Forcing Translations in Type Theory

M. Sozeau, Inria Paris & IRIF

jww. G. Jaber, G. Lewertowski, P.-M. Pédrot & N. Tabareau

Categorical Logic & Univalent Foundations
Leeds, UK
July 28th 2016

Forcing in a Nutshell

- Historically, forcing is a model transformation
- Several names for the same concept

Forcing translation	$\cong$	**Kripke** models	$\cong$	**Presheaf** construction
(Set theory)		*(Modal logic)*		*(Category theory)*

- Usually, set-theoretic forcing is classical
- We will study intuitionistic forcing, in intuitionistic type theory

Why use forcing?

Why use forcing?

- Set theory: a lot of independence results continuum hypothesis, AC, ...
- Modal logic and Kripke Models

Why use forcing?

- Set theory: a lot of independence results continuum hypothesis, AC, . . .
- Modal logic and Kripke Models
- Category theory: a HoTT topic!
  - Many models arise from presheaf constructions
  - Coquand & al.'s cubical model of univalence is an example
  - Also step-indexing, parametricity...
  - But this targets sets or topoi usually

We want forcing in Type Theory!

Intuitionistic Forcing in **LJ** (Kripke, presheaf)

Assume a preorder $(\mathbb{P}, \leq)$. We summarize the forcing translation in **LJ**.

- To a formula A, we associate a $\mathbb{P}$-indexed formula $[\![A]\!]_p$.
- To a proof $\vdash A$, we associate a proof of $\forall p : \mathbb{P}, [\![A]\!]_p$.
- (Target theory not really specified here, think $\lambda\Pi$.)

$\ll$ $\mathbb{P}$ are possible worlds, $[\![A]\!]_p$ is truth at world p $\gg$

Intuitionistic Forcing in **LJ** (Kripke, presheaf)

Assume a preorder $(\mathbb{P}, \leq)$. We summarize the forcing translation in **LJ**.

- To a formula A, we associate a $\mathbb{P}$-indexed formula $[\![A]\!]_p$.
- To a proof $\vdash A$, we associate a proof of $\forall p : \mathbb{P}, [\![A]\!]_p$.
- (Target theory not really specified here, think $\lambda\Pi$.)

$\ll$ $\mathbb{P}$ are possible worlds, $[\![A]\!]_p$ is truth at world p $\gg$

Most notably,

$$[\![A \to B]\!]_p := \forall q \leq p. \, [\![A]\!]_q \to [\![B]\!]_q$$

Intuitionistic Forcing in **LJ** (Kripke, presheaf)

Assume a preorder $(\mathbb{P}, \leq)$. We summarize the forcing translation in **LJ**.

- To a formula A, we associate a $\mathbb{P}$-indexed formula $[\![A]\!]_p$.
- To a proof $\vdash A$, we associate a proof of $\forall p : \mathbb{P}, [\![A]\!]_p$.
- (Target theory not really specified here, think $\lambda\Pi$.)

$\ll$ $\mathbb{P}$ are possible worlds, $[\![A]\!]_p$ is truth at world p $\gg$

Most notably,

$$[\![A \to B]\!]_p := \forall q \leq p. [\![A]\!]_q \to [\![B]\!]_q$$

Actually this can be adapted straightforwardly to any category $(\mathbb{P}, \mathtt{Hom})$.

Through the Curry-Howard Lens

The previous soundness theorem also makes sense in a *proof-relevant* world:

If $\vdash t : A$ then $p : \mathbb{P} \vdash [t]_p : [\![A]\!]_p$

Through the Curry-Howard Lens

The previous soundness theorem also makes sense in a *proof-relevant* world:

$$\text{If } \vdash t : A \text{ then } p : \mathbb{P} \vdash [t]_p : [\![A]\!]_p$$

... and the translation can be thought of as a monotonous monad reader

Reader	Forcing
$T\ A := \mathbb{P} \to A$	$T_p\ A := \forall q : \mathbb{P}, q \leq p \to A$
$\mathtt{read} : 1 \to \mathbb{P}$	$\mathtt{read} : 1 \to \mathbb{P}$
$\mathtt{enter} : (1 \to A) \to \mathbb{P} \to A$	$\mathtt{enter} : (1 \to A) \to \forall p : \mathbb{P}, p \leq \mathtt{read}() \to A$

