Russian Constructivism in a Prefascist Theory

Pierre-Marie Pédrot

Gallinette, INRIA

LICS’20
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
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
Our mission: to boldly extend CIC with new principles
Extending Coq

Our mission: to boldly extend CIC with new principles

⇝ we need to design models for that.
⇝ and ensure they satisfy the good properties.

- Consistency
- Canonicity
- Decidable type-checking
- Strong normalization
Our mission: to boldly extend CIC with new principles

we need to design models for that.
and ensure they satisfy the good properties.

- Consistency
- Canonicity
- Decidable type-checking
- Strong normalization

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

**Presheaves!**

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

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

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

- A family of $\mathcal{P}$-indexed sets $A_p : \text{Set}$
- Restriction morphisms $\theta_A : \prod_{p,q} (\alpha \in \mathcal{P}(q, p)). A_p \to A_q$ (+ functoriality)

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

- A family of $\mathcal{P}$-indexed functions $f_p : A_p \to B_p$ which is natural in $p$
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.)

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

**Objects:** A presheaf $(A, \theta_A)$ is given by
- A family of $\mathbb{P}$-indexed sets $A_p : \text{Set}$
- Restriction morphisms $\theta_A : \prod_{p,q} (\alpha \in \mathbb{P}(q,p)). A_p \to A_q$ (+ functoriality)

**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 in $p$

Theorem

$\text{Psh}(\mathbb{P})$ is a model of CIC.
Cantor’s Hell

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?

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

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

**Consistency** There is no proof of False. ☺

**Canonicity** Closed integers are integers... are they?

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

**Implementability** Type-checking is **not** decidable. 😞
Cantor’s Hell

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

**Consistency** There is no proof of False. 😊

**Canonicity** Closed integers are integers... are they?

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

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

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

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

**Consistency** There is no proof of False. ☺

**Canonicity** Closed integers are integers... are they?

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

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

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

**Phenomenological Law**

Set-theoretical models suck.
Down With Semantics

Instead

**Syntactic Models**

\[ \vdash_S M : A \quad \text{implies} \quad \vdash_T [M] : [A] \]

Does not require non-type-theoretical foundations (monism)
Can be implemented in Coq (software monism)
Automatically inherit the good properties from CIC
Instead

Syntactic Models

\[ \vdash_S M : A \]

implies

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

- Does not require non-type-theoretical foundations (*monism*)
- Can be implemented in Coq (*software monism*)
- Automatically inherit the good properties from CIC
Is it possible to see presheaves as a syntactic model?
Is it possible to see presheaves as a syntactic model?

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?
Is it possible to see presheaves as a syntactic model?

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
(We were warned.)
“A presheaf is just a functor $\mathbb{P}^{\text{op}} \rightarrow \text{Set}$.”

Easy peasy: just replace $\text{Set}$ everywhere with CIC.
“A presheaf is just a functor $\mathbb{P}^{\text{op}} \to \text{Set}$.”

Easy peasy: just replace $\text{Set}$ everywhere with CIC.

$$\text{Cat} : \Box := \begin{cases} \mathbb{P} : \Box \\ \leq : \mathbb{P} \to \mathbb{P} \to \Box \\ \text{id} : \Pi_{p.} p \leq p \\ \circ : \Pi_{p q r.} p \leq q \to q \leq r \to p \leq r \\ \text{eqn} : \ldots; \end{cases}$$

$$\text{Psh} : \Box := \begin{cases} A : \mathbb{P} \to \Box \\ \theta_A : \Pi{(p q : \mathbb{P}) (\alpha : q \leq p).} A_p \to A_q \\ \text{eqn} : \ldots; \end{cases}$$
"A presheaf is just a functor $\mathbb{P}^{\text{op}} \to \textbf{Set}.$"

Easy peasy: just replace $\textbf{Set}$ everywhere with CIC.

