p.\, p \leq p \\ \circ : \Pi p\, q\, r.\, p \leq q \rightarrow q \leq r \rightarrow p \leq r \\ \texttt{eqn} : \ldots; \end{array} \right\}$$

$$\texttt{Psh} : \square \quad := \quad \left\{ \begin{array}{l} \mathbf{A} : \mathbb{P} \rightarrow \square \\ \theta_{\mathbf{A}} : \Pi (p\, q : \mathbb{P})\, (\alpha : q \leq p).\, \mathbf{A}_p \rightarrow \mathbf{A}_q \\ \texttt{eqn} : \ldots; \end{array} \right\}$$

$$\texttt{El}\ (\mathbf{A}, \theta_{\mathbf{A}}, \mathtt{e}) : \square \quad := \quad \left\{ \begin{array}{l} \texttt{el} : \Pi (p : \mathbb{P}).\, \mathbf{A}\ p \\ \texttt{eqn} : \ldots; \end{array} \right\}$$

Close Encounters of the Third Type

Replace **Set** everywhere with CIC.

$$\texttt{Cat} : \Box \quad := \quad \left\{ \begin{array}{l} \mathbb{P} : \Box \\ \leq : \mathbb{P} \to \mathbb{P} \to \Box \\ \texttt{id} : \Pi p.\, p \leq p \\ \circ : \Pi p\, q\, r.\, p \leq q \to q \leq r \to p \leq r \\ \texttt{eqn} : \ldots; \end{array} \right\}$$

$$\texttt{Psh} : \Box \quad := \quad \left\{ \begin{array}{l} \mathbf{A} : \mathbb{P} \to \Box \\ \theta_{\mathbf{A}} : \Pi (p\, q : \mathbb{P})\, (\alpha : q \leq p).\, \mathbf{A}_p \to \mathbf{A}_q \\ \texttt{eqn} : \ldots; \end{array} \right\}$$

$$\texttt{El}\ (\mathbf{A}, \theta_{\mathbf{A}}, \mathtt{e}) : \Box \quad := \quad \left\{ \begin{array}{l} \texttt{el} : \Pi (p : \mathbb{P}).\, \mathbf{A}\ p \\ \texttt{eqn} : \ldots; \end{array} \right\}$$

And voilá, the Great Typification is an utter success!

Equality is Too Serious a Matter

This **almost** works...

Equality is Too Serious a Matter

This **almost** works...

... except that equations are propositional !!!

$$\texttt{El}\ (\mathbf{A}, \theta_{\mathbf{A}}, \mathtt{e}) : \square \quad := \quad \left\{ \begin{array}{l} \mathtt{el} : \Pi(p : \mathbb{P}).\ \mathbf{A}\ p \\ \mathtt{eqn} : \ldots ; \end{array} \right\}$$

$$\vdash_{\mathsf{CIC}} M \equiv N \quad \not\longrightarrow \quad \vdash [M] \equiv [N]$$

$$\vdash_{\mathsf{CIC}} M \equiv N \quad \longrightarrow \quad \vdash e : [M] = [N]$$

Equality is Too Serious a Matter

This **almost** works...

... except that equations are propositional !!!

$$\mathtt{El}\ (\mathbf{A}, \theta_{\mathbf{A}}, \mathtt{e}) : \Box \quad := \quad \left\{ \begin{array}{l} \mathtt{el} : \Pi(p : \mathbb{P}).\, \mathbf{A}\ p \\ \mathtt{eqn} : \ldots ; \end{array} \right\}$$

$$\begin{array}{lcl} \vdash_{\mathsf{CIC}} M \equiv N & \not\longrightarrow & \vdash [M] \equiv [N] \\ \vdash_{\mathsf{CIC}} M \equiv N & \longrightarrow & \vdash e : [M] = [N] \end{array}$$

You need to introduce rewriting everywhere

"The Coherence Hell"

Equality is Too Serious a Matter

This **almost** works...

... except that equations are propositional !!!

$$\texttt{El}\ (\mathbf{A}, \theta_{\mathbf{A}}, \mathtt{e}) : \square \quad := \quad \left\{ \begin{array}{l} \texttt{el} : \Pi(p : \mathbb{P}).\ \mathbf{A}\ p \\ \texttt{eqn} : \ldots ; \end{array} \right\}$$

$$\vdash_{\mathsf{CIC}} M \equiv N \quad \not\longrightarrow \quad \vdash [M] \equiv [N]$$
$$\vdash_{\mathsf{CIC}} M \equiv N \quad \longrightarrow \quad \vdash e : [M] = [N]$$

You need to introduce rewriting everywhere

"The Coherence Hell"

Thus the target theory must be EXTENSIONAL

That Was Not My Intension

Extensional Type Theory (ETT) is defined by *Santa Claus conversion*.

$$\frac{\Gamma \vdash e : M = N}{\Gamma \vdash M \equiv N}$$

That Was Not My Intension

Extensional Type Theory (ETT) is defined by *Santa Claus conversion*.

$$\frac{\Gamma \vdash e : M = N}{\Gamma \vdash M \equiv N}$$

- Arguably better than ZFC ("constructive")

That Was Not My Intension

Extensional Type Theory (ETT) is defined by *Santa Claus conversion*.

$$\frac{\Gamma \vdash e : M = N}{\Gamma \vdash M \equiv N}$$

- Arguably better than ZFC (“constructive”)
- ... but undecidable type checking
- ... no computation, e.g. β-reduction is undecidable
- See Théo Winterhalter’s soon to be defended PhD for more horrors

That Was Not My Intension

Extensional Type Theory (ETT) is defined by *Santa Claus conversion*.

$$\frac{\Gamma \vdash e : M = N}{\Gamma \vdash M \equiv N}$$

- Arguably better than ZFC ("constructive")
- ... but undecidable type checking
- ... no computation, e.g. β-reduction is undecidable
- See Théo Winterhalter's soon to be defended PhD for more horrors

No True Scotsman

Syntactic models into ETT are not really syntactic models†.

That Was Not My Intension

No True Scotsman

Syntactic models into ETT are not really syntactic models†.

(†) To be more precise, I believe that ETT is not really a type theory.

2016

(Make conversion great again, and break everything else.)

Squaring the Circle

(Me to Guilhem, Nicolas and Matthieu, some time before defending PhD.)

— You people are doing it wrong. It cannot work!

Squaring the Circle

(Me to Guilhem, Nicolas and Matthieu, some time before defending PhD.)

— You people are doing it wrong. It cannot work!

— Why?

Squaring the Circle

(Me to Guilhem, Nicolas and Matthieu, some time before defending PhD.)

— You people are doing it wrong. It cannot work!

— Why?

— Because presheaves are *call-by-value*!

Squaring the Circle

(Me to Guilhem, Nicolas and Matthieu, some time before defending PhD.)

— You people are doing it wrong. It cannot work!

— Why?

— Because presheaves are *call-by-value*!

... and you're trying to intepret a *call-by-name* language!

(Me to Guilhem, Nicolas and Matthieu, some time before defending PhD.)

— You people are doing it wrong. It cannot work!

— Why?

— Because presheaves are *call-by-value*!

... and you're trying to intepret a *call-by-name* language!

— What on earth does that even mean?

This is the Left Adjoint, Right?

CBPV is a nice framework to study effects.

This is the Left Adjoint, Right?

CBPV is a nice framework to study effects.

Yet I won't present it here because it's Birmingham.

This is the Left Adjoint, Right?

CBPV is a nice framework to study effects.

Yet I won't present it here because it's Birmingham.

Theorem (Somewhere inside PBL's humongous PhD)

Kripke models factorize through CBPV*.*

This is the Left Adjoint, Right?

CBPV is a nice framework to study effects.

Yet I won't present it here because it's Birmingham.

Theorem (Somewhere inside PBL's humongous PhD)

Kripke models factorize through CBPV*.*

$$
\begin{array}{cccc}
X & \text{computation type} & \mapsto & [\![X]\!]^{\mathtt{c}} : |\mathbb{P}| \to \mathbf{Set} \\
A & \text{value type} & \mapsto & [\![A]\!]^{\mathtt{v}} : \mathsf{Fun}(\mathbb{P}^{op}, \mathbf{Set})
\end{array}
$$

$$
\begin{array}{ccl}
[\![A \to X]\!]^{\mathtt{c}}_p & := & [\![A]\!]^{\mathtt{v}}_p \to [\![X]\!]^{\mathtt{c}}_p \\
[\![\mathcal{F}\ A]\!]^{\mathtt{c}}_p & := & |[\![A]\!]^{\mathtt{v}}_p|
\end{array}
$$

$$
[\![\mathcal{U}\ X]\!]^{\mathtt{v}}_p \quad := \quad \Pi(q : \mathbb{P})(\alpha : q \leq p).\, [\![X]\!]^{\mathtt{c}}_q \qquad \text{(free functoriality)}
$$

$$
\theta_{[\![\mathcal{U}\ X]\!]^{\mathtt{v}}}\ (\alpha : q \leq p)(x : [\![\mathcal{U}\ X]\!]^{\mathtt{v}}_p) := \lambda(r : \mathbb{P})(\beta : r \leq q).\, x\ r\ (\alpha \circ \beta)
$$

More Than One Way to Do It

Theorem

Kripke models factorize through CBPV.

