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
CIC, the Calculus of Inductive Constructions.
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, 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
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

\[ \vdash M : \mathbb{N} \quad \text{implies} \quad M \equiv S \ldots S 0 \]

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

\[ \vdash M : \mathbb{N} \quad \text{implies} \quad M \equiv S \ldots S 0 \]

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
Our mission

To boldly extend the logical / computational expressivity of CIC
Our mission

To boldly extend the logical / computational expressivity of CIC

〜 we need to design **models** for that.

〜 and ensure they satisfy **The Good Properties™**.

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

To boldly extend the logical / computational expressivity of CIC

we need to design **models** for that.

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, ...
Definition

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

(In what follows we will fix the base category $\mathcal{P}$ once and for all.)
Definition
Let \( P \) be a category. A presheaf over \( P \) is just a functor \( P^{\text{op}} \to \text{Set} \).

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

Theorem
Presheaves with nat. transformations as morphisms form a category \( \text{Psh}(P) \).
Definition
Let $\mathcal{P}$ be a category. A presheaf over $\mathcal{P}$ is just a functor $\mathcal{P}^{\text{op}} \to \text{Set}$.

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

Theorem
Presheaves with nat. transformations as morphisms form a category $\text{Psh}(\mathcal{P})$.

Actually $\text{Psh}(\mathcal{P})$ is even a **topos**!
**Definition**

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

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

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

Bear with me, we will handwave through this in the next slides.
What is $\text{Psh}(\mathcal{P})$?

A presheaf $(A, \theta)$ is given by a family of $\mathcal{P}$-indexed sets $A_p$:

- A family of "restriction morphisms" (a.k.a. monotonicity) $\theta: \prod f p q \to g(\alpha)$.
- $A_p! A_q$ "\(\theta \alpha \)

s.t. given $x \in A_p$, $\alpha \in \mathcal{P}(q, p)$ and $\beta \in \mathcal{P}(r, q)$:

$$\theta(\text{id}_p x) = \theta(\alpha x) = \theta(\beta \theta(\alpha x))$$

"Lowering is compatible with the structure of $\mathcal{P}$."
What is \( \text{Psh}(\mathbb{P}) \)?

**Objects:** A presheaf \((A, \theta_A)\) is given by

- A family of \( \mathbb{P} \)-indexed sets \( A_p : \text{Set} \)
- A family of “restriction morphisms” (a.k.a. monotonicity)

\[
\theta_A : \prod\{p, q \in \mathbb{P}\} (\alpha \in \mathbb{P}(q, p)). A_p \to A_q
\]
What is Psh(\(\mathbb{P}\))?

**Objects:** A presheaf \((A, \theta_A)\) is given by

- A family of \(\mathbb{P}\)-indexed sets \(A_p : \text{Set}\)
- A family of “restriction morphisms” (a.k.a. monotonicity)

\[
\theta_A : \Pi\{p, q \in \mathbb{P}\} (\alpha \in \mathbb{P}(q, p)). A_p \rightarrow A_q
\]

“\(\theta_A \alpha x\) lowers its argument \(x\) along \(\alpha \in \mathbb{P}(q, p)\)”
What is $\text{Psh}(\mathbb{P})$?

**Objects:** A presheaf $(A, \theta_A)$ is given by
- A family of $\mathbb{P}$-indexed sets $A_p : \text{Set}$
- A family of “restriction morphisms” (a.k.a. monotonicity)

$$\theta_A : \Pi\{p, q \in \mathbb{P}\} (\alpha \in \mathbb{P}(q, p)). A_p \to A_q$$

“$\theta_A \alpha x$ lowers its argument $x$ along $\alpha \in \mathbb{P}(q, p)$”

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

$$\theta_A \text{id}_p x \equiv x \quad \theta_A (\beta \circ \alpha) x \equiv \theta_A \beta (\theta_A \alpha x)$$
What is Psh(\(\mathbb{P}\))?

**Objects:** A presheaf \((A, \theta_A)\) is given by
- A family of \(\mathbb{P}\)-indexed sets \(A_p : \text{Set}\)
- A family of “restriction morphisms” (a.k.a. monotonicity)

\[
\theta_A : \prod\{p, q \in \mathbb{P}\} (\alpha \in \mathbb{P}(q, p)). A_p \rightarrow A_q
\]

“\(\theta_A \alpha x\) lowers its argument \(x\) along \(\alpha \in \mathbb{P}(q, p)\)”

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

\[
\theta_A \text{id}_p x \equiv x \quad \theta_A (\beta \circ \alpha) x \equiv \theta_A \beta (\theta_A \alpha x)
\]

“Lowering is compatible with the structure of \(\mathbb{P}\)”
What is $\text{Psh}(\mathcal{P})$?
What is $\text{Psh}(\mathbb{P})$?

