All your base categories are belong to us

A syntactic model of presheaves in type theory

Pierre-Marie Pédrot

INRIA, Gallinette team

MSFP 2020

31st August 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

The Good Properties™

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

The Good Properties™

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 in Three Easy Steps

Our mission

To boldly extend the logical / computational expressivity of CIC

Extending Coq in Three Easy Steps

Our mission

To boldly extend the logical / computational expressivity of CIC

$\rightsquigarrow$ 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 in Three Easy Steps

Our mission

To boldly extend the logical / computational expressivity of CIC

$\rightsquigarrow$ 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!

- Proof-relevant Kripke semantics / Intuitionistic Forcing
- Bread and Butter of Model Construction
- They Are Everywhere: Cubical, Modal, Guarded, NbE, ...

I Herd U Liek Murphims

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.)

I Herd U Liek Murphims

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 $\mathsf{Psh}(\mathbb{P})$.

I Herd U Liek Murphims

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**!

I Herd U Liek Murphims

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**!

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

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" (a.k.a. monotonicity)

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

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" (a.k.a. monotonicity)

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

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

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" (a.k.a. monotonicity)

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

"$\theta_{\mathbf{A}}\,\alpha\,x$ 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 $\beta \in \mathbb{P}(r, q)$:

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

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" (a.k.a. monotonicity)

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

"$\theta_\mathbf{A}\,\alpha\,x$ 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 $\mathbb{P}$"

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

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

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

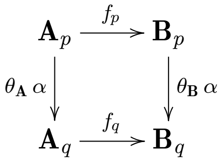
What is Psh($\mathbb{P}$)?

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

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

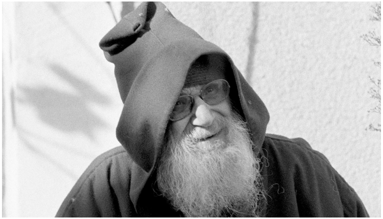
$$\theta_{\mathbf{B}}\, \alpha\, (f_p\, x) \equiv f_q\, (\theta_{\mathbf{A}}\, \alpha\, x)$$

"f is compatible with restriction"



The Wise Speak Only of What They Know

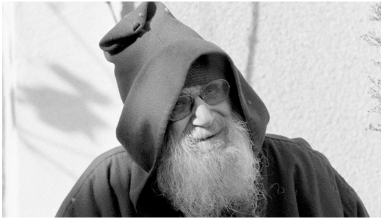
Psh($\mathbb{P}$) is a topos.



"Speak, friend, and pullback."

The Wise Speak Only of What They Know

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



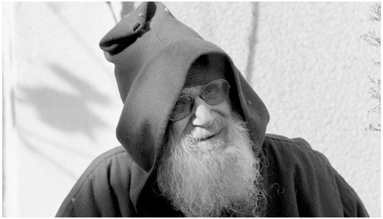
"Speak, friend, and pullback."

Who cares?

Presheaves actually form a model of CIC.

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

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



"Speak, friend, and pullback."

Who cares?

Presheaves actually form a model of CIC.

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

Yet another *set-theoretical* model!

ZF Set Up Us The Bomb

Let's have a look at **The Good Properties**™ we long for.

ZF Set Up Us The Bomb

Let's have a look at **The Good Properties**™ we long for.

Consistency There is no proof of `False`. ☺

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?

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

ZF Set Up Us The Bomb

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. ☹

ZF Set Up Us The Bomb

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?

ZF Set Up Us The Bomb

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**.

ZF Set Up Us The Bomb

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...

"This is a *compiler*!"

On Curry-Howard Poetry

General models are more like **interpreters**.

No separation between target vs. host languages

$$\vdash_{\mathcal{S}} M : A \quad \xrightarrow{\text{host}} \quad \vDash_{\mathcal{M}} A \qquad \text{“a blob”}$$

On Curry-Howard Poetry

General models are more like **interpreters**.