Canonical embeddings of λ-calculus into CBPV:

CBN	$(\sigma \to \tau)^{\mathsf{N}}$	$:=$	$\mathcal{U}\,\sigma^{\mathsf{N}} \to \tau^{\mathsf{N}}$	(a computation type)
CBV	$(\sigma \to \tau)^{\mathsf{V}}$	$:=$	$\mathcal{U}\,(\sigma^{\mathsf{V}} \to \mathcal{F}\,\tau^{\mathsf{V}})$	(a value type)

More Than One Way to Do It

Theorem

Kripke models factorize through CBPV.

Canonical embeddings of λ-calculus into CBPV:

CBN	$(\sigma \to \tau)^{\mathsf{N}}$	$:=$	$\mathcal{U}\,\sigma^{\mathsf{N}} \to \tau^{\mathsf{N}}$	(a computation type)
CBV	$(\sigma \to \tau)^{\mathsf{V}}$	$:=$	$\mathcal{U}\,(\sigma^{\mathsf{V}} \to \mathcal{F}\,\tau^{\mathsf{V}})$	(a value type)

Thus, composing the CBV embedding with the "Kripke" interpretation:

$$[\![(\sigma \to \tau)^{\mathsf{V}}]\!]^{\mathtt{v}}_p := \Pi(q : \mathbb{P})(\alpha : q \leq p).\,[\![\sigma^{\mathsf{V}}]\!]^{\mathtt{v}}_q \to [\![\tau^{\mathsf{V}}]\!]^{\mathtt{v}}_q$$

More Than One Way to Do It

Theorem

Kripke models factorize through CBPV.

Canonical embeddings of λ-calculus into CBPV:

CBN	$(\sigma \to \tau)^{\mathsf{N}}$	$:=$	$\mathcal{U}\,\sigma^{\mathsf{N}} \to \tau^{\mathsf{N}}$	(a computation type)
CBV	$(\sigma \to \tau)^{\mathsf{V}}$	$:=$	$\mathcal{U}\,(\sigma^{\mathsf{V}} \to \mathcal{F}\,\tau^{\mathsf{V}})$	(a value type)

Thus, composing the CBV embedding with the "Kripke" interpretation:

$$[\![(\sigma \to \tau)^{\mathsf{V}}]\!]^{\mathtt{v}}_p := \Pi(q : \mathbb{P})(\alpha : q \le p).\, [\![\sigma^{\mathsf{V}}]\!]^{\mathtt{v}}_q \to [\![\tau^{\mathsf{V}}]\!]^{\mathtt{v}}_q$$

This is the presheaf interpretation of arrows! (up to naturality)**

Le Clash

Presheaves are *call-by-value*!

Le Clash

Presheaves are *call-by-value*!

In particular, they only satisfy the CBV equational theory generated by

$$(\lambda x.\, t)\ V \equiv_{\beta v} t\{x := V\}$$

because

$$t \equiv_{\beta v} u \quad \longrightarrow \quad t^{\mathsf{V}} \equiv_{\mathsf{CBPV}} u^{\mathsf{V}} \quad \longrightarrow \quad [t^{\mathsf{V}}]_p \equiv_{\mathcal{T}} [u^{\mathsf{V}}]_p$$

Le Clash

Presheaves are *call-by-value*!

In particular, they only satisfy the CBV equational theory generated by

$$(\lambda x.\, t)\ V \equiv_{\beta v} t\{x := V\}$$

because

$$t \equiv_{\beta v} u \quad \longrightarrow \quad t^{\mathsf{V}} \equiv_{\mathsf{CBPV}} u^{\mathsf{V}} \quad \longrightarrow \quad [t^{\mathsf{V}}]_p \equiv_{\mathcal{T}} [u^{\mathsf{V}}]_p$$

Type theory is *call-by-name*!

Le Clash

Presheaves are *call-by-value*!

In particular, they only satisfy the CBV equational theory generated by

$$(\lambda x.\, t)\ V \equiv_{\beta v} t\{x := V\}$$

because

$$t \equiv_{\beta v} u \quad \longrightarrow \quad t^{\mathsf{V}} \equiv_{\mathsf{CBPV}} u^{\mathsf{V}} \quad \longrightarrow \quad [t^{\mathsf{V}}]_p \equiv_{\mathcal{T}} [u^{\mathsf{V}}]_p$$

Type theory is *call-by-name*!

$$\frac{\Gamma \vdash M : B \qquad \Gamma \vdash A \equiv_{\beta} B}{\Gamma \vdash M : A}\ (\mathsf{Conv})$$

Le Clash

Presheaves are *call-by-value*!

In particular, they only satisfy the CBV equational theory generated by

$$(\lambda x.\, t)\ V \equiv_{\beta v} t\{x := V\}$$

because

$$t \equiv_{\beta v} u \quad \longrightarrow \quad t^{\mathsf{V}} \equiv_{\mathsf{CBPV}} u^{\mathsf{V}} \quad \longrightarrow \quad [t^{\mathsf{V}}]_p \equiv_{\mathcal{T}} [u^{\mathsf{V}}]_p$$

Type theory is *call-by-name*!

$$\frac{\Gamma \vdash M : B \qquad \Gamma \vdash A \equiv_{\beta} B}{\Gamma \vdash M : A}\ (\mathsf{Conv})$$

Folklore

Call-by-name is not call-by-value!

If There is No Solution, There is No Problem

Easy solution! Pick the CBN decomposition instead.

$$[\![(\sigma \to \tau)^{\mathsf{N}}]\!]_p^{\mathsf{c}} := (\Pi(q : \mathbb{P})(\alpha : q \leq p).\, [\![\sigma^{\mathsf{N}}]\!]_q^{\mathsf{c}}) \to [\![\tau^{\mathsf{N}}]\!]_p^{\mathsf{c}}$$

If There is No Solution, There is No Problem

Easy solution! Pick the CBN decomposition instead.

$$[\![(\sigma \to \tau)^{\mathsf{N}}]\!]_p^{\mathsf{c}} := (\Pi(q : \mathbb{P})(\alpha : q \leq p).\, [\![\sigma^{\mathsf{N}}]\!]_q^{\mathsf{c}}) \to [\![\tau^{\mathsf{N}}]\!]_p^{\mathsf{c}}$$

This adapts straightforwardly to the dependently-typed setting.

If There is No Solution, There is No Problem

Easy solution! Pick the CBN decomposition instead.

$$[\![(\sigma \to \tau)^{\mathsf{N}}]\!]_p^{\mathsf{c}} := (\Pi(q : \mathbb{P})(\alpha : q \leq p).\, [\![\sigma^{\mathsf{N}}]\!]_q^{\mathsf{c}}) \to [\![\tau^{\mathsf{N}}]\!]_p^{\mathsf{c}}$$

This adapts straightforwardly to the dependently-typed setting.

Theorem (Jaber & al. 2016)

There is a syntactic presheaf model of CC^ω *into* CIC.

where CC^ω is CIC without inductive types.

If There is No Solution, There is No Problem

Easy solution! Pick the CBN decomposition instead.

$$[\![(\sigma \to \tau)^{\mathsf{N}}]\!]_p^{\mathsf{c}} := (\Pi(q : \mathbb{P})(\alpha : q \leq p).\, [\![\sigma^{\mathsf{N}}]\!]_q^{\mathsf{c}}) \to [\![\tau^{\mathsf{N}}]\!]_p^{\mathsf{c}}$$

This adapts straightforwardly to the dependently-typed setting.

Theorem (Jaber & al. 2016)

There is a syntactic presheaf model of CC^ω *into* CIC.

where CC^ω is CIC without inductive types.

$$\begin{array}{rcl}
\vdash_{\mathsf{CC}^\omega} A : \Box & \longrightarrow & p : \mathbb{P} \vdash_{\mathsf{CIC}} [A]_p : \Pi(q : \mathbb{P})(\alpha : q \leq p).\, \Box \\
\vdash_{\mathsf{CC}^\omega} M : A & \longrightarrow & p : \mathbb{P} \vdash_{\mathsf{CIC}} [M]_p : [A]_p\ p\ \mathtt{id}_p \\
\vdash_{\mathsf{CC}^\omega} M \equiv N & \longrightarrow & p : \mathbb{P} \vdash_{\mathsf{CIC}} [M]_p \equiv [N]_p
\end{array}$$

Robbing Peter to Pay Paul

There is a syntactic presheaf model of CC^{ω} into CIC.

Robbing Peter to Pay Paul

There is a syntactic presheaf model of CC^{ω} into CIC.

"What about inductive types?"

Robbing Peter to Pay Paul

There is a syntactic presheaf model of CC^ω into CIC.

"What about inductive types?"

The model disproves dependent elimination!

in general $\quad \not\vdash \Pi(P : \mathbb{B} \rightarrow \square).\, P\ \mathtt{tt} \rightarrow P\ \mathtt{ff} \rightarrow \Pi(b : \mathbb{B}).\, P\ b$

because there are **non-standard** booleans.