\[
\begin{align*}
\text{Cat} : & \quad \square := \\
\text{Psh} : & \quad \square := \\
\begin{aligned}
\mathbb{P} & : \square \\
\leq & : \mathbb{P} \to \mathbb{P} \to \square \\
\text{id} & : \Pi_p. p \leq p \\
\circ & : \Pi p q r. p \leq q \to q \leq r \to p \leq r \\
\text{eqn} & : \ldots;
\end{aligned} \\
\begin{aligned}
\theta_A & : \Pi (p q : \mathbb{P}) (\alpha : q \leq p). A_p \to A_q \\
\text{eqn} & : \ldots;
\end{aligned}
\end{align*}
\]

This almost works...
“A presheaf is just a functor $\mathbb{P}^{\text{op}} \to \text{Set}$."

Easy peasy: just replace $\text{Set}$ everywhere with CIC.

$$\text{Cat} : \square := \begin{cases} 
\mathbb{P} : \square \\
\leq : \mathbb{P} \to \mathbb{P} \to \square \\
\text{id} : \Pi p. p \leq p \\
\circ : \Pi p q r. p \leq q \to q \leq r \to p \leq r \\
\text{eqn} : \ldots;
\end{cases}$$

$$\text{Psh} : \square := \begin{cases} 
\mathbb{A} : \mathbb{P} \to \square \\
\theta_\mathbb{A} : \Pi(p q : \mathbb{P})(\alpha : q \leq p). \mathbb{A}_p \to \mathbb{A}_q \\
\text{eqn} : \ldots;
\end{cases}$$

This almost works... except that equations are propositional !!!

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

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

😱 You need to introduce rewriting everywhere 😱
Equality is Too Serious a Matter

“The Coherence Hell”: the target theory must be EXTENSIONAL

\[ \Gamma \vdash e : M = N \]
\[ \Gamma \vdash M \equiv N \]
Equality is Too Serious a Matter

“The Coherence Hell”: the target theory must be **extensional**

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

- Arguably better than ZFC (“constructive”)
- ... but undecidable type-checking
- ... computation destroyed, e.g. \(\beta\)-reduction is undecidable
- See Théo Winterhalter’s soon to be defended PhD for more horrors
Equality is Too Serious a Matter

“The Coherence Hell”: the target theory must be **extensional**

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

- Arguably better than ZFC ("constructive")
- ... but undecidable type-checking
- ... computation destroyed, e.g. \( \beta \)-reduction is undecidable
- See Théo Winterhalter’s soon to be defended PhD for more horrors

**Bold Claim**

ETT is not really a type theory, so we don’t have a syntactic model.
2016

(Make conversion great again, and break everything else.)
Key Observation 1

Presheaves factorize in CBPV through a call-by-value decomposition

They only satisfy definitionally the CBV equational theory generated by

\[(\lambda x. \, t) \, V \equiv_{\beta_v} \, t\{x := V\}\]
Squaring the Circle

**Key Observation 1**

Presheaves factorize in CBPV through a *call-by-value* decomposition.

They only satisfy definitionally the CBV equational theory generated by

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

**Key Observation 2**

Type theory is *call-by-name*!

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

**Key Observation 1**

Presheaves factorize in CBPV through a *call-by-value* decomposition.

They only satisfy definitionally the CBV equational theory generated by

$$ (\lambda x. t) \ V \equiv_{\beta_v} t\{x := V\} $$

**Key Observation 2**

Type theory is *call-by-name*!

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

**Someone Had To Say It**

CBV and CBN are not the same.
If There is No Solution, There is No Problem

Easy solution! Pick the **call-by-name** decomposition instead.

\[
\begin{align*}
\text{CBV} \quad [A \rightarrow B]_p & := \Pi(q \leq p). ([A]_q \rightarrow [B]_q) \\
\text{CBN} \quad [A \rightarrow B]_p & := (\Pi(q \leq p). [A]_q) \rightarrow [B]_p
\end{align*}
\]
Easy solution! Pick the **call-by-name** decomposition instead.