Through the Curry-Howard Lens

The previous soundness theorem also makes sense in a *proof-relevant* world:

$$\text{If } \vdash t : A \text{ then } p : \mathbb{P} \vdash [t]_p : [\![A]\!]_p$$

... and the translation can be thought of as a monotonous monad reader

Reader	Forcing
$T\ A := \mathbb{P} \to A$	$T_p\ A := \forall q : \mathbb{P}, q \leq p \to A$
$\mathtt{read} : 1 \to \mathbb{P}$	$\mathtt{read} : 1 \to \mathbb{P}$
$\mathtt{enter} : (1 \to A) \to \mathbb{P} \to A$	$\mathtt{enter} : (1 \to A) \to \forall p : \mathbb{P}, p \leq \mathtt{read}() \to A$

In particular, taking $(\mathbb{P}, \leq)$ to be a full preorder gives the reader monad.

Idea of the proof and use

- Substitution lemma for the interpretation.
- "Computational soundness": $t \to_\beta u \Rightarrow [t] \equiv_\beta [u]$

Idea of the proof and use

- Substitution lemma for the interpretation.
- "Computational soundness": $t \to_\beta u \Rightarrow [t] \equiv_\beta [u]$

One can add "generic" elements in the forcing layer by inhabiting their translations:

$$[\vdash_{\mathbb{F}} a : \psi] \triangleq \; a^\bullet : \forall p : \mathbb{P}, [\![\psi]\!]_p$$

Thanks to soundness of the translation, and (assumed) consistency of the source system, as soon as $\mathbb{P}$ is inhabited:

$$\vdash_{\mathbb{F}} t : \bot \Rightarrow p : \mathbb{P} \vdash [t]_p : [\![\bot]\!]_p \equiv \Pi\; q \leq p.\bot$$

We have equiconsistency.

Do it, or do not: there is no try

In 2012, we gave a forcing translation from $\mathrm{CC}_\omega + \Sigma$ into itself.

Do it, or do not: there is no try

In 2012, we gave a forcing translation from $\mathsf{CC}_\omega + \Sigma$ into itself.

Intuitively, not that difficult.

- To a type $\vdash A : \Box$ associate $p : \mathbb{P} \vdash [\![A]\!]_p : \Box$.
- To a term $\vdash t : A$ associate $p : \mathbb{P} \vdash [t]_p : [\![A]\!]_p$ by induction on t

Do it, or do not: there is no try

In 2012, we gave a forcing translation from $\mathsf{CC}_\omega + \Sigma$ into itself.

Intuitively, not that difficult.

- To a type $\vdash A : \square$ associate $p : \mathbb{P} \vdash [\![A]\!]_p : \square$.
- To a term $\vdash t : A$ associate $p : \mathbb{P} \vdash [t]_p : [\![A]\!]_p$ by induction on t
- To handle types-as-terms uniformly, $[\![\cdot]\!]$ is defined through $[\cdot]$

$$\begin{array}{lcl} [A]_p & : & (\Pi q \leq p \to \square). \quad (A \text{ type}) \\ [\![A]\!]_p & := & [A]_p\ p\ \mathtt{id}_p \end{array}$$

- Translation of the dependent arrow is almost the same:

$$[\![\Pi x : A.\, B]\!]_p \equiv \Pi q \leq p.\, \Pi x : [\![A]\!]_q.\, [\![B]\!]_q$$

Do it, or do not: there is no try

In 2012, we gave a forcing translation from $\mathrm{CC}_\omega + \Sigma$ into itself.

Intuitively, not that difficult.

- To a type $\vdash A : \square$ associate $p : \mathbb{P} \vdash [\![A]\!]_p : \square$.
- To a term $\vdash t : A$ associate $p : \mathbb{P} \vdash [t]_p : [\![A]\!]_p$ by induction on t
- To handle types-as-terms uniformly, $[\![\cdot]\!]$ is defined through $[\cdot]$

$$\begin{array}{lcl} [A]_p & : & (\Pi q \leq p \to \square). \quad (A \text{ type}) \\ [\![A]\!]_p & := & [A]_p\ p\ \mathtt{id}_p \end{array}$$

- Translation of the dependent arrow is almost the same:

$$[\![\Pi x : A.\, B]\!]_p \equiv \Pi q \leq p.\, \Pi x : [\![A]\!]_q.\, [\![B]\!]_q$$

... except that we must add restrictions!