Robbing Peter to Pay Paul

There is a syntactic presheaf model of CC^ω into CIC.

"What about inductive types?"

The model disproves dependent elimination!

in general $\quad \not\vdash \Pi(P : \mathbb{B} \to \square).\, P\ \mathtt{tt} \to P\ \mathtt{ff} \to \Pi(b : \mathbb{B}).\, P\ b$

because there are **non-standard** booleans.

It only validates it for specific predicates P

$$\vdash P\ \mathtt{tt} \to P\ \mathtt{ff} \to \Pi(b : \mathbb{B}).\, P\ b \qquad \text{if } P \text{ strict}$$

- Any predicate P can be made strict canonically (using storage operators)
- In presence of dep. elim. strictification is the identity

Robbing Peter to Pay Paul

In retrospective, this is not surprising.

Robbing Peter to Pay Paul

In retrospective, this is not surprising.

The Kripke translation introduces an effect!

It can be seen as a monotonic variant of the reader effect.

Robbing Peter to Pay Paul

In retrospective, this is not surprising.

The Kripke translation introduces an effect!

It can be seen as a monotonic variant of the reader effect.

The Proverbial Paul

CBPV Folklore

- In effectful CBV, functions are not functions. (no substitution)
- In effectful CBN, inductive types are not inductive types. (no dep. elim.)

Conclusion of the Episode II

Good News

This is one of the first reasonable example of dependent effects.

Conclusion of the Episode II

Good News

This is one of the first reasonable example of dependent effects.

Bad News

We still don't have a syntactic presheaf model.

Interlude

Interlude

In the meantime we worked quite a bit on effectful type theories

- Weaning translation
- Baclofen Type Theory
- Exceptional Type Theory
- ...

Interlude

In the meantime we worked quite a bit on effectful type theories

- Weaning translation
- Baclofen Type Theory
- Exceptional Type Theory
- ...

This helped us understand what we first missed!

Values Are Not What They Once Were

Categorical presheaves form a model of the whole λ-calculus.

... in particular, it does interpret full β-conversion (although extensionally).

Values Are Not What They Once Were

Categorical presheaves form a model of the whole λ-calculus.

... in particular, it does interpret full β-conversion (although extensionally).

This is because of the **naturality** requirement on functions.

$$[\![A \to B]\!]_p \quad := \quad f : \Pi(q \le p).\, [\![A]\!]_q \to [\![B]\!]_q \quad \text{s.t.} \quad \begin{array}{ccc} [\![A]\!]_q & \xrightarrow{f_q\ \alpha} & [\![B]\!]_q \\ {\scriptstyle \theta_{\mathbf{A}}\ \beta}\downarrow & & \downarrow{\scriptstyle \theta_{\mathbf{B}}\ \beta} \\ [\![A]\!]_r & \xrightarrow{f_r\ (\alpha \circ \beta)} & [\![B]\!]_r \end{array}$$

Values Are Not What They Once Were

Categorical presheaves form a model of the whole λ-calculus.

... in particular, it does interpret full β-conversion (although extensionally).

This is because of the **naturality** requirement on functions.

$$[\![A \to B]\!]_p \quad := \quad f : \Pi(q \leq p).\, [\![A]\!]_q \to [\![B]\!]_q \quad \text{s.t.} \quad \begin{array}{ccc} [\![A]\!]_q & \xrightarrow{f_q\ \alpha} & [\![B]\!]_q \\ {\scriptstyle \theta_{\mathbf{A}}\ \beta}\downarrow & & \downarrow{\scriptstyle \theta_{\mathbf{B}}\ \beta} \\ [\![A]\!]_r & \xrightarrow{f_r\ (\alpha\circ\beta)} & [\![B]\!]_r \end{array}$$

- We do not have an equivalent in our CBN interpretation
- Isn't this some ad-hoc trick?

Completely Unrelated Slide

Consider an effectful CBV λ-calculus.

Definition (Führmann '99)

A term $t : A$ is said to be **thunkable** if it satisfies the equation

$$\texttt{let}\ x := t\ \texttt{in}\ \lambda().\, x \quad \equiv \quad \lambda().\, t$$

Completely Unrelated Slide

Consider an effectful CBV λ-calculus.

Definition (Führmann '99)

A term $t : A$ is said to be **thunkable** if it satisfies the equation

$$\texttt{let}\ x := t\ \texttt{in}\ \lambda().\,x \quad \equiv \quad \lambda().\,t$$

- Thunkability intuitively captures "purity"
- It does so generically, i.e. does not depend on effect considered
- In a pure language, all terms are thunkable

Completely Unrelated Slide

Consider an effectful CBV λ-calculus.

Definition (Führmann '99)

A term $t : A$ is said to be **thunkable** if it satisfies the equation

$$\texttt{let}\ x := t\ \texttt{in}\ \lambda().\, x \quad \equiv \quad \lambda().\, t$$

- Thunkability intuitively captures "purity"
- It does so generically, i.e. does not depend on effect considered
- In a pure language, all terms are thunkable

Theorem (Folklore Realizability)

The sublanguage of hereditarily thunkable terms satisfies full β-conversion.

$$f \Vdash A \to B \quad := \quad \forall u. \quad u \Vdash A \quad \longrightarrow \quad f\ u\ \mathsf{thk} \quad \wedge \quad f\ u \Vdash B$$

Presheaves Are (Pure) Call-By-Value!

Theorem

A term $x : A \vdash t : B$ *is thunkable in the Kripke semantics iff* $[t]_p$ *is natural.*

Presheaves Are (Pure) Call-By-Value!

Theorem

A term $x : A \vdash t : B$ is thunkable in the Kripke semantics iff $[t]_p$ is natural.

Proof.

Literal unfolding of the definitions. □

Presheaves Are (Pure) Call-By-Value!

Theorem

A term $x : A \vdash t : B$ *is thunkable in the Kripke semantics iff* $[t]_p$ *is natural.*

Proof.

Literal unfolding of the definitions. □

Psh($\mathbb{P}$) is the "pure" subcategory of an effectful CBV language!

- This is a systematic construction.
- Unfortunately it relies on extensionality.
- We *know* how to port this to the CBN setting **intensionally**.

Presheaves Are (Pure) Call-By-Value!

Theorem

A term $x : A \vdash t : B$ is thunkable in the Kripke semantics iff $[t]_p$ is natural.

Proof.

Literal unfolding of the definitions. □

Psh($\mathbb{P}$) is the "pure" subcategory of an effectful CBV language!

- This is a systematic construction.
- Unfortunately it relies on extensionality.
- We *know* how to port this to the CBN setting **intensionally**.

The CBN equivalent is parametricity!

Syntactic Models For Free

Bernardy-Lasson '11

There is a well-known parametricity interpretation for type theory

$$\Gamma \vdash_{\mathsf{CIC}} M : A \quad \longrightarrow \quad [\![\Gamma]\!]_\varepsilon \vdash_{\mathsf{CIC}} [M]_\varepsilon : [\![A]\!]_\varepsilon\ M$$

where $\quad [\![\cdot]\!]_\varepsilon := \cdot \quad$ and $\quad [\![\Gamma, x : A]\!]_\varepsilon := [\![\Gamma]\!]_\varepsilon, x : A, x_\varepsilon : [\![A]\!]_\varepsilon\ x$

Syntactic Models For Free

Bernardy-Lasson '11

There is a well-known parametricity interpretation for type theory

$$\Gamma \vdash_{\mathsf{CIC}} M : A \quad \longrightarrow \quad [\![\Gamma]\!]_\varepsilon \vdash_{\mathsf{CIC}} [M]_\varepsilon : [\![A]\!]_\varepsilon \; M$$

where $\quad [\![\cdot]\!]_\varepsilon := \cdot \quad$ and $\quad [\![\Gamma, x : A]\!]_\varepsilon := [\![\Gamma]\!]_\varepsilon, x : A, x_\varepsilon : [\![A]\!]_\varepsilon \; x$

Turns out it is a syntactic model!

Syntactic Models For Free

Bernardy-Lasson '11

There is a well-known parametricity interpretation for type theory

$$\Gamma \vdash_{\mathsf{CIC}} M : A \quad \longrightarrow \quad [\![\Gamma]\!]_\varepsilon \vdash_{\mathsf{CIC}} [M]_\varepsilon : [\![A]\!]_\varepsilon \ M$$

where $\quad [\![\cdot]\!]_\varepsilon := \cdot \quad$ and $\quad [\![\Gamma, x : A]\!]_\varepsilon := [\![\Gamma]\!]_\varepsilon, x : A, x_\varepsilon : [\![A]\!]_\varepsilon \ x$

Turns out it is a syntactic model!

It is a special case of a more general **internal realizability** interpretation.

$$[\![A]\!]_\varepsilon \ M \quad := \quad M \Vdash A$$

Syntactic Models For Free

Bernardy-Lasson '11

There is a well-known parametricity interpretation for type theory

$$\Gamma \vdash_{\mathsf{CIC}} M : A \quad \longrightarrow \quad [\![\Gamma]\!]_\varepsilon \vdash_{\mathsf{CIC}} [M]_\varepsilon : [\![A]\!]_\varepsilon\ M$$

where $\quad [\![\cdot]\!]_\varepsilon := \cdot \quad$ and $\quad [\![\Gamma, x : A]\!]_\varepsilon := [\![\Gamma]\!]_\varepsilon, x : A, x_\varepsilon : [\![A]\!]_\varepsilon\ x$

Turns out it is a syntactic model!