- **CBV** \([A \rightarrow B]_p := \Pi(q \leq p). ([A]_q \rightarrow [B]_q)\)
- **CBN** \([A \rightarrow B]_p := (\Pi(q \leq p). [A]_q) \rightarrow [B]_p\)

- In CBN, types are not interpreted as functors in general
- Functoriality given freely by **thunking** over all lower conditions
- This adapts straightforwardly to the dependently-typed setting.
If There is No Solution, There is No Problem

Easy solution! Pick the **call-by-name** decomposition instead.

**CBV** \( [A \to B]_p := \Pi(q \leq p). ([A]_q \to [B]_q) \)

**CBN** \( [A \to B]_p := (\Pi(q \leq p). [A]_q) \to [B]_p \)

- In CBN, types are not interpreted as functors in general
- Functoriality given freely by **thunking** over all lower conditions
- This adapts straightforwardly to the dependently-typed setting.

**Theorem (Jaber & al. 2016)**

*There is a syntactic CBN presheaf model of \( CC^\omega \) into CIC.*

where \( CC^\omega \) is CIC without inductive types.
If There is No Solution, There is No Problem

Easy solution! Pick the call-by-name decomposition instead.

\[
\begin{align*}
\text{CBV} & \quad [A \to B]_p := \Pi(q \leq p). ([A]_q \to [B]_q) \\
\text{CBN} & \quad [A \to B]_p := (\Pi(q \leq p). [A]_q) \to [B]_p
\end{align*}
\]

- In CBN, types are not interpreted as functors in general
- Functoriality given freely by thunking over all lower conditions
- This adapts straightforwardly to the dependently-typed setting.

Theorem (Jaber & al. 2016)

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

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

... but the model disproves dependent elimination!

We still don’t have a syntactic presheaf model.
Interlude

Puzzle

Why does Psh(\(\mathbb{P}\)) interpret full \(\beta\)-conversion (although only extensionally)?
**Puzzle**

Why does $\text{Psh}(\mathbb{P})$ interpret full $\beta$-conversion (although *only extensionally*)?

**Answer**

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

Puzzle

Why does $\text{Psh}(P)$ interpret full $\beta$-conversion (although only extensionally)?

Answer

This is because of the naturality requirement on functions.

Theorem (Pédrot-Tabareau '20)

Naturality in CBV presheaves corresponds to Führmann’s thunkability.

- This is a well-known systematic construction from realizability
- $\text{Psh}(P)$ is the pure fragment of an effectful CBV language
- In CBV, effects break functions, in CBN they break inductive types
- We were missing the equivalent in the CBN presheaves!
Interlude

Puzzle

Why does $\text{Psh}(\mathbb{P})$ interpret full $\beta$-conversion (although only extensionally)?

Answer

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

Theorem (Pédrot-Tabareau '20)

*Naturality in CBV presheaves corresponds to Führmann’s thunkability.*

- This is a well-known **systematic** construction from realizability
- $\text{Psh}(\mathbb{P})$ is the **pure fragment** of an effectful CBV language
- In CBV, effects break functions, in CBN they break inductive types
- We were missing the equivalent in the CBN presheaves!

Theorem (Bernardy-Lasson '11)

*The CBN equivalent is parametricity. It is a syntactic model!*
On Parametric Presheaves

What does parametricity look like on the CBN presheaf model?

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

What does parametricity look like on the CBN presheaf model?