**Morphisms:** A morphism from $(A, \theta_A)$ to $(B, \theta_B)$ is given by

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

\[
\theta_B \alpha (f_p x) \equiv f_q (\theta_A \alpha x)
\]
**What is Psh(\(\mathbb{P}\))?**

**Morphisms:** A morphism from \((A, \theta_A)\) to \((B, \theta_B)\) is given by

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

\[
\theta_B \alpha (f_p x) \equiv f_q (\theta_A \alpha x)
\]

"\(f\) is compatible with restriction"
Psh(\mathcal{P}) is a topos.

“Speak, friend, and pullback.”
The Wise Speak Only of What They Know

Psh(ℙ) is a **topos**.

“Speak, friend, and pullback.”

Who cares?

Presheaves actually form a model of CIC.

The Wise Speak Only of What They Know

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

“Speak, friend, and pullback.”

Who cares?

Presheaves actually form a model of CIC.

\[
\vdash A : \square \rightsquigarrow \left[ A \right] \in \text{Psh}(\mathcal{P}) \quad \vdash M : A \rightsquigarrow \left[ M \right] \in \text{Nat}(1, \left[ A \right])
\]

Yet another set-theoretical model!
Let’s have a look at The Good Properties™ we long for.
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?

\[ \vdash M : \mathbb{N} \quad \text{“(C)ZF-implies”} \quad M \equiv S \ldots S 0 \quad 😞 \]
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 s \ldots s \ 0 \quad 😞

**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?

\[ \vdash M : \mathbb{N} \quad \text{“(C)ZF-implies”} \quad M = S \ldots S 0 \quad 😞 \]

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

**Reduction** Never heard of that. What’s syntax already? 😱
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 "(C)ZF-implies" \quad M \equiv S \ldots S 0 \quad 😞

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

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

⇝ Exeunt **Normalization** and **Subject reduction**.
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 S \ldots S 0 \quad 😞

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

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

⇝ 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

\[ \vdash_S M : A \quad \xrightarrow{\text{host}} \quad \models_M A \quad \text{“a blob”} \]
On Curry-Howard Poetry

General models are more like **interpreters**.

No separation between target vs. host languages

\[ \vdash_S M : A \xrightarrow{\text{host}} \models_M A \quad \text{“a blob”} \]

Syntactic models are proper **compilers**.

Target is unrelated to the host language.

\[ \vdash_S M : A \xrightarrow{\text{host}} \vdash_T [M] : [A] \quad \text{“an AST”} \]
General models are more like **interpreters**.

No separation between target vs. host languages

\[ \vdash_S M : A \xrightarrow{\text{host}} \models_M A \quad \text{“a blob”} \]

Syntactic models are proper **compilers**.

Target is unrelated to the host language.

\[ \vdash_S M : A \xrightarrow{\text{host}} \vdash_{\mathcal{T}} [M] : [A] \quad \text{“an AST”} \]

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

\[ \vdash_S M : A \] implies \[ \vdash_T [M] : [A] \]

Obviously, that's subtle.

The translation 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 \( T \) is CIC, can be implemented in Coq (software monism)

Inherit properties from CIC: computationality, decidability, implementation...
Why Syntactic Models?

\[ \vdash_S M : A \implies \vdash_T [M] : [A] \]

Obviously, that’s subtle.

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

\[ \vdash_S M : A \]

\[ \text{implies} \]

\[ \vdash_T [M] : [A] \]

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 \(T\) is CIC, can be implemented in Coq (software monism)
- Inherit properties from CIC: computationally, decidability, implementation...
“Is it possible to see the presheaf construction as a syntactic model?”
Persevere Diabolicum

Why the hell am I talking about syntactic presheaves today?
Persevere Diabolicum

Why the hell am I talking about syntactic presheaves today?

2012
Extending Type Theory with Forcing
(LICS, Jaber, Tabareau, Sozeau)
FAIL

2016
The Definitional Side of the Forcing
(LICS, Jaber, Lewertowski, Pédrot, Tabareau, Sozeau)
FAIL

2020
Russian Constructivism in a Prefascist Theory
(LICS, Pédrot)
YAY?

It is the journey, not the destination.

P.-M. Pédrot (INRIA, Gallinette team)
Why the hell am I talking about syntactic presheaves today?

2012
Extending Type Theory with Forcing (LICS, Jaber, Tabareau, Sozeau)

2016
The Definitional Side of the Forcing (LICS, Jaber, Lewertowski, Pédrot, Tabareau, Sozeau)

2020
Russian Constructivism in a Prefascist Theory (LICS, Pédrot)

FAIL FAIL YAY?

It is the journey, not the destination
(We were warned.)
“A presheaf is just a functor $\mathbb{P}^{\text{op}} \to \text{Set}$.”
“A presheaf is just a functor $\mathbb{P}^{\text{op}} \to \text{Set}$.”

“Hold my beer!”
“A presheaf is \textit{just} a functor $\mathbb{P}^{\text{op}} \to \text{Set}$.”