It is a special case of a more general **internal realizability** interpretation.

$$[\![A]\!]_\varepsilon\ M \quad := \quad M \Vdash A$$

Given another syntactic model $[-]/[\![-]\!]$ we can define

$$\Gamma \vdash_{\underline{\mathsf{CIC}}} M : A \quad \longrightarrow \quad [\![\Gamma]\!]_\varepsilon \vdash_{\mathsf{CIC}} [M] : [\![A]\!] \quad + \quad [\![\Gamma]\!]_\varepsilon \vdash_{\mathsf{CIC}} [M]_\varepsilon : [\![A]\!]_\varepsilon\ [M]$$

$$(x : A \quad \longrightarrow \quad x : [\![A]\!], x_\varepsilon : [\![A]\!]_\varepsilon\ x)$$

Syntactic Models For Free

Bernardy-Lasson '11

There is a well-known parametricity interpretation for type theory

$$\Gamma \vdash_{\mathsf{CIC}} M : A \quad \longrightarrow \quad [\![\Gamma]\!]_\varepsilon \vdash_{\mathsf{CIC}} [M]_\varepsilon : [\![A]\!]_\varepsilon\ M$$

where $\quad [\![\cdot]\!]_\varepsilon := \cdot \quad$ and $\quad [\![\Gamma, x : A]\!]_\varepsilon := [\![\Gamma]\!]_\varepsilon, x : A, x_\varepsilon : [\![A]\!]_\varepsilon\ x$

Turns out it is a syntactic model!

It is a special case of a more general **internal realizability** interpretation.

$$[\![A]\!]_\varepsilon\ M \quad := \quad M \Vdash A$$

Given another syntactic model $[-]/[\![-]\!]$ we can define

$$\Gamma \vdash_{\underline{\mathsf{CIC}}} M : A \quad \longrightarrow \quad [\![\Gamma]\!]_\varepsilon \vdash_{\mathsf{CIC}} [M] : [\![A]\!] \quad + \quad [\![\Gamma]\!]_\varepsilon \vdash_{\mathsf{CIC}} [M]_\varepsilon : [\![A]\!]_\varepsilon\ [M]$$

$$(x : A \quad \longrightarrow \quad x : [\![A]\!], x_\varepsilon : [\![A]\!]_\varepsilon\ x)$$

Bernardy-Lasson is parametricity over identity.

On Parametric Presheaves

What does parametricity look like on the CBN presheaf model?

$$x : \mathbb{B} \quad \longrightarrow \quad \begin{cases} x : (\Pi(q : \mathbb{P})(\alpha : q \leq p).\, \mathbb{B}) \\ x_\varepsilon : \mathbb{B}_\varepsilon\ p\ x \end{cases}$$

On Parametric Presheaves

What does parametricity look like on the CBN presheaf model?

$$x : \mathbb{B} \quad \longrightarrow \quad \begin{cases} x : (\Pi(q : \mathbb{P})(\alpha : q \leq p).\, \mathbb{B}) \\ x_\varepsilon : \mathbb{B}_\varepsilon\ p\ x \end{cases}$$

We have a bit of constraints. To get dependent elimination we need:

1. $\mathbb{B}_\varepsilon\ p\ x$ iff $(x = \lambda q\,\alpha.\,\texttt{tt})$ or $(x = \lambda q\,\alpha.\,\texttt{ff})$

On Parametric Presheaves

What does parametricity look like on the CBN presheaf model?

$$x : \mathbb{B} \quad \longrightarrow \quad \left\{ \begin{array}{l} x : (\Pi(q : \mathbb{P})(\alpha : q \leq p).\, \mathbb{B}) \\ x_\varepsilon : \mathbb{B}_\varepsilon\ p\ x \end{array} \right.$$

We have a bit of constraints. To get dependent elimination we need:

1. $\mathbb{B}_\varepsilon\ p\ x$ iff $(x = \lambda q\,\alpha.\, \mathtt{tt})$ or $(x = \lambda q\,\alpha.\, \mathtt{ff})$
2. in a **unique** way, i.e. $b_1, b_2 : \mathbb{B}_\varepsilon\ p\ x \vdash b_1 = b_2$ (i.e. a HoTT proposition)

On Parametric Presheaves

What does parametricity look like on the CBN presheaf model?

$$x : \mathbb{B} \quad \longrightarrow \quad \begin{cases} x : (\Pi(q : \mathbb{P})(\alpha : q \leq p).\, \mathbb{B}) \\ x_\varepsilon : \mathbb{B}_\varepsilon\ p\ x \end{cases}$$

We have a bit of constraints. To get dependent elimination we need:

1. $\mathbb{B}_\varepsilon\ p\ x$ iff $(x = \lambda q\, \alpha.\, \mathtt{tt})$ or $(x = \lambda q\, \alpha.\, \mathtt{ff})$
2. in a **unique** way, i.e. $b_1, b_2 : \mathbb{B}_\varepsilon\ p\ x \vdash b_1 = b_2$ (i.e. a HoTT proposition)

But we also critically need to be compatible with the presheaf structure!

3. That is, $\theta_{\mathbb{B}_\varepsilon}\ (\alpha : q \leq p) : \mathbb{B}_\varepsilon\ p\ x \to \mathbb{B}_\varepsilon\ q\ (\alpha \cdot x)$

On Parametric Presheaves

What does parametricity look like on the CBN presheaf model?

$$x : \mathbb{B} \quad \longrightarrow \quad \begin{cases} x : (\Pi(q : \mathbb{P})(\alpha : q \leq p).\, \mathbb{B}) \\ x_\varepsilon : \mathbb{B}_\varepsilon \; p \; x \end{cases}$$

We have a bit of constraints. To get dependent elimination we need:

1. $\mathbb{B}_\varepsilon \; p \; x$ iff $(x = \lambda q\, \alpha.\, \texttt{tt})$ or $(x = \lambda q\, \alpha.\, \texttt{ff})$
2. in a **unique** way, i.e. $b_1, b_2 : \mathbb{B}_\varepsilon \; p \; x \vdash b_1 = b_2$ (i.e. a HoTT proposition)

But we also critically need to be compatible with the presheaf structure!

3. That is, $\theta_{\mathbb{B}_\varepsilon} \; (\alpha : q \leq p) : \mathbb{B}_\varepsilon \; p \; x \to \mathbb{B}_\varepsilon \; q \; (\alpha \cdot x)$
4. with further **definitional** functoriality to avoid coherence issues

On Parametric Presheaves

What does parametricity look like on the CBN presheaf model?

$$x : \mathbb{B} \quad \longrightarrow \quad \begin{cases} x : (\Pi(q : \mathbb{P})(\alpha : q \leq p).\, \mathbb{B}) \\ x_\varepsilon : \mathbb{B}_\varepsilon\ p\ x \end{cases}$$

We have a bit of constraints. To get dependent elimination we need:

1. $\mathbb{B}_\varepsilon\ p\ x$ iff $(x = \lambda q\, \alpha.\, \mathtt{tt})$ or $(x = \lambda q\, \alpha.\, \mathtt{ff})$
2. in a **unique** way, i.e. $b_1, b_2 : \mathbb{B}_\varepsilon\ p\ x \vdash b_1 = b_2$ (i.e. a HoTT proposition)

But we also critically need to be compatible with the presheaf structure!

3. That is, $\theta_{\mathbb{B}_\varepsilon}\ (\alpha : q \leq p) : \mathbb{B}_\varepsilon\ p\ x \rightarrow \mathbb{B}_\varepsilon\ q\ (\alpha \cdot x)$
4. with further **definitional** functoriality to avoid coherence issues

😱 Guess what? The CBV vs. CBN conundrum is back. 😱

Trouble All The Way Up

This is exactly the CBV vs. CBN conundrum **one level higher**

Either you pick $\mathbb{B}_\varepsilon\ p\ x := (x = \lambda q\,\alpha.\,\mathtt{tt}) + (x = \lambda q\,\alpha.\,\mathtt{ff})$

$\rightsquigarrow$ this satisfies unicity but breaks definitionality (i.e. CBV).

Or you freeify $\mathbb{B}_\varepsilon\ p\ x := \Pi q\,\alpha.(\alpha \cdot x = \lambda r\,\beta.\,\mathtt{tt}) + (\alpha \cdot x = \lambda r\,\beta.\,\mathtt{ff})$

$\rightsquigarrow$ this satisfies definitionality but breaks unicity (i.e. CBN).

Trouble All The Way Up

This is exactly the CBV vs. CBN conundrum **one level higher**

Either you pick $\mathbb{B}_\varepsilon\ p\ x := (x = \lambda q\,\alpha.\,\mathtt{tt}) + (x = \lambda q\,\alpha.\,\mathtt{ff})$

$\rightsquigarrow$ this satisfies unicity but breaks definitionality (i.e. CBV).