To presheaves

We move to:

$$
\begin{array}{lcll}
[A]_p & : & \Sigma f : (\Pi q \leq p \rightarrow \square). & (A \text{ type}) \\
 & & \{\theta : \Pi q \leq p. \Pi r \leq q. f\ q \rightarrow f\ r \mid & (\Theta \text{ restriction}) \\
 & & \mathsf{refl}(\theta, p) \wedge \mathsf{trans}(\theta, p)\} & (\Theta \text{ functorial}) \\
[\![A]\!]_p & := & (\pi_1\ [A]_p)\ p\ \mathtt{id}_p &
\end{array}
$$

To presheaves

We move to:

$$
\begin{array}{lcll}
[A]_p & : & \Sigma f : (\Pi q \leq p \to \square). & (A \text{ type}) \\
 & & \{\theta : \Pi q \leq p . \Pi r \leq q . f\ q \to f\ r \mid & (\Theta \text{ restriction}) \\
 & & \mathsf{refl}(\theta, p) \wedge \mathsf{trans}(\theta, p)\} & (\Theta \text{ functorial}) \\
[\![A]\!]_p & := & (\pi_1\ [A]_p)\ p\ \mathtt{id}_p &
\end{array}
$$

In general, under a context σ of variables + forcing conditions:

$$[x]^{\sigma}_{p} \quad \stackrel{def}{=} \quad \theta^{\sigma, \sigma_1(x)}_{\sigma_2(x) \to p} x$$

To presheaves

We move to:

$$
\begin{array}{lcll}
[A]_p & : & \Sigma f : (\Pi q \leq p \to \Box). & (A \text{ type}) \\
& & \{\theta : \Pi q \leq p.\Pi r \leq q.f\ q \to f\ r \mid & (\Theta \text{ restriction}) \\
& & \mathsf{refl}(\theta, p) \wedge \mathsf{trans}(\theta, p)\} & (\Theta \text{ functorial}) \\
[\![A]\!]_p & := & (\pi_1\ [A]_p)\ p\ \mathtt{id}_p &
\end{array}
$$

In general, under a context σ of variables + forcing conditions:

$$
[x]_p^\sigma \quad \stackrel{def}{=} \quad \theta^{\sigma, \sigma_1(x)}_{\sigma_2(x) \to p} x
$$

Now we have witnesses everywhere

To presheaves

We move to:

$$
\begin{array}{lcll}
[A]_p & : & \Sigma f : (\Pi q \leq p \to \Box). & (A \text{ type}) \\
 & & \{\theta : \Pi q \leq p. \Pi r \leq q. f\ q \to f\ r \mid & (\Theta \text{ restriction}) \\
 & & \mathsf{refl}(\theta, p) \wedge \mathsf{trans}(\theta, p)\} & (\Theta \text{ functorial}) \\
[\![A]\!]_p & := & (\pi_1\ [A]_p)\ p\ \mathtt{id}_p &
\end{array}
$$

In general, under a context σ of variables + forcing conditions:

$$[x]_p^\sigma \quad \stackrel{def}{=} \quad \theta^{\sigma, \sigma_1(x)}_{\sigma_2(x) \to p} x$$

Now we have witnesses everywhere

... but it's no longer computationally sound!

Some proofs are more equal than others

The culprit is the conversion rule:

$$\frac{\vdash t : A \qquad A \equiv_\beta B}{\vdash t : B} \quad \rightsquigarrow \quad \frac{p : \mathbb{P} \vdash [t]_p : [\![A]\!]_p \qquad [\![A]\!]_p \equiv_\beta [\![B]\!]_p}{p : \mathbb{P} \vdash [t]_p : [\![B]\!]_p}$$

In general, $A \equiv_\beta B$ does not imply $[\![A]\!]_p \equiv_\beta [\![B]\!]_p$, as restrictions do not commute/compose "on the nose".