No separation between target vs. host languages

$$\vdash_{\mathcal{S}} M : A \quad \xrightarrow{\text{host}} \quad \vDash_{\mathcal{M}} A \qquad \text{“a blob”}$$

Syntactic models are proper **compilers**.

Target is unrelated to the host language.

$$\vdash_{\mathcal{S}} M : A \quad \xrightarrow{\text{host}} \quad \vdash_{\mathcal{T}} [M] : [\![A]\!] \qquad \text{“an AST”}$$

On Curry-Howard Poetry

General models are more like **interpreters**.

No separation between target vs. host languages

$$\vdash_{\mathcal{S}} M : A \quad \xrightarrow{\text{host}} \quad \vDash_{\mathcal{M}} A \qquad \text{“a blob”}$$

Syntactic models are proper **compilers**.

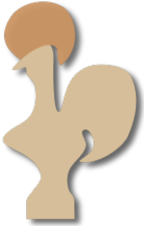
Target is unrelated to the host language.

$$\vdash_{\mathcal{S}} M : A \quad \xrightarrow{\text{host}} \quad \vdash_{\mathcal{T}} [M] : [\![A]\!] \qquad \text{“an AST”}$$

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

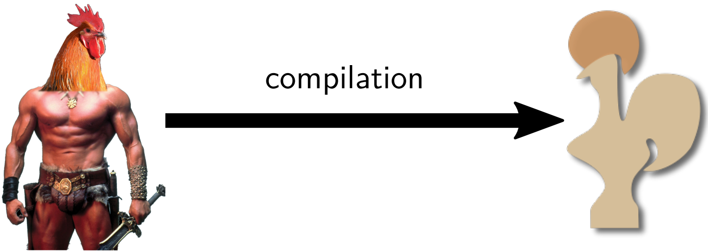
Why Syntactic Models?

compilation →



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

Why Syntactic Models?

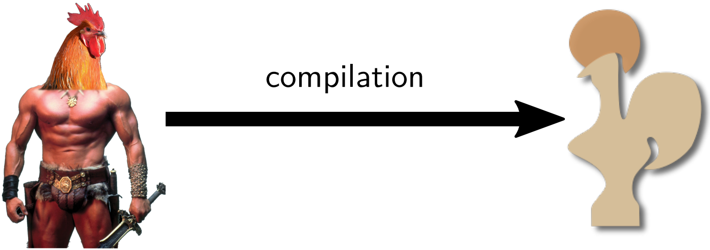


$\vdash_{\mathcal{S}} M : A$ implies $\vdash_{\mathcal{T}} [M] : [\![A]\!]$

Obviously, that's subtle.

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

Why Syntactic Models?



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

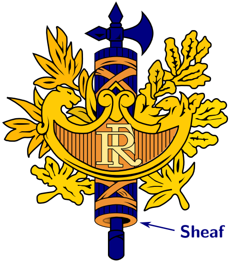
Obviously, that's subtle.

- 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*)
- If $\mathcal{T}$ is CIC, can be implemented in Coq (*software monism*)
- Inherit properties from CIC: computationality, decidability, **implementation**...

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



French Coat of Arms

Persevere Diabolicum

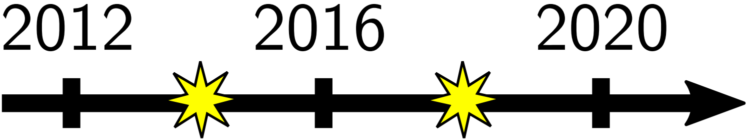
Why the hell am I talking about syntactic presheaves today?

Why the hell am I talking about syntactic presheaves today?

2012	2016	2020
Extending Type Theory with Forcing	The Definitional Side of the Forcing	Russian Constructivism in a Prefascist Theory
(LICS, Jaber, Tabareau, Sozeau)	(LICS, Jaber, Lewertowski, Pédrot, Tabareau, Sozeau)	(LICS, Pédrot)
FAIL	**FAIL**	**YAY?**

Persevere Diabolicum

Why the hell am I talking about syntactic presheaves today?