Or you freeify $\mathbb{B}_\varepsilon\ p\ x := \Pi q\,\alpha.(\alpha \cdot x = \lambda r\,\beta.\,\mathtt{tt}) + (\alpha \cdot x = \lambda r\,\beta.\,\mathtt{ff})$

$\rightsquigarrow$ this satisfies definitionality but breaks unicity (i.e. CBN).

It is not possible to get both at the same time in CIC!

Playing Cubes

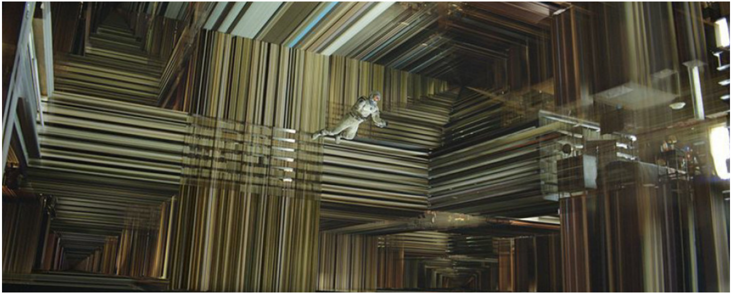
We could solve this with infinite towers of parametricity.

That is, the n-level proof is guaranteed to be pure by then $(n+1)$-level one.

Playing Cubes

We could solve this with infinite towers of parametricity.

That is, the n-level proof is guaranteed to be pure by then $(n+1)$-level one.

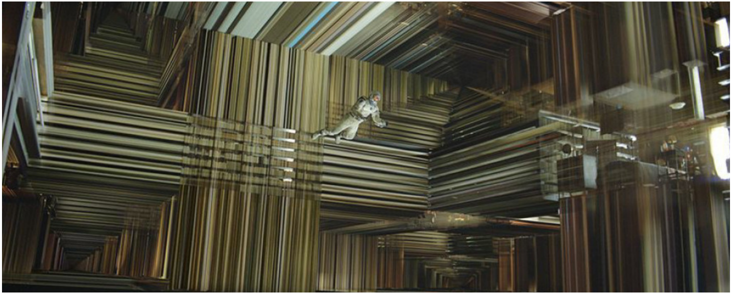


``Oh noes, not cubical type theory again!''

Playing Cubes

We could solve this with infinite towers of parametricity.

That is, the n-level proof is guaranteed to be pure by then $(n + 1)$-level one.



``Oh noes, not cubical type theory again!''

But CuTT itself is justified by presheaf models.

What would be the point to implement presheaves using presheaves?

(On the virtues of Authoritarianism.)

A New Hope

Essentially, we were blocked on this issue since then. When suddenly...

A New Hope

Essentially, we were blocked on this issue since then. When suddenly...

Gaëtan Gilbert, Jesper Cockx, Matthieu Sozeau, and Nicolas Tabareau.
Definitional proof-irrelevance without K.
Proc. ACM Program. Lang., 3(POPL):3:1–3:28, 2019.

A New Hope

Essentially, we were blocked on this issue since then. When suddenly...

Gaëtan Gilbert, Jesper Cockx, Matthieu Sozeau, and Nicolas Tabareau.
Definitional proof-irrelevance without K.
Proc. ACM Program. Lang., 3(POPL):3:1–3:28, 2019.

They introduce a new sort SProp of **strict propositions**.

$$M, N : A : \texttt{SProp} \quad \longrightarrow \quad \vdash M \equiv N$$

- It can be seen as a well-behaved subset of Prop
- It is compatible with HoTT
- It enjoys all good syntactic properties (SN, canonicity, decidability...)
- Coq has it impredicative, Agda has a parallel hierarchy $\texttt{SProp}_i$

Strict Propositions

Critically, `SProp` is closed under products.

$$\vdash A : \Box, \qquad x : A \vdash B : \texttt{SProp} \quad \longrightarrow \quad \vdash \Pi(x : A).\, B : \texttt{SProp}$$

Strict Propositions

Critically, `SProp` is closed under products.

$$\vdash A : \square, \qquad x : A \vdash B : \texttt{SProp} \quad \longrightarrow \quad \vdash \Pi(x : A).\, B : \texttt{SProp}$$

The hard question is elimination from `SProp` to $\square$

A restriction of singleton elimination: ≤ 1 constructor $+$ irrelevant args

Strict Propositions

Critically, `SProp` is closed under products.

$$\vdash A : \square, \qquad x : A \vdash B : \texttt{SProp} \quad \longrightarrow \quad \vdash \Pi(x : A).\, B : \texttt{SProp}$$

The hard question is elimination from `SProp` to $\square$

A restriction of singleton elimination: ≤ 1 constructor + irrelevant args

Three archetypical examples in `Prop`

Strict Propositions

Critically, `SProp` is closed under products.

$$\vdash A : \Box, \qquad x : A \vdash B : \texttt{SProp} \quad \longrightarrow \quad \vdash \Pi(x : A).\, B : \texttt{SProp}$$

The hard question is elimination from `SProp` to $\Box$

A restriction of singleton elimination: ≤ 1 constructor + irrelevant args

Three archetypical examples in `Prop`

- `False`
- `Acc`
- `eq`

Strict Propositions

Critically, `SProp` is closed under products.

$$\vdash A : \Box, \qquad x : A \vdash B : \texttt{SProp} \quad \longrightarrow \quad \vdash \Pi(x : A).\, B : \texttt{SProp}$$

The hard question is elimination from `SProp` to $\Box$

A restriction of singleton elimination: ≤ 1 constructor + irrelevant args

Three archetypical examples in `Prop`

- `False` $\rightsquigarrow$ elimination valid ☺
- `Acc`
- `eq`

Strict Propositions

Critically, SProp is closed under products.

$$\vdash A : \square, \qquad x : A \vdash B : \texttt{SProp} \quad \longrightarrow \quad \vdash \Pi(x : A).\, B : \texttt{SProp}$$

The hard question is elimination from SProp to $\square$

A restriction of singleton elimination: ≤ 1 constructor $+$ irrelevant args

Three archetypical examples in Prop

- False $\rightsquigarrow$ elimination valid ☺
- Acc $\rightsquigarrow$ implies undecidability of type-checking ☹
- eq

Strict Propositions

Critically, SProp is closed under products.

$$\vdash A : \square, \qquad x : A \vdash B : \texttt{SProp} \quad \longrightarrow \quad \vdash \Pi(x : A).\, B : \texttt{SProp}$$

The hard question is elimination from SProp to $\square$

A restriction of singleton elimination: ≤ 1 constructor + irrelevant args

Three archetypical examples in Prop

- False $\rightsquigarrow$ elimination valid ☺
- Acc $\rightsquigarrow$ implies undecidability of type-checking ☹
- eq $\rightsquigarrow$ implies UIP, incompatible with HoTT 😕 (who cares?)

Strict Propositions

Critically, `SProp` is closed under products.

$$\vdash A : \square, \qquad x : A \vdash B : \texttt{SProp} \quad \longrightarrow \quad \vdash \Pi(x : A).\, B : \texttt{SProp}$$

The hard question is elimination from `SProp` to $\square$

A restriction of singleton elimination: ≤ 1 constructor + irrelevant args

Three archetypical examples in `Prop`

- `False` $\rightsquigarrow$ elimination valid ☺
- `Acc` $\rightsquigarrow$ implies undecidability of type-checking ☹
- eq $\rightsquigarrow$ implies UIP, incompatible with HoTT 😕 (who cares?)

Accepting the elimination of eq gives rise to a **strict equality**.

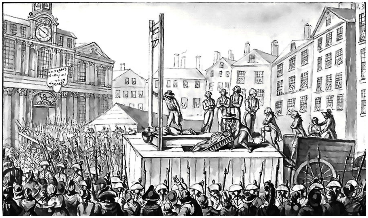
A Strict Doctrine

When the libertarian HoTT freely adds infinite towers of equalities...

A Strict Doctrine

When the libertarian HoTT freely adds infinite towers of equalities...

... the authoritarian $\mathfrak{s}$CIC will instead **guillotine** all higher equalities.

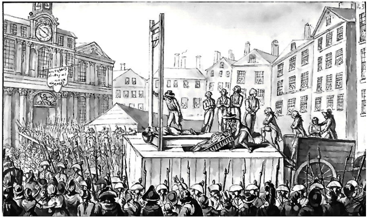


Art. 1. *All humans are born uniquely equal in rights.*

A Strict Doctrine

When the libertarian HoTT freely adds infinite towers of equalities...

... the authoritarian $\mathfrak{s}$CIC will instead **guillotine** all higher equalities.



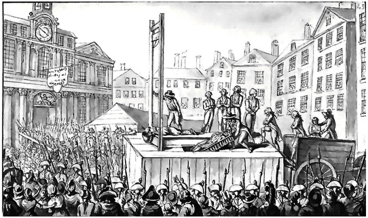
Art. 1. *All humans are born uniquely equal in rights.*

Strict equality is the authoritarian way to solve the coherence hell.

A Strict Doctrine

When the libertarian HoTT freely adds infinite towers of equalities...

... the authoritarian $\mathfrak{s}$CIC will instead **guillotine** all higher equalities.