Some proofs are more equal than others

The culprit is the conversion rule:

$$\frac{\vdash t : A \qquad A \equiv_\beta B}{\vdash t : B} \quad \rightsquigarrow \quad \frac{p : \mathbb{P} \vdash [t]_p : [\![A]\!]_p \qquad [\![A]\!]_p \equiv_\beta [\![B]\!]_p}{p : \mathbb{P} \vdash [t]_p : [\![B]\!]_p}$$

In general, $A \equiv_\beta B$ does not imply $[\![A]\!]_p \equiv_\beta [\![B]\!]_p$, as restrictions do not commute/compose "on the nose".

$$[\![\Pi x : T.U]\!]_p^\sigma \stackrel{def}{=} \{f : \Pi q : \mathcal{P}_p \Pi x : [\![T]\!]_q^\sigma . [\![U]\!]_q^{\sigma+(x,T,q)} \mid \mathbf{comm}_\Pi(f, T, U, p)\}$$

$$\begin{tikzcd}
{[\![T]\!]_p^\sigma} \arrow[r, "f_p"] \arrow[d, "\theta_{p\to q}^{\sigma,T}"'] & {[\![U]\!]_p^\sigma} \arrow[d, "\theta_{p\to q}^{\sigma,U}"] \\
{[\![T]\!]_q^\sigma} \arrow[r, "f_q"'] & {[\![U]\!]_q^\sigma}
\end{tikzcd}$$

When conversion matters

We only recover that $A \equiv_\beta B$ implies $p : \mathbb{P} \vdash [\![A]\!]_p =_\Box [\![B]\!]_p$.
In the end, you cannot interpret conversion by mere conversion.

$$\frac{\vdash t : A \qquad A \equiv_\beta B}{\vdash t : B} \quad \rightsquigarrow \quad \frac{p : \mathbb{P} \vdash [t]_p : [\![A]\!]_p \qquad \pi : [\![A]\!]_p = [\![B]\!]_p}{p : \mathbb{P} \vdash \texttt{transport}([\pi], [t]_p) : [\![B]\!]_p}$$

The ≪ diagram ≫ does not commute in ITT

When conversion matters

We only recover that $A \equiv_\beta B$ implies $p : \mathbb{P} \vdash [\![A]\!]_p =_\Box [\![B]\!]_p$.
In the end, you cannot interpret conversion by mere conversion.

$$\frac{\vdash t : A \qquad A \equiv_\beta B}{\vdash t : B} \quad \rightsquigarrow \quad \frac{p : \mathbb{P} \vdash [t]_p : [\![A]\!]_p \qquad \pi : [\![A]\!]_p \color{red}{=} [\![B]\!]_p}{p : \mathbb{P} \vdash \color{red}{\mathtt{transport}([\pi],}[t]_p\color{red}{)} : [\![B]\!]_p}$$

The « diagram » does not commute in ITT

It raises a hell of coherence issues.

- Breaks computation
- Requires *definitional* UIP in the target (i.e. OTT or ETT)
- Requires that $\leq$ is proof-irrelevant.
- Only preorder-based presheaf models!

Nonetheless

In a modified Coq with definitional proof-irrelevance (for `Prop`):

- We could adapt the proof of consistency of the negation of the continuum hypothesis.
- We could internalize step indexing as a forcing layer (i.e. to obtain a general fixpoint in type theory).

Step-indexing as a forcing layer

Take $\mathbb{P} \triangleq \mathbb{N}$ with the standard order relation.

- Define $\triangleright_{\square}$: $\square \to \square$ the "later" modality on $\square$ in the forcing layer.
  By translation we must provide a witness of $\Pi q \leq p.\Pi T : [\![\square]\!]_q, [\![\square]\!]_q$, which computes to the unit type when $q = 0$ and the nth-approximation of T at $n + 1$.

Step-indexing as a forcing layer

Take $\mathbb{P} \triangleq \mathbb{N}$ with the standard order relation.