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

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)
On Parametric Presheaves

What does parametricity look like on the CBN presheaf model?

\[ x : \mathbb{B} \quad \rightarrow \quad \begin{cases} 
  x : \Pi(q \leq p). \mathbb{B} \\
  x_\epsilon : \mathbb{B}_\epsilon \ p \ x 
\end{cases} \]

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

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

2. in a unique way, i.e. \( b_1, b_2 : \mathbb{B}_\epsilon \ p \ x \vdash b_1 = b_2 \) (i.e. a HoTT proposition)

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

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

4. with further \textit{definitional} functoriality to avoid coherence issues
On Parametric Presheaves

What does parametricity look like on the CBN presheaf model?

\[ x : \mathbb{B} \rightarrow \begin{cases} \ x : \Pi(q \leq p). \mathbb{B} \\
\ x_\varepsilon : \mathbb{B}_\varepsilon \ p \ x \end{cases} \]

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

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

You cannot have both at the same time in CIC

This is exactly the CBV vs. CBN conundrum **one level higher** 😱
(On the virtues of Authoritarianism.)
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 $K$.
Essentially, we were blocked on this issue since then. When suddenly...


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

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

- A well-behaved subset of $\mathsf{Prop}$ compatible with HoTT
- It enjoys all good syntactic properties
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

$\leadsto$ $\text{SProp}$ 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 $\text{Type}$. 
Possible Extension

\[ \text{§CIC additionally allows the elimination of } \text{eq from } \text{SProp to Type} \]

This gives rise to a **strict equality**, i.e. \( \text{§CIC has definitional UIP.} \)
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.

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

... the authoritarian §CIC will instead guillotine all higher equalities.

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

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

where critically $[A]_\varepsilon \quad p \quad x : SProp$. 
In the parametric presheaf translation

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

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

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

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

We call the result the prefascist translation. (lat. fascis : sheaf)
In the parametric presheaf translation

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

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

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

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

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

**Theorem**

*The prefascist translation is a syntactic model of CIC into sCIC.*

- 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!).
$\mathcal{CIC}$ 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. $\mathcal{CIC} + \text{an impredicative universe is not } SN$
- 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.
Set is a model of $\Sigma$CIC

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

Theorem

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

$\rightsquigarrow$ they have a distinct realizability flavour compared to presheaves
Set is a model of 5CIC

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

Theorem

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

$\leadsto$ they have a distinct realizability flavour compared to presheaves

Theorem

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

Set is a model of $\aleph$CIC

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

**Theorem**

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

$\Rightarrow$ they have a distinct realizability flavour compared to presheaves

**Theorem**

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

- Proving this requires extensionality principles
- Yet, $\text{Pfs}(\mathbb{P})$ is better behaved in an intensional setting
- This could come in handy for higher category theory...
Set is a model of 5CIC

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

**Theorem**

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

$\Rightarrow$ they have a distinct realizability flavour compared to presheaves

**Theorem**

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

- Proving this requires extensionality principles
- 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
APPLICATION

Russian Constructivism

P.-M. Pédrot (INRIA)
A splinter group of constructivists, whose core tenet can be summarized as:

Proofs are Kleene realizers

Semi-classical: $HA_\omega \subseteq HA_\omega + MP \subseteq PA_\omega$

Known to preserve existence property (i.e. canonicity)

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

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

- Semi-classical: $\text{HA}^\omega \not\subseteq \text{HA}^\omega + \text{MP} \not\subseteq \text{PA}^\omega$
- Known to preserve existence property (i.e. canonicity)
- Often required to prove various completeness results
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)

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

- Semi-classical: \( \text{HA}^\omega \not\subset \text{HA}^\omega + \text{MP} \not\subset \text{PA}^\omega \)
- Known to preserve existence property (i.e. canonicity)
- Often required to prove various completeness results

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

Let's look at the realizer

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

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

Proving \( \text{mp} \vdash \text{MP} \) needs \( \text{MP} \) in the meta-theory!

As such, this is cheating.

The realizer doesn't use the doubly-negated proof.

Relies on unbounded loops in realizers.

We have little hope to implement this in CIC with a syntactic model.

We need something else...
Let’s look at the realizer

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

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

Proving \( mp \vdash \text{MP} \) needs \( \text{MP} \) in the meta-theory!