Art. 1. *All humans are born uniquely equal in rights.*

Strict equality is the authoritarian way to solve the coherence hell.

(By default, `SProp` as implemented in Coq doesn't take side, you have to opt-in.)

Strict Parametricity

In the parametric presheaf translation,

- make the parametricity predicate free $\rightsquigarrow$ **definitional functoriality**
- require it to be a strict proposition $\rightsquigarrow$ **proof uniqueness**

$$x : A \quad \longrightarrow \quad \left\{ \begin{array}{l} x : (\Pi(q : \mathbb{P})(\alpha : q \leq p).\, [\![A]\!]_q) \\ x_\varepsilon : (\Pi(q : \mathbb{P})(\alpha : q \leq p).\, [\![A]\!]_\varepsilon\ q\ (\alpha \cdot x)) \end{array} \right.$$

where critically $[\![A]\!]_\varepsilon\ p\ x : \texttt{SProp}$.

Strict Parametricity

In the parametric presheaf translation,

- make the parametricity predicate free $\rightsquigarrow$ **definitional functoriality**
- require it to be a strict proposition $\rightsquigarrow$ **proof uniqueness**

$$x : A \quad \longrightarrow \quad \left\{ \begin{array}{l} x : (\Pi(q : \mathbb{P})(\alpha : q \leq p).\, [\![A]\!]_q) \\ x_\varepsilon : (\Pi(q : \mathbb{P})(\alpha : q \leq p).\, [\![A]\!]_\varepsilon \; q \; (\alpha \cdot x)) \end{array} \right.$$

where critically $[\![A]\!]_\varepsilon \; p \; x : \texttt{SProp}$.

We call the result the **prefascist translation**. (lat. *fascis* : sheaf)

Strict Parametricity

In the parametric presheaf translation,

- make the parametricity predicate free $\rightsquigarrow$ **definitional functoriality**
- require it to be a strict proposition $\rightsquigarrow$ **proof uniqueness**

$$x : A \quad \longrightarrow \quad \begin{cases} x : (\Pi(q : \mathbb{P})(\alpha : q \leq p).\, [\![A]\!]_q) \\ x_\varepsilon : (\Pi(q : \mathbb{P})(\alpha : q \leq p).\, [\![A]\!]_\varepsilon\ q\ (\alpha \cdot x)) \end{cases}$$

where critically $[\![A]\!]_\varepsilon\ p\ x : \mathtt{SProp}$.

We call the result the **prefascist translation**. (lat. *fascis* : sheaf)

Theorem (Pédrot '20)

The prefascist translation is a syntactic model of CIC *into* $\mathfrak{s}$CIC.

- Full conversion, full dependent elimination.
- The actual construction is a tad involved, but boils down to the above.
- Unsurprinsingly, UIP is required to interpret universes (tricky!).

No Pain, No Gain

$\mathfrak{s}$CIC is way weaker than ETT

$\mathfrak{s}$CIC is **conjectured** to enjoy the usual good syntactic properties.

- Canonicity seems relatively easy to show
- UIP makes reduction depend on conversion though
- SN is problematic, e.g. $\mathfrak{s}$CIC + an impredicative universe is **not** SN
- Hoping that SN holds in the predicative case, decidability follows

No Pain, No Gain

$\mathfrak{s}$CIC is way weaker than ETT

$\mathfrak{s}$CIC is **conjectured** to enjoy the usual good syntactic properties.

- Canonicity seems relatively easy to show
- UIP makes reduction depend on conversion though
- SN is problematic, e.g. $\mathfrak{s}$CIC + an impredicative universe is **not** SN
- Hoping that SN holds in the predicative case, decidability follows

We don't rely on impredicativity in the prefascist model

We would inherit the purported good properties $\mathfrak{s}$CIC for free.

Back to Set

Set is a model of $\mathfrak{s}$CIC

Thus, the prefascist model can also be described set-theoretically.

$\mathbf{Set}$ is a model of $\mathfrak{s}$CIC

Thus, the prefascist model can also be described set-theoretically.

A prefascist set $\mathcal{A} := (\mathcal{A}_p, (-) \Vdash_p \mathcal{A})$ over a category $\mathbb{P}$ is given by

- a family of sets $\mathcal{A}_p$ for $p \in \mathbb{P}$.
- a family of predicates $(-) \Vdash_p \mathcal{A} \quad \subseteq \quad \text{Cone}_p(\mathcal{A}) := \Pi(q : \mathbb{P})(\alpha : q \leq p).\, \mathcal{A}_q$

Back to Set

Set is a model of $\mathfrak{s}$CIC

Thus, the prefascist model can also be described set-theoretically.

A prefascist set $\mathcal{A} := (\mathcal{A}_p, (-) \Vdash_p \mathcal{A})$ over a category $\mathbb{P}$ is given by

- a family of sets $\mathcal{A}_p$ for $p \in \mathbb{P}$.
- a family of predicates $(-) \Vdash_p \mathcal{A} \quad \subseteq \quad \mathsf{Cone}_p(\mathcal{A}) := \Pi(q : \mathbb{P})(\alpha : q \leq p).\, \mathcal{A}_q$

A prefascist morphism f from $\mathcal{A}$ to $\mathcal{B}$ is

- a family of functions $f_p : \mathsf{El}_p\ \mathcal{A} \rightarrow \mathcal{B}_p$
- preserving predicates, i.e.

$$\forall x : \mathsf{El}_p\ \mathcal{A}. \quad \mathsf{app}_p(f, x) \Vdash_p \mathcal{B}$$

where

$$\begin{array}{lcl} \mathsf{El}_p\ \mathcal{A} & := & \{x : \mathsf{Cone}_p(\mathcal{A}) \mid \forall q\, (\alpha : q \leq p).\, (\alpha \cdot x) \Vdash_q \mathcal{A}\} \\ \mathsf{app}_p(f, x) & := & \lambda q\, (\alpha : q \leq p).\, f_q\ (\alpha \cdot x) \end{array}$$

Through The Looking Glass

Theorem

Prefascist sets over $\mathbb{P}$ form a category $\mathbf{Pfs}(\mathbb{P})$ with ***definitional*** *laws.*

Through The Looking Glass

Theorem

*Prefascist sets over $\mathbb{P}$ form a category $\mathbf{Pfs}(\mathbb{P})$ with **definitional** laws.*

Theorem

As categories, $\mathbf{Psh}(\mathbb{P})$ and $\mathbf{Pfs}(\mathbb{P})$ are equivalent.

Through The Looking Glass

Theorem

Prefascist sets over $\mathbb{P}$ form a category $\mathbf{Pfs}(\mathbb{P})$ with ***definitional*** *laws.*

Theorem

As categories, $\mathbf{Psh}(\mathbb{P})$ and $\mathbf{Pfs}(\mathbb{P})$ are equivalent.

Proving this requires extensionality principles!

- Hence, in a set-theoretical meta, both describe the same objects
- Yet, $\mathbf{Pfs}(\mathbb{P})$ is better behaved in an intensional setting
- This could come in handy for higher category theory...

Through The Looking Glass

Theorem

*Prefascist sets over $\mathbb{P}$ form a category $\mathbf{Pfs}(\mathbb{P})$ with **definitional** laws.*

Theorem

As categories, $\mathbf{Psh}(\mathbb{P})$ and $\mathbf{Pfs}(\mathbb{P})$ are equivalent.

Proving this requires extensionality principles!

- Hence, in a set-theoretical meta, both describe the same objects
- Yet, $\mathbf{Pfs}(\mathbb{P})$ is better behaved in an intensional setting
- This could come in handy for higher category theory...

Takeaway: prefascist sets are a better presentation of presheaves

Application

Russian Constructivism

Russian Constructivist School

A splinter group of constructivists, whose core tenet can be summarized as:

Proofs are Kleene realizers

Russian Constructivist School

A splinter group of constructivists, whose core tenet can be summarized as:

Proofs are Kleene realizers

Thus, the principle that puts it apart both from Brouwer **and** Bishop:

Markov's Principle (MP)

$$\forall (f : \mathbb{N} \to \mathbb{B}).\, \neg\neg(\exists n : \mathbb{N}.\, f\, n = \mathtt{tt}) \to \exists n : \mathbb{N}.\, f\, n = \mathtt{tt}$$

- A lot of equivalent statements, e.g. a TM that doesn't loop terminates
- Semi-classical: $\mathbf{HA}^\omega \subsetneq \mathbf{HA}^\omega + \mathsf{MP} \subsetneq \mathbf{PA}^\omega$
- Known to preserve existence property (i.e. canonicity)

Russian Constructivist School

A splinter group of constructivists, whose core tenet can be summarized as:

Proofs are Kleene realizers

Thus, the principle that puts it apart both from Brouwer **and** Bishop:

Markov's Principle (MP)

$$\forall (f : \mathbb{N} \to \mathbb{B}).\, \neg\neg(\exists n : \mathbb{N}.\, f\, n = \mathtt{tt}) \to \exists n : \mathbb{N}.\, f\, n = \mathtt{tt}$$

- A lot of equivalent statements, e.g. a TM that doesn't loop terminates
- Semi-classical: $\mathbf{HA}^\omega \subsetneq \mathbf{HA}^\omega + \mathsf{MP} \subsetneq \mathbf{PA}^\omega$
- Known to preserve existence property (i.e. canonicity)

What if we tried to extend CIC with MP through a syntactic model?