“Hold my beer!”

Replace \texttt{Set} everywhere with CIC.
“A presheaf is just a functor $\mathbb{P}^{\text{op}} \to \text{Set}$.”

“Hold my beer!”

Replace $\text{Set}$ 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 \textit{Set} everywhere with CIC.

\[
\begin{align*}
\text{Cat: } \square & := \begin{cases} 
\mathbb{P}: \square \\
\leq: \mathbb{P} \to \mathbb{P} \to \square \\
id: \Pi_p. p \leq p \\
o: \Pi_p q r. p \leq q \to q \leq r \to p \leq r \\
eqn: \ldots;
\end{cases} \\
\text{Psh: } \square & := \begin{cases} 
\mathbf{A}: \mathbb{P} \to \square \\
\theta_\mathbf{A}: \Pi(p q : \mathbb{P}) (\alpha : q \leq p). \mathbf{A}_p \to \mathbf{A}_q \\
eqn: \ldots;
\end{cases} \\
\text{El } (\mathbf{A}, \theta_\mathbf{A}, e): \square & := \begin{cases} 
el: \Pi(p : \mathbb{P}). \mathbf{A} \ p \\
eqn: \ldots;
\end{cases}
\end{align*}
\]
Replace \texttt{Set} everywhere with CIC.

\begin{align*}
\text{Cat : } &\Box &:=&\begin{cases}
P : \Box \\
\preceq : P \to P \to \Box \\
id : \Pi p . p \preceq p \\
o : \Pi p q r . p \preceq q \to q \preceq r \to p \preceq r \\
ev \text{n} : \ldots ;
\end{cases} \\
Psh : &\Box &:=&\begin{cases}
A : P \to \Box \\
\theta_A : \Pi (p q : P) (\alpha : q \preceq p) . A_p \to A_q \\
ev \text{n} : \ldots ;
\end{cases} \\
\text{El } (A , \theta_A , e) : &\Box &:=&\begin{cases}
el : \Pi (p : P) . A_p \\
ev \text{n} : \ldots ;
\end{cases}
\end{align*}

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 \textbf{almost} works...

... except that equations are propositional !!!

\[
\text{El} \left( A, \theta_A, e \right) : \square := \left\{ \begin{array}{l}
e1 : \Pi(p : P). A \ p \\
eqn : \ldots;
\end{array} \right\}
\]

\[\vdash_{\text{CIC}} M \equiv N \quad \not\leftrightarrow \quad \vdash [M] \equiv [N]\]

\[\vdash_{\text{CIC}} M \equiv N \quad \rightarrow \quad \vdash e : [M] = [N]\]
Equality is Too Serious a Matter

This **almost** works...

... except that equations are propositional !!!

\[
\text{El } (A, \theta_A, e) : \square := \left\{ \begin{array}{l}
\text{el} : \Pi(p : P). A \ p \\
\text{eqn} : \ldots ;
\end{array} \right\}
\]

\[
\vdash_{\text{CIC}} M \equiv N \not\rightarrow \vdash [M] \equiv [N]
\]

\[
\vdash_{\text{CIC}} M \equiv N \rightarrow \vdash e : [M] = [N]
\]

😱 You need to introduce rewriting everywhere 😱

“The Coherence Hell”
Equality is Too Serious a Matter

This almost works...

... except that equations are propositional !!!

\[ \text{El} (A, \theta_A, e) : \square \quad := \quad \{ \text{el} : \Pi(p : P). A \ p \ \} \]

\[ \vdash_{\text{CIC}} M \equiv N \quad \not\leftrightarrow \quad \vdash [M] \equiv [N] \]

\[ \vdash_{\text{CIC}} M \equiv N \quad \rightarrow \quad \vdash e : [M] = [N] \]

😱 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.

\[
\Gamma \vdash e : M = N \\
\Gamma \vdash M \equiv N
\]
Extensional Type Theory (ETT) is defined by *Santa Claus conversion*.

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

- Arguably better than ZFC ("constructive")
That Was Not My Intension

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

\[
\Gamma \vdash e : M = N \\
\hline
\Gamma \vdash M \equiv N
\]

- Arguably better than ZFC (“constructive”)
- ... but undecidable type checking
- ... no computation, e.g. $\beta$-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*.

\[
\begin{align*}
\Gamma \vdash e : M = N \\
\hline
\Gamma \vdash M \equiv N
\end{align*}
\]

- Arguably better than ZFC ("constructive")
- ... but undecidable type checking
- ... no computation, e.g. $\beta$-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

\[ \Gamma \vdash e : M = N \]

Arguably better than ZFC ("constructive")... but undecidable type checking... no computation, e.g. \( \beta \)-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\(^\dagger\).

\(^\dagger\) To be more precise, I believe that ETT is not really a type theory.
2016

(Make conversion great again, and break everything else.)
(Me to the authors of the 2012 paper, some time before defending PhD.)