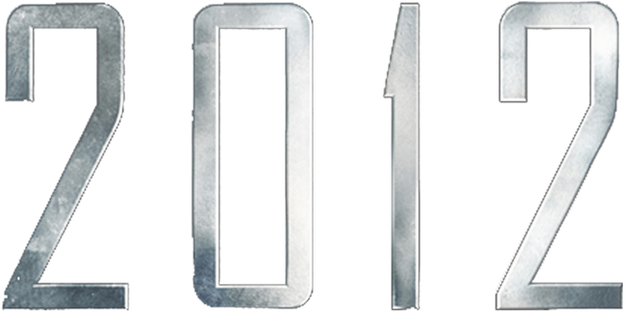


Extending Type Theory with Forcing
(LICS, Jaber, Tabareau, Sozeau)
FAIL

The Definitional Side of the Forcing
(LICS, Jaber, Lewertowski, Pédrot, Tabareau, Sozeau)
FAIL

Russian Constructivism in a Prefascist Theory
(LICS, Pédrot)
YAY?

It is the journey, not the destination



(We were warned.)

Syntactic Presheaves, 2012 Edition

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

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

"Hold my beer!"

Syntactic Presheaves, 2012 Edition

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

"Hold my beer!"

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

Syntactic Presheaves, 2012 Edition

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

"Hold my beer!"

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

What could possibly go wrong?

Close Encounters of the Third Type

Replace **Set** everywhere with CIC.

Close Encounters of the Third Type

Replace **Set** everywhere with CIC.

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

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

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

Close Encounters of the Third Type

Replace **Set** everywhere with CIC.

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

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

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

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 !!!

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

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

Equality is Too Serious a Matter

This **almost** works...

... except that equations are propositional !!!

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

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

You need to introduce rewriting everywhere

"The Coherence Hell"

Equality is Too Serious a Matter

This **almost** works...

... except that equations are propositional !!!

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

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

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 the authors of the 2012 paper, some time before defending PhD.)

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

— Why?

Squaring the Circle

(Me to the authors of the 2012 paper, 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!

Squaring the Circle

(Me to the authors of the 2012 paper, 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.

... but I don't have enough time to present it here.

This is the Left Adjoint, Right?

CBPV is a nice framework to study effects.

... but I don't have enough time to present it here.

Theorem

Kripke models factorize through CBPV.

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

This is the Left Adjoint, Right?

CBPV is a nice framework to study effects.

… but I don't have enough time to present it here.

Theorem

Kripke models factorize through CBPV*.*

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

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

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

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:

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

More Than One Way to Do It

Theorem

Kripke models factorize through CBPV.

Canonical embeddings of λ-calculus into CBPV:

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

Thus, composing the CBV embedding with the “Kripke” interpretation:

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

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

Le Clash

Presheaves are *call-by-value*!

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*!

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

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.

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

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.

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

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

"What about inductive types?"

Robbing Peter to Pay Paul

"What about inductive types?"

The model disproves induction principles...

Robbing Peter to Pay Paul

"What about inductive types?"

The model disproves induction principles...

Not A Suprise

The Kripke translation introduces an effect! (a monotonic reader)

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 examples of dependent effects.

Conclusion of the Episode II

Good News

This is one of the first reasonable examples 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.

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

- 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 "observational 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 "observational 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!

Now the magic trick.

Theorem

A term t is thunkable in the Kripke semantics iff $[t]_p$ is natural in p.

Presheaves Are (Pure) Call-By-Value!

Now the magic trick.

Theorem

A term t is thunkable in the Kripke semantics iff $[t]_p$ is natural in p.

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

- This is a systematic construction that isn't tied to Kripke semantics.
- Unfortunately it relies on extensionality.
- What about CBN?