Let's look at the realizer

$$\forall (f : \mathbb{N} \rightarrow \mathbb{B}). \neg\neg(\exists n : \mathbb{N}. f\ n = \texttt{tt}) \rightarrow \exists n : \mathbb{N}. f\ n = \texttt{tt}$$

```
let mp f _ :=
  let n := ref 0 in
  while true do
    if f !n then return n else n := n + 1
  done
```

MP in Kleene Realizability

Let's look at the realizer

$$\forall (f : \mathbb{N} \to \mathbb{B}). \neg\neg(\exists n : \mathbb{N}. f\ n = \texttt{tt}) \to \exists n : \mathbb{N}. f\ n = \texttt{tt}$$

```
let mp f _ :=
  let n := ref 0 in
  while true do
    if f !n then return n else n := n + 1
  done
```

Proving `mp` $\Vdash$ MP needs MP in the meta-theory!

- As such, this is **cheating**
- The realizer doesn't use the doubly-negated proof
- Relies on a semi-classical meta-theory and unbounded loops
- We have little hope to implement this in CIC with a syntactic model

MP in Kleene Realizability

Let's look at the realizer

$$\forall (f : \mathbb{N} \to \mathbb{B}).\, \neg\neg(\exists n : \mathbb{N}.\, f\, n = \mathtt{tt}) \to \exists n : \mathbb{N}.\, f\, n = \mathtt{tt}$$

```
let mp f _ :=
  let n := ref 0 in
  while true do
    if f !n then return n else n := n + 1
  done
```

Proving mp $\Vdash$ MP needs MP in the meta-theory!

- As such, this is **cheating**
- The realizer doesn't use the doubly-negated proof
- Relies on a semi-classical meta-theory and unbounded loops
- We have little hope to implement this in CIC with a syntactic model

We need something else...

What Else?

Not one, but at least **two** alternatives!

What Else?

Not one, but at least **two** alternatives!

- Coquand-Hofmann's syntactic model for $\mathbf{HA}^\omega + \mathrm{MP}$
- Herbelin's direct style proof using static exceptions

$$\begin{aligned}&\texttt{mp}\ (p : \neg\neg(\exists n.\, f\ n = \texttt{tt})) :=\\ &\quad \texttt{try}_\alpha\ \bot_e\ (p\ (\lambda k.\, k\ (\lambda n.\, \texttt{raise}_\alpha\ n)))\ \texttt{with}\ \alpha\ n \mapsto n\end{aligned}$$

What Else?

Not one, but at least **two** alternatives!

- Coquand-Hofmann's syntactic model for $\mathbf{HA}^\omega + \mathrm{MP}$
- Herbelin's direct style proof using static exceptions

$$\begin{array}{l}\mathtt{mp}\ (p : \neg\neg(\exists n.\, f\ n = \mathtt{tt})) := \\ \qquad \mathtt{try}_\alpha\ \bot_e\ (p\ (\lambda k.\, k\ (\lambda n.\, \mathtt{raise}_\alpha\ n)))\ \mathtt{with}\ \alpha\ n \mapsto n\end{array}$$

In the remainder, we'll show that

- Coquand-Hofmann's model scales to CIC
- It can be presented as the composition of two translations
- It has the same computational content as Herbelin's proof

High-level view

CH's model is a mix of Kripke semantics and Friedman's A-translation.

- Kripke semantics $\rightsquigarrow$ global cell
- A-translation $\rightsquigarrow$ exceptions

High-level view

CH's model is a mix of Kripke semantics and Friedman's A-translation.

- Kripke semantics $\rightsquigarrow$ global cell
- A-translation $\rightsquigarrow$ exceptions

They specifically pick:

- Kripke cell of type $\mathbb{N} \to \mathbb{B}$, where

$$q \leq p := \forall n : \mathbb{N}.\, p\ n = \mathtt{tt} \to q\ n = \mathtt{tt} \qquad (q \text{ truer than } p)$$

- Exceptions of type $\mathbf{E}_p := \exists n : \mathbb{N}.\, p\ n = \mathtt{tt}$

High-level view

CH's model is a mix of Kripke semantics and Friedman's A-translation.

- Kripke semantics $\rightsquigarrow$ global cell
- A-translation $\rightsquigarrow$ exceptions

They specifically pick:

- Kripke cell of type $\mathbb{N} \rightarrow \mathbb{B}$, where

$$q \leq p := \forall n : \mathbb{N}.\, p\ n = \texttt{tt} \rightarrow q\ n = \texttt{tt} \qquad (q \text{ truer than } p)$$

- Exceptions of type $\mathbf{E}_p := \exists n : \mathbb{N}.\, p\ n = \texttt{tt}$

The secret sauce is that the exception type depends on the current p

Pipelining

Coquand-Hofmann's model is a bit ad-hoc

Pipelining

Coquand-Hofmann's model is a bit ad-hoc

Instead, we present our CIC variant synthetically as the composition

$$\mathsf{CIC} \quad \xrightarrow{\mathbf{Exn}} \quad \mathsf{CIC} + \mathcal{E} \quad \xrightarrow{\mathbf{Pfs}} \quad \mathfrak{s}\mathsf{CIC}$$

where

- **Pfs** is the prefascist model described before
- **Exn** is the exceptional model, a CIC-worthy A-translation

Failure is Not an Option

Exn is a very simple syntactic model of CIC

Failure is Not an Option

Exn is a very simple syntactic model of CIC

Pick a fixed exception type $\mathcal{E}$ in the target theory.

$$\vdash_{\mathcal{S}} A : \square \quad \longrightarrow \quad \vdash_{\mathcal{T}} [A] := (\llbracket A \rrbracket, [A]_{\varnothing}) : \Sigma A_0 : \square.\, (\mathcal{E} \rightarrow A_0)$$
$$\vdash_{\mathcal{S}} M : A \quad \longrightarrow \quad \vdash_{\mathcal{T}} [M] : \llbracket A \rrbracket$$

Every type $\llbracket A \rrbracket$ comes with its failure function $[A]_{\varnothing} : \mathcal{E} \rightarrow \llbracket A \rrbracket$

Failure is Not an Option

Exn is a very simple syntactic model of CIC

Pick a fixed exception type $\mathcal{E}$ in the target theory.

$$\vdash_{\mathcal{S}} A : \square \quad \longrightarrow \quad \vdash_{\mathcal{T}} [A] := ([\![A]\!], [A]_{\varnothing}) : \Sigma A_0 : \square.\, (\mathcal{E} \rightarrow A_0)$$

$$\vdash_{\mathcal{S}} M : A \quad \longrightarrow \quad \vdash_{\mathcal{T}} [M] : [\![A]\!]$$

Every type $[\![A]\!]$ comes with its failure function $[A]_{\varnothing} : \mathcal{E} \rightarrow [\![A]\!]$

- Functions are interpreted as $[\![\Pi x : A.\, B]\!] := \Pi x : [\![A]\!].\, [\![B]\!]$
- Inductive types are interpreted pointwise + a dedicated constructor for error

$$[\![\mathbb{B}]\!] := \mathtt{tt}_{\mathcal{E}} : [\![\mathbb{B}]\!] \mid \mathtt{ff}_{\mathcal{E}} : [\![\mathbb{B}]\!] \mid \mathbb{B}_{\varnothing} : \mathcal{E} \rightarrow [\![\mathbb{B}]\!]$$

Failure is Not an Option

Exn is a very simple syntactic model of CIC

Pick a fixed exception type $\mathcal{E}$ in the target theory.

$$\vdash_{\mathcal{S}} A : \square \quad \longrightarrow \quad \vdash_{\mathcal{T}} [A] := (\llbracket A \rrbracket, [A]_{\varnothing}) : \Sigma A_0 : \square.\, (\mathcal{E} \to A_0)$$
$$\vdash_{\mathcal{S}} M : A \quad \longrightarrow \quad \vdash_{\mathcal{T}} [M] : \llbracket A \rrbracket$$

Every type $\llbracket A \rrbracket$ comes with its failure function $[A]_{\varnothing} : \mathcal{E} \to \llbracket A \rrbracket$

- Functions are interpreted as $\llbracket \Pi x : A.\, B \rrbracket := \Pi x : \llbracket A \rrbracket.\, \llbracket B \rrbracket$
- Inductive types are interpreted pointwise + a dedicated constructor for error

$$\llbracket \mathbb{B} \rrbracket := \mathtt{tt}_{\mathcal{E}} : \llbracket \mathbb{B} \rrbracket \mid \mathtt{ff}_{\mathcal{E}} : \llbracket \mathbb{B} \rrbracket \mid \mathbb{B}_{\varnothing} : \mathcal{E} \to \llbracket \mathbb{B} \rrbracket$$

Theorem

Provided there is no closed $M : \mathcal{E}$ in the target theory, the source theory enjoys canonicity. In particular, it is consistent.