- Define $\triangleright_\Box : \Box \to \Box$ the "later" modality on $\Box$ in the forcing layer.
  By translation we must provide a witness of $\Pi q \leq p.\Pi T : [\![\Box]\!]_q, [\![\Box]\!]_q$, which computes to the unit type when $q = 0$ and the nth-approximation of T at $n+1$.
- Define $\texttt{fix}_T : (\triangleright_\Box T \to T) \to T$ (the Löb rule) by providing a witness using the "step-index".
- Define the lifting $\texttt{next}_T : (T \to \triangleright_\Box T)$, morally "delay".

Step-indexing as a forcing layer

Take $\mathbb{P} \triangleq \mathbb{N}$ with the standard order relation.

- Define $\triangleright_\square \colon \square \to \square$ the "later" modality on $\square$ in the forcing layer.
  By translation we must provide a witness of $\Pi q \leq p.\Pi T : [\![\square]\!]_q, [\![\square]\!]_q$, which computes to the unit type when $q = 0$ and the nth-approximation of T at $n + 1$.
- Define $\texttt{fix}_T : (\triangleright_\square T \to T) \to T$ (the Löb rule) by providing a witness using the "step-index".
- Define the lifting $\texttt{next}_T : (T \to \triangleright_\square T)$, morally "delay".

In the forcing layer, it becomes possible to reason with general fixpoints on types having the unfolding lemma:

$$\texttt{fix}_\square\ f = f\ (\texttt{next}\ (\texttt{fix}_\square\ f))$$

.

Issues

The setup is not very satisfactory though:

- Doubts about coherence of the whole translation.
- Tedious proofs involving rewriting appear when reasoning with these fixpoints.

A new hope

Interestingly the Curry-Howard isomorphism explains the difficulties with this translation.

Root of the failure

The usual forcing $[\cdot]_p$ translation is **call-by-value**.

That is, assuming $(\mathbb{P}, \leq)$ has definitional laws:

$$t \equiv_{\beta v} u \qquad \text{implies} \qquad [t]_p \equiv_\beta [u]_p$$

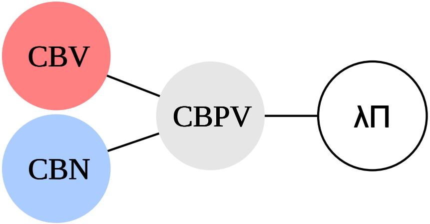
where βv is generated by the rule:

$$(\lambda x.\, t)\, V \longrightarrow_{\beta v} t\{x := V\} \qquad (V \text{ a value})$$

This problem is already here in the simply-typed case but less troublesome.

The Two Sides of Forcing

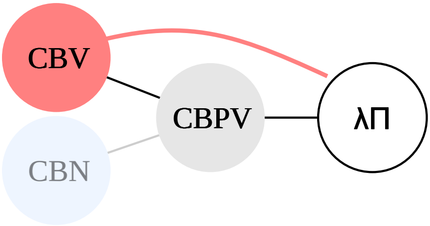
There is an easy Call-by-Push-Value decomposition of forcing.



The Two Sides of Forcing

There is an easy Call-by-Push-Value decomposition of forcing.

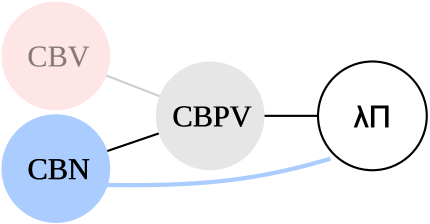
- Precomposing by the CBV decomposition we recover the usual forcing



The Two Sides of Forcing

There is an easy Call-by-Push-Value decomposition of forcing.

- Precomposing by the CBV decomposition we recover the usual forcing
- Precomposing by the CBN decomposition we obtain a new translation
- ... much closer to Krivine and Miquel's classical variant



CBN provides new abilities

You only have to change the interpretation of the arrow.

$$\begin{array}{ll} \text{CBV} & [\![\Pi x : A.\, B]\!]_p \cong \Pi q \leq p.\, \Pi x : [\![A]\!]_q.\, [\![B]\!]_q \\ \text{CBN} & [\![\Pi x : A.\, B]\!]_p \equiv \Pi(x : \Pi q \leq p.\, [\![A]\!]_q).\, [\![B]\!]_p \end{array}$$

CBN provides new abilities

You only have to change the interpretation of the arrow.