Syntactic Models For Free

A CBN counterpart of thunkability is **parametricity**

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

A CBN counterpart of thunkability is **parametricity**

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, compatible with intensionality!

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

On Parametric Presheaves

What does parametricity look like on the CBN presheaf model?

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

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 $(x = \lambda q\,\alpha.\, \mathtt{ff})$

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, $\theta_{\mathbb{B}_\varepsilon} \; (\alpha : q \leq p) : \mathbb{B}_\varepsilon \; p \; x \rightarrow \mathbb{B}_\varepsilon \; q \; (\alpha \cdot x)$

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 $(x = \lambda q\, \alpha.\, \mathtt{tt})$ 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

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

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

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

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

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

Playing Cubes

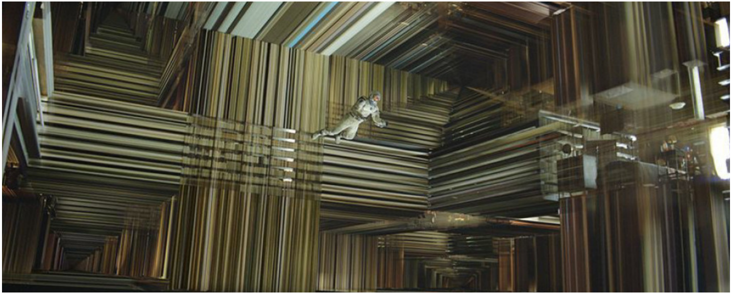
We could solve this with infinite towers of parametricity.

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.

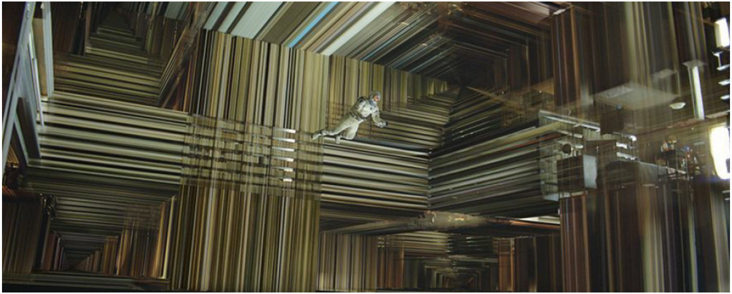


"Oh noes, not cubical type theory again!"

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?

2020

(On the virtues of Authoritarianism.)

It is a Revolution

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

It is a Revolution

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.

It is a Revolution

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**.

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

- A well-behaved subset of `Prop` compatible with HoTT
- It enjoys all good syntactic properties

It is a Revolution

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**.

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

- A well-behaved subset of Prop compatible with HoTT
- It enjoys all good syntactic properties

$\leadsto$ SProp is closed under products.

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

$\leadsto$ Only False is eliminable from SProp into Type.

A Strict Doctrine

Possible Extension

$\mathfrak{s}$CIC additionally allows the elimination of `eq` from `SProp` to `Type`

This gives rise to a **strict equality**, i.e. $\mathfrak{s}$CIC has definitional UIP.

A Strict Doctrine

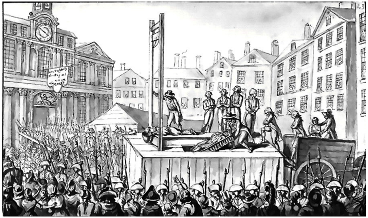
Possible Extension

$\mathfrak{s}$CIC additionally allows the elimination of `eq` from `SProp` to `Type`

This gives rise to a **strict equality**, i.e. $\mathfrak{s}$CIC has definitional UIP.