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

— Why?
(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 \textit{call-by-value}!

... and you’re trying to interpret a \textit{call-by-name} language!
(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 interpret a call-by-name language!

— What on earth does that even mean?
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.
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.*

\[
A \text{ value type} \quad \mapsto \quad [A]^v : \text{Fun}(\mathcal{P}^{\text{op}}, \text{Set})
\]

\[
X \text{ computation type} \quad \mapsto \quad [X]^c : |\mathcal{P}| \rightarrow \text{Set}
\]
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** CFPV.

<table>
<thead>
<tr>
<th>Value type</th>
<th>Computation type</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A$</td>
<td>$[A]^v : \text{Fun}(\mathbb{P}^{op}, \text{Set})$</td>
</tr>
<tr>
<td>$X$</td>
<td>$[X]^c :</td>
</tr>
</tbody>
</table>

\[
[U X]^v_p := \Pi(q : \mathbb{P})(\alpha : q \leq p). [X]^c_q \quad \text{(free functoriality)}
\]

\[
\theta_{[U X]^v}(\alpha : q \leq p)(x : [U X]^v_p) := \lambda(r : \mathbb{P})(\beta : r \leq q). x \ r \ (\alpha \circ \beta)
\]
More Than One Way to Do It

**Theorem**

*Kripke models factorize through CBPV.*

Canonical embeddings of $\lambda$-calculus into CBPV:

- **CBN**
  \[(\sigma \to \tau)^N := U \sigma^N \to \tau^N\] (a computation type)

- **CBV**
  \[(\sigma \to \tau)^V := U (\sigma^V \to \mathcal{F} \tau^V)\] (a value type)
More Than One Way to Do It

**Theorem**

*Kripke models factorize through CBPV.*

Canonical embeddings of λ-calculus into CBPV:

- **CBN** \((\sigma \rightarrow \tau)^N := U \sigma^N \rightarrow \tau^N\) (a computation type)
- **CBV** \((\sigma \rightarrow \tau)^V := U (\sigma^V \rightarrow F \tau^V)\) (a value type)

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

\[
\llbracket (\sigma \rightarrow \tau)^V \rrbracket_p^V := \Pi(q : P)(\alpha : q \leq p). [\sigma^V]_q^V \rightarrow [\tau^V]_q^V
\]
Theorem

*Kripke models factorize through CBPV.*

Canonical embeddings of $\lambda$-calculus into CBPV:

\[
\begin{align*}
\text{CBN} & \quad (\sigma \to \tau)^N := \mathcal{U} \sigma^N \to \tau^N \quad \text{(a computation type)} \\
\text{CBV} & \quad (\sigma \to \tau)^V := \mathcal{U} (\sigma^V \to \mathcal{F} \tau^V) \quad \text{(a value type)}
\end{align*}
\]

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

\[
\begin{align*}
\llbracket (\sigma \to \tau)^V \rrbracket_p^v & := \Pi(q : \mathbb{P})(\alpha : q \leq p) \cdot [\sigma^V]_q^v \to [\tau^V]_q^v
\end{align*}
\]

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

\[
t \equiv_{\beta_v} u \quad \rightarrow \quad t^V \equiv_{\text{CBPV}} u^V \quad \rightarrow \quad [t^V]_p \equiv_T [u^V]_p
\]
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

\[t \equiv_{\beta_v} u \quad \rightarrow \quad t^V \equiv_{\text{CBPV}} u^V \quad \rightarrow \quad [t^V]_p \equiv_T [u^V]_p\]

Type theory is *call-by-name*!
Presheaves are call-by-value!

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

\[(\lambda x \cdot t) \equiv_{\beta v} t\{x := V\}\]

because

\[t \equiv_{\beta v} u \quad \rightarrow \quad t^V \equiv_{\text{CBPV}} u^V \quad \rightarrow \quad [t^V]_p \equiv_T [u^V]_p\]

Type theory is call-by-name!

\[\frac{\Gamma \vdash M : B \quad \Gamma \vdash A \equiv_\beta B}{\Gamma \vdash M : A} \text{ (Conv)}\]
Presheaves are \textit{call-by-value}!