CBV	$[\![\Pi x : A.\, B]\!]_p \cong \Pi q \leq p.\, \Pi x : [\![A]\!]_q.\, [\![B]\!]_q$
CBN	$[\![\Pi x : A.\, B]\!]_p \equiv \Pi(x : \Pi q \leq p.\, [\![A]\!]_q).\, [\![B]\!]_p$
CBN	$[\![x]\!]_p \equiv x\ p\ \mathtt{id}_p$

CBN provides new abilities

You only have to change the interpretation of the arrow.

CBV	$[\![\Pi x : A.\, B]\!]_p \cong \Pi q \leq p.\, \Pi x : [\![A]\!]_q.\, [\![B]\!]_q$
CBN	$[\![\Pi x : A.\, B]\!]_p \equiv \Pi(x : \Pi q \leq p.\, [\![A]\!]_q).\, [\![B]\!]_p$
CBN	$[\![x]\!]_p \equiv x\ p\ \mathtt{id}_p$

... and everything follows naturally (CBN is somehow a ≪ free ≫ construction).

Interpretation of $\mathbf{CC}_\omega$

Assuming that $\mathbb{P}$ has definitional laws (for identity and composition), then $[\![\cdot]\!]$ provides a non-trivial translation from $\mathbf{CC}_\omega$ into itself preserving typing and conversion.

This is to the best of our knowledge, the first effectful translation of $\mathbf{CC}_\omega$.

The translation

$$
\begin{array}{lcl}
[*]_\sigma & := & \lambda(q\, f : \sigma).\, \Pi(r\, g : \sigma \cdot (q, f)).\, * \\
[\square_i]_\sigma & := & \lambda(q\, f : \sigma).\, \Pi(r\, g : \sigma \cdot (q, f)).\, \square_i \\
[x]_\sigma & := & x\ \sigma_e\ \sigma(x) \\
[\lambda x : A.\, M]_\sigma & := & \lambda x : [\![A]\!]^!_\sigma.\, [M]_{\sigma \cdot x} \\
[M\ N]_\sigma & := & [M]_\sigma\ [N]^!_\sigma \\
[\Pi x : A.\, B]_\sigma & := & \lambda(q\, f : \sigma).\, \Pi x : [\![A]\!]^!_{\sigma\cdot(q,f)}.\, [\![B]\!]_{\sigma\cdot(q,f)\cdot x} \\
[\![A]\!]_\sigma & := & [A]_\sigma\ \sigma_e\ \mathtt{id}_{\sigma_e} \\
[M]^!_\sigma & := & \lambda(q\, f : \sigma).\, [M]_{\sigma\cdot(q,f)} \\
[\![A]\!]^!_\sigma & := & \Pi(q\, f : \sigma).\, [\![A]\!]_{\sigma\cdot(q,f)}
\end{array}
$$

$$
\begin{array}{lcl}
[\![\cdot]\!]_p & := & p : \mathbb{P} \\
[\![\Gamma]\!]_{\sigma\cdot(q,f)} & := & [\![\Gamma]\!]_\sigma, q : \mathbb{P}, f : \mathtt{Hom}(\sigma_e, q) \\
[\![\Gamma, x : A]\!]_{\sigma\cdot x} & := & [\![\Gamma]\!]_\sigma, x : [\![A]\!]^!_\sigma
\end{array}
$$

Is the definitional side stronger?

This variant is motivated by a Curry-Howard stance.

- No categorical equivalent from the literature (?).
- Definitely not a presheaf construction!
- In particular, no monotonicity / restrictions
- Only known relative comes from Krivine and Miquel (also CH)
- Yet, still the same object in the simply-typed case.
- Can be used for NBE as well

What is this beast?

Yoneda to the rescue

Technical issue: how can $\mathbb{P}$ have definitional laws?

Yoneda to the rescue

Technical issue: how can $\mathbb{P}$ have definitional laws?