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 Parametricity

In the parametric presheaf translation

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

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

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

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

Strict Parametricity

In the parametric presheaf translation

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

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

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

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

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

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

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

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

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

Theorem

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

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

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

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 $(-) \Vdash_p \mathcal{A} \quad \subseteq \quad \mathsf{Cone}_p(\mathcal{A}) := \Pi(q : \mathbb{P})(\alpha : q \leq p).\, \mathcal{A}_q$

$\mathbf{Set}$ 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 $(-) \Vdash_p \mathcal{A} \quad \subseteq \quad \mathsf{Cone}_p(\mathcal{A}) := \Pi(q : \mathbb{P})(\alpha : q \leq p).\, \mathcal{A}_q$

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

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

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

MP in Kleene Realizability

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
```

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

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

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

- Kripke semantics $\rightsquigarrow$ global cell $p : \mathbb{N} \rightarrow \mathbb{B}$ where

$$q \leq p \quad := \quad q \text{ pointwise truer than } p$$

- A-translation $\rightsquigarrow$ exceptions of type $A_p := \exists n : \mathbb{N}.\, p\ n = \mathtt{tt}$

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

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

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

- Kripke semantics $\leadsto$ global cell $p : \mathbb{N} \to \mathbb{B}$ where

$$q \leq p \quad := \quad q \text{ pointwise truer than } p$$

- A-translation $\leadsto$ exceptions of type $A_p := \exists n : \mathbb{N}.\, p\ n = \texttt{tt}$

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 define the *Calculus of Constructions with Completeness Principles* as

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

- $\mathbf{Pfs}$ is the prefascist model described before
- $\mathbf{Exn}$ is the exceptional model, a CIC-worthy A-translation

Theorem

If $\mathfrak{s}$CIC *enjoys The Good Properties™ then so does* CCCP.

Pipelining

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

Instead, we define the *Calculus of Constructions with Completeness Principles* as

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

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

Theorem

If $\mathfrak{s}$CIC *enjoys The Good Properties™ then so does* CCCP.

Exn is a very simple syntactic model of CIC

Pick a fixed type $\mathcal{E}$ of **exceptions** in the target theory.

$$\vdash_{\mathcal{S}} A : \square \quad \longrightarrow \quad \vdash_{\mathcal{T}} [\![A]\!]_{\mathcal{E}} : \square \quad + \quad \vdash_{\mathcal{T}} [A]_{\mathcal{E}}^{\varnothing} : \mathcal{E} \to [\![A]\!]_{\mathcal{E}}$$

In particular $\quad [\![\neg A]\!]_{\mathcal{E}} \quad \cong \quad [\![A]\!]_{\mathcal{E}} \to \mathcal{E}$

Monic Fail

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

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

In the the prefascist model over $\mathbb{N} \to \mathbb{B}$, $\qquad \mathcal{E}_p := \Sigma n : \mathbb{N}.\, p\ n = \mathtt{tt}$

Monic Fail

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

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

In the the prefascist model over $\mathbb{N} \to \mathbb{B}$, $\mathcal{E}_p := \Sigma n : \mathbb{N}.\, p\ n = \texttt{tt}$

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

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

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

Monic Fail

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

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

In the the prefascist model over $\mathbb{N} \to \mathbb{B}$, $\mathcal{E}_p := \Sigma n : \mathbb{N}.\, p\ n = \mathtt{tt}$

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

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

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

Theorem

CCCP *validates* MP.

Proof by symbol pushing in CIC + $\mathcal{E}$ by the above and $[\![\neg A]\!]_{\mathcal{E}} \cong [\![A]\!]_{\mathcal{E}} \to \mathcal{E}$.

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.

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

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.

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

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

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 the pure fragment of an effectful CBV language
- We gave a computationally 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 the pure fragment of an effectful CBV language
- We gave a computationally 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:

- Static exceptions as a composition prefascist $+$ exceptions
- This provides a computational extension of CIC that validates MP

Conclusion

On presheaves:

- Presheaves are the pure fragment of an effectful CBV language
- We gave a computationally 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:

- Static exceptions as a composition prefascist $+$ exceptions
- This provides a computational extension of CIC that validates MP

TODO:

- Implement cubical type theory in this model

Thanks for your attention.