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

\[
(\lambda x. \, t) \stackrel{\beta v}{\equiv} t\{x := V\}
\]

because

\[
t \stackrel{\beta v}{\equiv} u \quad \rightarrow \quad t^V \equiv_{\text{CBPV}} u^V \quad \rightarrow \quad [t^V]_p \equiv_T [u^V]_p
\]

Type theory is \textit{call-by-name}!

\[
\frac{\Gamma \vdash M : B \quad \Gamma \vdash A \equiv_{\beta} B}{\Gamma \vdash M : A} \quad \text{(Conv)}
\]

Folklore

\textbf{Call-by-name is not call-by-value!}
Easy solution! Pick the CBN decomposition instead.

\[
\left[ (\sigma \rightarrow \tau)^N \right]_p^c := (\Pi (q : P) (\alpha : q \leq p)). \left[ \sigma^N \right]_q^c \rightarrow \left[ \tau^N \right]_p^c
\]
If There is No Solution, There is No Problem

Easy solution! Pick the CBN decomposition instead.

\[ [(σ → τ)^N]_p^c := (Π(q : P)(α : q ≤ p). [σ^N]_q^c) → [τ^N]_p^c \]

This adapts straightforwardly to the dependently-typed setting.
If There is No Solution, There is No Problem

Easy solution! Pick the CBN decomposition instead.

\[ [(\sigma \rightarrow \tau)^N]^c_p := (\Pi (q : P)(\alpha : q \leq p). [\sigma^N]_q^c) \rightarrow [\tau^N]^c_p \]

This adapts straightforwardly to the dependently-typed setting.

Theorem (Jaber & al. 2016)

*There is a syntactic “presheaf model” of $\text{CC}^\omega$ into CIC.*

where $\text{CC}^\omega$ is CIC without inductive types.
If There is No Solution, There is No Problem

Easy solution! Pick the CBN decomposition instead.

\[
[(\sigma \rightarrow \tau)^N]_p^c := (\Pi(q : P) (\alpha : q \leq p). [\sigma^N]_q^c) \rightarrow [\tau^N]_p^c
\]

This adapts straightforwardly to the dependently-typed setting.

Theorem (Jaber & al. 2016)

There is a syntactic “presheaf model” of \( \text{CC}^\omega \) into \( \text{CIC} \).

where \( \text{CC}^\omega \) is \( \text{CIC} \) without inductive types.

\[
\begin{align*}
P.M. \text{ Pédrot (INRIA, Gallinette team)} & \quad \text{All your base categories are belong to us} \\
31/08/2020 & \quad 26 / 54
\end{align*}
\]
“What about inductive types?”
“What about inductive types?”

The model disproves induction principles...
“What about inductive types?”

The model disproves induction principles...

Not A Suprise

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

---

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

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

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

P.-M. Pédrot (INRIA, Gallinette team)
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 \( \lambda \)-calculus.

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

Isn't this some ad-hoc trick?
Categorical presheaves form a model of the whole $\lambda$-calculus.

... in particular, it does interpret full $\beta$-conversion (although extensionally).

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

\[
[A \rightarrow B]_p := f : \Pi(q \leq p). [A]_q \rightarrow [B]_q \quad \text{s.t.}
\]

\[
\begin{align*}
\frac{[A]_q \xrightarrow{f_q \alpha} [B]_q}{\theta_A \beta} \quad & \quad \frac{[A]_r \xrightarrow{f_r (\alpha \circ \beta)} [B]_r}{\theta_B \beta}
\end{align*}
\]
Categorical presheaves form a model of the whole $\lambda$-calculus.

... in particular, it does interpret full $\beta$-conversion (although extensionally).

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

\[
[A \to B]_p := f: \Pi(q \leq p). \, [A]_q \to [B]_q \quad \text{s.t.}
\]

\[
\begin{array}{ccc}
[A]_q & \xrightarrow{f_q} & [B]_q \\
\downarrow \theta_A \beta & & \downarrow \theta_B \beta \\
[A]_r & \xrightarrow{f_r \circ \beta} & [B]_r
\end{array}
\]

- We do not have an equivalent in our CBN interpretation
- Isn’t this some ad-hoc trick?
Consider an effectful CBV $\lambda$-calculus.

**Definition (Führmann ’99)**

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

$$\text{let } x := t \text{ in } \lambda().x \equiv \lambda().t$$

Thunkability intuitively captures “observational purity” and does so generically, i.e., it does not depend on the effect considered. In a pure language, all terms are thunkable.

**Theorem (Folklore Realizability)**

The sublanguage of hereditarily thunkable terms satisfies full $\beta$-conversion.
Consider an effectful CBV $\lambda$-calculus.

**Definition (Führmann ’99)**

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

$$\text{let } x := t \text{ in } \lambda(). x \equiv \lambda(). t$$

- 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
Consider an effectful CBV $\lambda$-calculus.

**Definition (Führmann ’99)**

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

$$\text{let } x := t \text{ in } \lambda(). x \equiv \lambda(). t$$

- 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 $\beta$-conversion.*

$$f \Vdash A \rightarrow B := \forall u. \ u \Vdash A \rightarrow f u \text{ thk} \land f u \Vdash B$$
Now the magic trick.

**Theorem**

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

$\text{Presheaves Are (Pure) Call-By-Value!}$
Now the magic trick.