Answer: using this one weird old Yoneda trick!

$$(\mathbb{P}, \leq) \qquad \mapsto \qquad (\mathbb{P}_{\mathcal{Y}}, \leq_{\mathcal{Y}})$$

$$\begin{array}{lcl} \mathbb{P}_{\mathcal{Y}} & := & \mathbb{P} \\ p \leq_{\mathcal{Y}} q & := & \Pi r : \mathbb{P}.\, q \leq r \to p \leq r \end{array}$$

Yoneda lemma

- The category $(\mathbb{P}_{\mathcal{Y}}, \leq_{\mathcal{Y}})$ is equivalent to $(\mathbb{P}, \leq)$ (assuming parametricity and functional extensionality).
- Furthermore, it has definitional laws as associativity of functions is on the nose in ITT.

Inductive types

Up to now, we only interpret the negative fragment $(\Pi + \Box)$.

Inductive types

Up to now, we only interpret the negative fragment ($\Pi + \square$).

Adapting to (positive) inductive types.
We just need to **box** all subterms!

$$[\![\Sigma x : A.\, B]\!]_p \quad := \quad \Sigma(x : \Pi q \leq p.\, [\![A]\!]_q).\, (\Pi q \leq p.\, [\![B]\!]_q)$$

$$[\![A + B]\!]_p \quad := \quad (\Pi q \leq p.\, [\![A]\!]_q) + (\Pi q \leq p.\, [\![B]\!]_q)$$

$$\texttt{Inductive}\ [\![\mathbb{N}]\!]_p : \square := [\texttt{O}] : [\![\mathbb{N}]\!]_p \mid [\texttt{S}] : (\Pi q \leq p.\, [\![\mathbb{N}]\!]_q) \to [\![\mathbb{N}]\!]_p$$

Dependent elimination

Yet, the translation does not interpret full dependent elimination.

$\mathbb{N}_{\texttt{rec}}$ $\quad \Pi(P : \square).\, P \to (P \to P) \to \mathbb{N} \to P$ ✓

$\mathbb{N}_{\texttt{ind}}$ $\quad \Pi(P : \mathbb{N} \to \square).\, P\ \texttt{O} \to (\Pi n : \mathbb{N}.\, P\ n \to P\ (\texttt{S}\ n)) \to \Pi n : \mathbb{N}.\, P\ n$ ✠

Dependent elimination

Yet, the translation does not interpret full dependent elimination.

$\mathbb{N}_{\mathtt{rec}}$ $\quad \Pi(P : \square).\, P \to (P \to P) \to \mathbb{N} \to P$ ✓

$\mathbb{N}_{\mathtt{ind}}$ $\quad \Pi(P : \mathbb{N} \to \square).\, P\ \mathtt{0} \to (\Pi n : \mathbb{N}.\, P\ n \to P\ (\mathtt{S}\ n)) \to \Pi n : \mathbb{N}.\, P\ n$ ✠

Effects $\rightsquigarrow$ Non-standard inductive terms

(A well-known issue. See e.g. Herbelin's $\mathbf{CIC} + \mathtt{callcc}$)

Dependent elimination

Yet, the translation does not interpret full dependent elimination.

$\mathbb{N}_{\texttt{rec}}$	$\Pi(P : \square).\, P \to (P \to P) \to \mathbb{N} \to P$	✓
$\mathbb{N}_{\texttt{ind}}$	$\Pi(P : \mathbb{N} \to \square).\, P\ \texttt{0} \to (\Pi n : \mathbb{N}.\, P\ n \to P\ (\texttt{S}\ n)) \to \Pi n : \mathbb{N}.\, P\ n$	✠

Effects $\rightsquigarrow$ Non-standard inductive terms

(A well-known issue. See e.g. Herbelin's $\mathbf{CIC} + \texttt{callcc}$)

Luckily there is a surprise solution coming from classical realizability.

Storage operators!