- As such, this is **cheating**
- The realizer doesn’t use the doubly-negated proof
- Relies on unbounded loops in realizers
- We have little hope to implement this in CIC with a syntactic model
Let's look at the realizer

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

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

Proving \( mp \models \text{MP} \) needs \( \text{MP} \) in the meta-theory!

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

---

We need something 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
What Else?

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

- Coquand-Hofmann’s syntactic model for \( \text{HA}_w + \text{MP} \)
- Herbelin’s direct style proof using static exceptions

**CH’s model is a mix of Kripke semantics and Friedman’s \( A \)-translation**

- Kripke semantics \( \sim \) global cell \( p : \mathbb{N} \rightarrow \mathbb{B} \) where

\[
q \leq p \quad := \quad \forall n : \mathbb{N}. \ p \ n = \text{tt} \rightarrow q \ n = \text{tt} \quad (q \ \text{truer than} \ p)
\]

- \( A \)-translation \( \sim \) exceptions of type \( A_p := \exists n : \mathbb{N}. \ p \ n = \text{tt} \)
What Else?

Not one, but at least two alternatives!

- Coquand-Hofmann’s syntactic model for $\textsf{HA}^\omega + \textsf{MP}$
- 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

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

- $A$-translation $\leadsto$ 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 $p$
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

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

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

**Theorem**

*If \( \mathcal{s}\text{CIC} \) 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

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

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

**Theorem**

*If \( s\text{CIC} \) enjoys the good properties then so does CCCP.*

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

\[
\vdash_S A : \Box \quad \rightarrow \quad \vdash_T [A]\mathcal{E} : \Box \quad + \quad \vdash_T [A]^{\bigcirc} : \mathcal{E} \rightarrow [A]\mathcal{E}
\]

In particular

\[
[[\neg A]_{\mathcal{E}} \ \cong \ [[A]_{\mathcal{E}} \rightarrow \mathcal{E}}
\]
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

\[
\begin{array}{c}
\text{CCCP} \xrightarrow{\text{Exn}} \text{CIC} + \mathcal{E} \xrightarrow{\text{Pfs}} \mathcal{S}\text{CIC}
\end{array}
\]

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 \( \text{CIC} + \mathcal{E} \)

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

\[
[\text{local } \varphi \ A]_p := [A]_{p \land \varphi}
\]

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

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

\[
\begin{align*}
\text{CCCP} & \xrightarrow{\text{Exn}} \text{CIC} + \mathcal{E} & \xrightarrow{\text{Pfs}} \mathcal{s}\text{CIC}
\end{align*}
\]

In 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 CIC + \( \mathcal{E} \)

\[
\text{local} \quad : \quad (\mathbb{N} \to \mathbb{B}) \to \Box \to \Box
\]
\[
[\text{local } \varphi \ A]_p \ \overset{\sim}{=} \ [A]_{p \wedge \varphi}
\]

- return : \( A \to \text{local } \varphi \ A \)
- local commutes to arrows and positive types
- local \( \varphi \ \mathcal{E} \quad \overset{\sim}{=} \quad \mathcal{E} + (\Sigma n : \mathbb{N}. \varphi \ n = \text{tt}) \)

**Theorem**

CCCP *validates* MP.

Proof by symbol pushing in CIC + \( \mathcal{E} \) by the above and \( [\neg A]_{\mathcal{E}} \overset{\sim}{=} [A]_{\mathcal{E}} \to \mathcal{E} \).
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!

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

return is a delimited continuation prompt / static exception binder.

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

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

In particular \( p \) can raise exceptions from outside, which is reflected here.
Every time we go under local we get new exceptions!

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

return is a delimited continuation prompt / static exception binder.

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

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

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

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

On presheaves:

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

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

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

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

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

TODO:
- Implement cubical type theory in this model
Thanks for your attention.