**Theorem**

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

$\text{Psh}(\mathcal{P})$ 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?
A CBN counterpart of thunkability is **parametricity**

Bernardy-Lasson '11

There is a well-known parametricity interpretation for type theory

\[ \Gamma \vdash_{\text{CIC}} M : A \quad \rightarrow \quad [\Gamma]_\varepsilon \vdash_{\text{CIC}} [M]_\varepsilon : [A]_\varepsilon \quad M \]

where

\[ [\cdot]_\varepsilon := \cdot \quad \text{and} \quad [\Gamma, x : A]_\varepsilon := [\Gamma]_\varepsilon, x : A, x_\varepsilon : [A]_\varepsilon \quad x \]
A CBN counterpart of thunkability is **parametricity**

Bernardy-Lasson ’11

There is a well-known parametricity interpretation for type theory

\[
\Gamma \vdash_{\text{CIC}} M : A \quad \rightarrow \quad \Gamma[\varepsilon] \vdash_{\text{CIC}} [M][\varepsilon] : [A][\varepsilon] M
\]

where \(\cdot[\varepsilon] := \cdot\) and \(\Gamma, x : A[\varepsilon] := \Gamma[\varepsilon], x : A, x[\varepsilon] : [A][\varepsilon] x\)

Turns out it is a syntactic model, compatible with intensionality!

It is a special case of a more general **internal realizability** interpretation.
What does parametricity look like on the CBN presheaf model?

\[ x : \mathbb{B} \rightarrow \left\{ \begin{array}{l} x : (\Pi (q : \mathbb{P})(\alpha : q \leq p). \mathbb{B}) \\ x_\varepsilon : \mathbb{B}_\varepsilon p x \end{array} \right\} \]
What does parametricity look like on the CBN presheaf model?

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

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

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

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)
What does parametricity look like on the CBN presheaf model?

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

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

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

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, \( \theta_{\mathbb{B}_{\varepsilon}} (\alpha : q \leq p) : \mathbb{B}_{\varepsilon} \ p \ x \rightarrow \mathbb{B}_{\varepsilon} \ q \ (\alpha \cdot x) \)
What does parametricity look like on the CBN presheaf model?

\[ x : B \rightarrow \left\{ \begin{array}{l} x : (\Pi (q : P) (\alpha : q \leq p) . B) \\ x_{\varepsilon} : B_{\varepsilon} p x \end{array} \right\} \]

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

1. \( B_{\varepsilon} p x \text{ iff } (x = \lambda q \alpha. \text{tt}) \text{ or } (x = \lambda q \alpha. \text{ff}) \)

2. in a unique way, i.e. \( b_1, b_2 : B_{\varepsilon} p x \vdash b_1 = b_2 \) (i.e. a HoTT proposition)

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

3. That is, \( \theta_{B_{\varepsilon}} (\alpha : q \leq p) : B_{\varepsilon} p x \rightarrow B_{\varepsilon} q (\alpha \cdot x) \)

4. with further definitionnal functoriality to avoid coherence issues
What does parametricity look like on the CBN presheaf model?

\[ x : \mathbb{B} \rightarrow \left\{ \begin{array}{ll} x : (\Pi(q : \mathbb{P})(\alpha : q \leq p). \mathbb{B}) \\ x_{\varepsilon} : \mathbb{B}_{\varepsilon} p x \end{array} \right. \]

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

1. \( \mathbb{B}_{\varepsilon} p x \text{ iff } (x = \lambda q \alpha. \text{tt}) \text{ or } (x = \lambda q \alpha. \text{ff}) \)
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. 😱
This is exactly the CBV vs. CBN conundrum \textbf{one level higher}

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

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

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

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

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

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

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

It is not possible to get both at the same time in CIC!
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.)
Essentially, we were blocked on this issue since then. When suddenly...
Essentially, we were blocked on this issue since then. When suddenly...

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


They introduce a new sort \( \text{SProp} \) of strict propositions.

\[
M, N : A : \text{SProp} \quad \rightarrow \quad \vdash M \equiv N
\]

- A well-behaved subset of \( \text{Prop} \) compatible with HoTT
- It enjoys all good syntactic properties
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.

They introduce a new sort $\text{SProp}$ of strict propositions.

\[
M, N : A : \text{SProp} \quad \Rightarrow \quad \vdash M \equiv N
\]

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

$\leadsto$ $\text{SProp}$ is closed under products.

\[
\vdash A : \Box, \quad x : A \vdash B : \text{SProp} \quad \Rightarrow \quad \vdash \Pi(x : A). B : \text{SProp}
\]

$\leadsto$ Only $\text{False}$ is eliminable from $\text{SProp}$ into $\text{Type}$. 
A Strict Doctrine

Possible Extension

§CIC additionally allows the elimination of eq from SProp to Type

This gives rise to a **strict equality**, i.e. §CIC has definitional UIP.
Possible Extension

$\mathsf{CIC}$ additionally allows the elimination of $\text{eq}$ from $\mathsf{SProp}$ to $\mathsf{Type}$

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

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

... the authoritarian $\mathsf{CIC}$ 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**

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

where critically $[A]_\varepsilon \ p \ x : SProp$. 
Strict Parametricity

In the parametric presheaf translation

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

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

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

where critically $[A]_\varepsilon p x : SProp$.