Storage operators

- They allow to prove induction principles in presence of `callcc`
- Essentially emulate CBV in CBN through a CPS
- Defined in terms of non-dependent recursion

$$
\begin{array}{lcl}
\theta_{\mathbb{N}} & : & \mathbb{N} \to \Pi R : \square.\, (\mathbb{N} \to R) \to R \\
\theta_{\mathbb{N}} & := & \mathbb{N}_{\mathtt{rec}}\ (\lambda R\ k.\, k\ 0)(\lambda \tilde{n}\ R\ k.\, \tilde{n}\ R\ (\lambda n.\, k\ (S\ n)))
\end{array}
$$

Storage operators

- They allow to prove induction principles in presence of `callcc`
- Essentially emulate CBV in CBN through a CPS
- Defined in terms of non-dependent recursion

$$\begin{array}{lcl} \theta_{\mathbb{N}} & : & \mathbb{N} \to \Pi R : \square.\, (\mathbb{N} \to R) \to R \\ \theta_{\mathbb{N}} & := & \mathbb{N}_{\mathtt{rec}}\ (\lambda R\ k.\, k\ 0)(\lambda \tilde{n}\ R\ k.\, \tilde{n}\ R\ (\lambda n.\, k\ (S\ n))) \end{array}$$

- Trivial in $\mathbf{CIC}$: $\mathbf{CIC} \vdash \Pi n\ R\ k. \quad \theta_{\mathbb{N}}\ n\ R\ k =_R k\ n$
- The above propositional η-rule is negated by the forcing translation
- But it interprets a restricted dependent elimination!

Storage operators

- They allow to prove induction principles in presence of `callcc`
- Essentially emulate CBV in CBN through a CPS
- Defined in terms of non-dependent recursion

$$\begin{array}{lcl} \theta_{\mathbb{N}} & : & \mathbb{N} \to \Pi R : \square .\, (\mathbb{N} \to R) \to R \\ \theta_{\mathbb{N}} & := & \mathbb{N}_{\mathtt{rec}}\ (\lambda R\ k.\, k\ 0)(\lambda \tilde{n}\ R\ k.\, \tilde{n}\ R\ (\lambda n.\, k\ (S\ n))) \end{array}$$

- Trivial in $\mathbf{CIC}$: $\mathbf{CIC} \vdash \Pi n\ R\ k.\quad \theta_{\mathbb{N}}\ n\ R\ k =_R k\ n$
- The above propositional η-rule is negated by the forcing translation
- But it interprets a restricted dependent elimination!

$\mathbb{N}_{\widetilde{\mathtt{ind}}}$ $\quad \Pi P.\, P\ \mathtt{0} \to (\Pi n : \mathbb{N}.\, P\ n \to \theta_{\mathbb{N}}\ (\mathtt{S}\ n)\ \square\ P) \to \Pi n : \mathbb{N}.\, \theta_{\mathbb{N}}\ n\ \square\ P$ ✓

Implementation & examples

- A plugin for Coq generating translated terms

A truly definitional translation!

Implementation & examples

- A plugin for Coq generating translated terms

A truly definitional translation!

- A handful of independence results and usecases

$\rightsquigarrow$ Preserves UIP and functional extensionality
$\rightsquigarrow$ Generate anomalous types that negate univalence
$\rightsquigarrow$ Preserves (a simple version of) univalence for modal types
$\rightsquigarrow$ Step indexing (FRP, ≪ fuel trick ≫)
$\rightsquigarrow$ Give some intuition for the cubical model

Implementation & examples

- A plugin for Coq generating translated terms

A truly definitional translation!

- A handful of independence results and usecases

$\rightsquigarrow$ Preserves UIP and functional extensionality
$\rightsquigarrow$ Generate anomalous types that negate univalence
$\rightsquigarrow$ Preserves (a simple version of) univalence for modal types
$\rightsquigarrow$ Step indexing (FRP, « fuel trick »)
$\rightsquigarrow$ Give some intuition for the cubical model

- Demo

What remains to be done

- Recover a propositional η-rule by using parametricity
- Understanding the cubical model in CBN.
- Design a general theory of **CIC** $+$ effects using storage operators
- The next 700 translations of **CIC** into itself, degenerate translations. E.g. breaking parametricity with built-in `quote` operators.

Related Work & References

- The Independence of Markov's Principle in Type Theory. T. Coquand, B. Mannaa, FSCD 2016
- Forcing as a Program Transformation, A. Miquel, LICS 2011.
- The Definitional Side of Forcing - G. Jaber, G. Lewertowski, P.-M. Pédrot, M. Sozeau, N. Tabareau, LICS'16
- Forcing in Type Theory - G. Jaber, M. Sozeau & N. Tabareau, LICS'12

The End

`https://github.com/CoqHott/coq-forcing`