Somebody Set Up Us The Bomb

We perform the exceptional translation over an **exotic** type of exceptions

$$\mathsf{CIC} \quad \xrightarrow{\textbf{Exn}} \quad \mathsf{CIC} + \mathcal{E} \quad \xrightarrow{\textbf{Pfs}} \quad \mathfrak{s}\mathsf{CIC}$$

$\mathcal{E}$ exists in the prefascist model over $\mathbb{P} := \mathbb{N} \to \mathbb{B}$.

$$\mathcal{E}_p := \Sigma n : \mathbb{N}.\, p\ n = \mathtt{tt}$$

Somebody Set Up Us The Bomb

We perform the exceptional translation over an **exotic** type of exceptions

$$\mathsf{CIC} \xrightarrow{\textbf{Exn}} \mathsf{CIC} + \mathcal{E} \xrightarrow{\textbf{Pfs}} \mathfrak{s}\mathsf{CIC}$$

$\mathcal{E}$ exists in the prefascist model over $\mathbb{P} := \mathbb{N} \to \mathbb{B}$.

$$\mathcal{E}_p := \Sigma n : \mathbb{N}.\, p\ n = \texttt{tt}$$

There is no closed proof of $\mathcal{E}$ in $\mathsf{CIC} + \mathcal{E}$ since

$$\mathcal{E}_p := \Sigma n : \mathbb{N}.\, \texttt{ff} = \texttt{tt} \qquad \text{for } p \text{ constantly } \texttt{ff}$$

(We **do not** have $\vdash_{\mathsf{CIC}+\mathcal{E}} \neg\mathcal{E}$ though.)

Somebody Set Up Us The Bomb

We perform the exceptional translation over an **exotic** type of exceptions

$$\mathsf{CIC} \xrightarrow{\textbf{Exn}} \mathsf{CIC} + \mathcal{E} \xrightarrow{\textbf{Pfs}} \mathfrak{s}\mathsf{CIC}$$

$\mathcal{E}$ exists in the prefascist model over $\mathbb{P} := \mathbb{N} \rightarrow \mathbb{B}$.

$$\mathcal{E}_p := \Sigma n : \mathbb{N}.\, p\ n = \mathtt{tt}$$

There is no closed proof of $\mathcal{E}$ in $\mathsf{CIC} + \mathcal{E}$ since

$$\mathcal{E}_p := \Sigma n : \mathbb{N}.\, \mathtt{ff} = \mathtt{tt} \qquad \text{for } p \text{ constantly } \mathtt{ff}$$

(We **do not** have $\vdash_{\mathsf{CIC}+\mathcal{E}} \neg\mathcal{E}$ though.)

Therefore, the leftmost source theory is consistent.

Realizing MP

We also have a modality in $\mathsf{CIC} + \mathcal{E}$

$$\begin{array}{lcl} \texttt{local} & : & (\mathbb{N} \rightarrow \mathbb{B}) \rightarrow \Box \rightarrow \Box \\ [\texttt{local}\ \varphi\ A]_p & \mathrel{:\stackrel{\sim}{=}} & [A]_{p \wedge \varphi} \end{array}$$

- $\texttt{return} : A \rightarrow \texttt{local}\ \varphi\ A$
- `local` commutes to arrows and positive types
- $\texttt{local}\ \varphi\ \mathcal{E} \quad \cong \quad \mathcal{E} + (\Sigma n : \mathbb{N}.\, \varphi\ n = \texttt{tt})$

Realizing MP

We also have a modality in $\mathsf{CIC} + \mathcal{E}$

$$\begin{array}{lcl} \texttt{local} & : & (\mathbb{N} \to \mathbb{B}) \to \square \to \square \\ [\texttt{local}\ \varphi\ A]_p & :\stackrel{\sim}{=} & [A]_{p \wedge \varphi} \end{array}$$

- $\texttt{return} : A \to \texttt{local}\ \varphi\ A$
- `local` commutes to arrows and positive types
- $\texttt{local}\ \varphi\ \mathcal{E} \quad \cong \quad \mathcal{E} + (\Sigma n : \mathbb{N}.\, \varphi\ n = \texttt{tt})$

To realize MP, we perform intuitionistic symbol pushing in $\mathsf{CIC} + \mathcal{E}$

$$\begin{array}{lcl} [\![\neg\neg(\Sigma n : \mathbb{N}.\, \varphi\ n = \texttt{tt})]\!]_{\mathcal{E}} & \cong & ((\Sigma n : \mathbb{N}.\, \varphi\ n = \texttt{tt}) \to \mathcal{E}) \to \mathcal{E} \\ & \to & \texttt{local}\ \varphi\ (((\Sigma n : \mathbb{N}.\, \varphi\ n = \texttt{tt}) \to \mathcal{E}) \to \mathcal{E}) \\ & \cong & ((\Sigma n : \mathbb{N}.\, \varphi\ n = \texttt{tt}) \to \texttt{local}\ \varphi\ \mathcal{E}) \to \texttt{local}\ \varphi\ \mathcal{E} \\ & \to & \mathcal{E} + (\Sigma n : \mathbb{N}.\, \varphi\ n = \texttt{tt}) \\ & \to & [\![\Sigma n : \mathbb{N}.\, \varphi\ n = \texttt{tt}]\!]_{\mathcal{E}} \end{array}$$

A Computational Analysis of MP

Every time we go under `local` we get new exceptions!

$$\texttt{local}\ \varphi\ \mathcal{E} \quad \cong \quad \mathcal{E} + (\Sigma n : \mathbb{N}.\, \varphi\ n = \texttt{tt})$$

`return` is a delimited continuation prompt / static exception binder.

A Computational Analysis of MP

Every time we go under `local` we get new exceptions!

$$\texttt{local}\ \varphi\ \mathcal{E} \quad \cong \quad \mathcal{E} + (\Sigma n : \mathbb{N}.\, \varphi\ n = \texttt{tt})$$

`return` is a delimited continuation prompt / static exception binder.

The structure of the realizer thus follows closely Herbelin's proof.

$$\begin{array}{l} \texttt{mp}\ (p : \neg\neg(\exists n.\, f\ n = \texttt{tt})) := \\ \qquad \texttt{try}_\alpha\ \bot_e\ (p\ (\lambda k.\, k\ (\lambda n.\, \texttt{raise}_\alpha\ n)))\ \texttt{with}\ \alpha\ n \mapsto n \end{array}$$

In particular p can raise exceptions from outside, which is reflected here.

A Computational Analysis of MP

Every time we go under `local` we get new exceptions!

$$\texttt{local}\ \varphi\ \mathcal{E} \quad \cong \quad \mathcal{E} + (\Sigma n : \mathbb{N}.\ \varphi\ n = \texttt{tt})$$

`return` is a delimited continuation prompt / static exception binder.

The structure of the realizer thus follows closely Herbelin's proof.

$$\begin{array}{l} \texttt{mp}\ (p : \neg\neg(\exists n.\, f\ n = \texttt{tt})) := \\ \qquad \texttt{try}_\alpha\ \bot_e\ (p\ (\lambda k.\, k\ (\lambda n.\, \texttt{raise}_\alpha\ n)))\ \texttt{with}\ \alpha\ n \mapsto n \end{array}$$

In particular p can raise exceptions from outside, which is reflected here.

Thus, Herbelin's proof is the direct style variant of Coquand-Hofmann

Final Digression

This is also highly reminiscent of NbE models

Two canonical ways to extend Kripke completeness to positive types:

- Add neutral terms to the semantic of positive types
- Add MP in the meta

Final Digression

This is also highly reminiscent of NbE models

Two canonical ways to extend Kripke completeness to positive types:

- Add neutral terms to the semantic of positive types
- Add MP in the meta

Neutral terms behave as statically bound exceptions

As our model shows, this two techniques are morally equivalent.

This also highlights suspicious ties between delimited continuations and presheaves.

Conclusion

On presheaves:

- Presheaves are a purified sublanguage of a monotonic reader effect
- We have given a better-behaved presentation of presheaves
- It is a syntactic model that relies on strict equality in the target
- Provides for free extensions of CIC with SN, canonicity and the like
- ... assuming $\mathfrak{s}$CIC enjoys this $(\dagger)$

Conclusion

On presheaves:

- Presheaves are a purified sublanguage of a monotonic reader effect
- We have given a better-behaved presentation of presheaves
- It is a syntactic model that relies on strict equality in the target
- Provides for free extensions of CIC with SN, canonicity and the like
- ... assuming $\mathfrak{s}$CIC enjoys this $(\dagger)$

On MP:

- Composition of the prefascist model with another model of ours
- This provides a computational extension of CIC that validates MP
- Once again, good properties for free

Conclusion

On presheaves:

- Presheaves are a purified sublanguage of a monotonic reader effect
- We have given a better-behaved presentation of presheaves
- It is a syntactic model that relies on strict equality in the target
- Provides for free extensions of CIC with SN, canonicity and the like
- ... assuming $\mathfrak{s}$CIC enjoys this $(\dagger)$

On MP:

- Composition of the prefascist model with another model of ours
- This provides a computational extension of CIC that validates MP
- Once again, good properties for free

TODO:

- Implement cubical type theory in this model

Thanks for your attention.