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

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

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

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.
- Unsurprisingly, UIP is required to interpret universes (tricky!).
§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. §CIC + an impredicative universe is not SN
- Hoping that SN holds in the predicative case, decidability follows
§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. §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 §CIC for free.
Thus, the prefascist model can also be described set-theoretically.
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 \( (-) \vdash_p \mathcal{A} \) \( \subseteq \) \( \text{Cone}_p(\mathcal{A}) := \Pi(q : \mathbb{P})(\alpha : q \leq p). \mathcal{A}_q \).
Set is a model of $\Sigma$CIC

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

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

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

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

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

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

where

$$\text{El}_p \mathcal{A} := \{x : \text{Cone}_p(\mathcal{A}) \mid \forall q (\alpha : q \leq p). (\alpha \cdot x) \models_q \mathcal{A}\}$$

$$\text{app}_p(f, x) := \lambda q (\alpha : q \leq p). f_q (\alpha \cdot x)$$
Theorem

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

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

Theorem

As categories, $\text{Psh}(\mathbb{P})$ and $\text{Pfs}(\mathbb{P})$ are equivalent.
Prefascist sets over $\mathbb{P}$ form a category $\text{Pfs}(\mathbb{P})$ with \textit{definitional} laws.

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

Proving this requires extensionality principles!

- Hence, in a set-theoretical meta, both describe the same objects
- Yet, $\text{Pfs}(\mathbb{P})$ 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 $\text{Pfs}(\mathbb{P})$ with \textit{definitional} laws.

Theorem

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

Proving this requires extensionality principles!

- Hence, in a set-theoretical meta, both describe the same objects
- Yet, $\text{Pfs}(\mathbb{P})$ 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
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 \textbf{and} Bishop:

**Markov's Principle (MP)**

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

- A lot of equivalent statements, e.g. a TM that doesn’t loop terminates
- Semi-classical: $\text{HA}^\omega \subsetneq \text{HA}^\omega + \text{MP} \subsetneq \text{PA}^\omega$
- 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)

\[ \forall (f : \mathbb{N} \to \mathbb{B}). \neg \neg (\exists n : \mathbb{N}. f n = \text{tt}) \rightarrow \exists n : \mathbb{N}. f n = \text{tt} \]

- A lot of equivalent statements, e.g. a TM that doesn’t loop terminates
- Semi-classical: $\text{HA}^\omega \subsetneq \text{HA}^\omega + \text{MP} \subsetneq \text{PA}^\omega$
- 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

\[ \forall (f : \mathbb{N} \to \mathbb{B}). \neg \neg (\exists n : \mathbb{N}. f n = \text{tt}) \rightarrow \exists n : \mathbb{N}. f n = \text{tt} \]

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

Proving \( \vdash \text{mp} \) needs \( \vdash \text{mp} \) 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...
MP in Kleene Realizability

Let's look at the realizer

\[ \forall (f : \mathbb{N} \rightarrow \mathbb{B}). \neg (\exists n : \mathbb{N}. f \ n = \text{tt}) \rightarrow \exists n : \mathbb{N}. f \ n = \text{tt} \]

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
Let's look at the realizer

\[ \forall (f : \mathbb{N} \to \mathbb{B}). \neg \neg (\exists n : \mathbb{N}. f\ n = \text{tt}) \rightarrow \exists n : \mathbb{N}. f\ n = \text{tt} \]

\[
\text{let mp f _ := } \\
\text{let n := ref 0 in } \\
\text{while true do } \\
\text{if f !n then return n else n := n + 1 } \\
\text{done}
\]

Proving \(\text{mp} \vdash \text{MP}\) needs \(\text{MP}\) 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!
Not one, but at least **two** alternatives!

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

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

- $A$-translation $\leadsto$ exceptions of type $A_p := \exists n : \mathbb{N}. p n = \text{tt}$
Not one, but at least **two** alternatives!

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

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

- Kripke semantics $\rightsquigarrow$ global cell $p : \mathbb{N} \to \mathbb{B}$ where
  \[
  q \leq p \quad := \quad q \text{ pointwise truer than } p
  \]
- $A$-translation $\rightsquigarrow$ exceptions of type $A_p := \exists n : \mathbb{N}. p n = \text{tt}$

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

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

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

\[
\text{CCCP} \ (\supset CIC) \xrightarrow{\text{Exn}} \ CIC + \mathcal{E} \xrightarrow{\text{Pfs}} \mathcal{sCIC}
\]

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

**Theorem**

*If $\mathcal{sCIC}$ 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

\[
\text{CCCP } (\supseteq \text{CIC}) \xrightarrow{\text{Exn}} \text{CIC} + \mathcal{E} \xrightarrow{\text{Pfs}} \mathcal{S}\text{CIC}
\]

- \text{Pfs} is the prefascist model described before
- \text{Exn} is the exceptional model, a CIC-worthy \(A\)-translation

**Theorem**

*If \(\mathcal{S}\text{CIC} \) enjoys *The Good Properties™* then so does CCCP.*

\textbf{Exn} is a very simple syntactic model of CIC

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

\[
\vdash_{\mathcal{S}} A : \Box \quad \rightarrow \quad \vdash_{\mathcal{T}} [A]_{\mathcal{E}} : \Box \quad + \quad \vdash_{\mathcal{T}} [A]_{\mathcal{E}}^\Box : \mathcal{E} \to [A]_{\mathcal{E}}
\]

In particular

\[
[\neg A]_{\mathcal{E}} \cong [A]_{\mathcal{E}} \to \mathcal{E}
\]
We perform the exceptional translation over an **exotic** type of exceptions

\[
\text{CCCP} \xrightarrow{\text{Exn}} \text{CIC} + \mathcal{E} \xrightarrow{\text{Pfs}} \mathcal{s}\text{CIC}
\]

In the prefascist model over \( \mathbb{N} \to \mathbb{B} \),

\[
\mathcal{E}_p := \sum n : \mathbb{N}. \ p \ n = \text{tt}
\]
We perform the exceptional translation over an **exotic** type of exceptions

$$\text{CCCP} \xrightarrow{\text{Exn}} \text{CIC} + \mathcal{E} \xrightarrow{\text{Pfs}} \mathcal{S}\text{CIC}$$

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

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

- $\text{local} \ : \ (\mathbb{N} \to \mathbb{B}) \to \Box \to \Box$
- $[\text{local } \varphi \ A]_p \ := \ [A]_{p \land \varphi}$

- $\text{return} : A \to \text{local } \varphi \ A$
- $\text{local} \text{ commutes to arrows and positive types}$
- $\text{local } \varphi \ \mathcal{E} \ \cong \ \mathcal{E} + (\Sigma n : \mathbb{N}. \varphi \ n = \text{tt})$
We perform the exceptional translation over an **exotic** type of exceptions.

\[
\text{CCCP} \xrightarrow{\text{Exn}} \text{CIC} + \mathcal{E} \xrightarrow{\text{Pfs}} 5\text{CIC}
\]

In the prefascist model over \(\mathbb{N} \rightarrow \mathbb{B}\),

\[
\mathcal{E}_p := \sum n : \mathbb{N}. p \ n = \text{tt}
\]

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

\[
\text{local} : (\mathbb{N} \rightarrow \mathbb{B}) \rightarrow \Box \rightarrow \Box
\]

\[
[\text{local } \varphi \ A]_p \triangleq [A]_p \land \varphi
\]

- \(\text{return} : A \rightarrow \text{local } \varphi \ A\)
- \(\text{local} \text{ commutes to arrows and positive types}\)
- \(\text{local } \varphi \ \mathcal{E} \cong \mathcal{E} + (\sum n : \mathbb{N}. \varphi \ n = \text{tt})\)

**Theorem**

**CCCP validates MP.**

Proof by symbol pushing in \(\text{CIC} + \mathcal{E}\) by the above and \([\neg A]_{\mathcal{E}} \cong [A]_{\mathcal{E}} \rightarrow \mathcal{E}\).
A Computational Analysis of MP

Every time we go under local we get new exceptions!

\[
\text{local } \varphi \mathcal{E} \quad \cong \quad \mathcal{E} + (\sum n : \mathbb{N}. \varphi \ n = \text{tt})
\]

return is a delimited continuation prompt / static exception binder.
Every time we go under local we get new exceptions!

\[ \text{local } \varphi \ E \quad \cong \quad E + (\sum n: \mathbb{N}. \varphi \ n = \text{tt}) \]

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

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

\[
\begin{align*}
\text{mp } (p : \neg \neg (\exists n. f \ n = \text{tt})) := \\
\text{try}_\alpha \perp_e (p \ (\lambda k. k \ (\lambda n. \text{raise}_\alpha \ n))) \text{ with } \alpha \ n \mapsto n
\end{align*}
\]
Every time we go under `local` we get new exceptions!

\[
\text{local } \varphi \ E \cong E + (\Sigma n : \mathbb{N}. \varphi \ n = \text{tt})
\]

Return is a **delimited continuation** prompt / static exception binder.

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

\[
\text{mp } (p : \neg \neg (\exists n. f \ n = \text{tt})) := \\
\text{try}_{\alpha} \perp_e (p (\lambda k. k (\lambda n. \text{raise}_{\alpha} \ n))) \text{ with } \alpha \ n \mapsto n
\]

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.
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 §CIC enjoys this (†)
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 $\mathcal{CIC}$ enjoys this (†)

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 $\text{CIC}$ enjoys this ($\uparrow$)

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.