

Russian Constructivism in a Prefascist Theory

Pierre-Marie Pédrot

▶ **To cite this version:**

Pierre-Marie Pédrot. Russian Constructivism in a Prefascist Theory. LICS 2020 - Thirty-Fifth Annual ACM/IEEE Symposium on Logic in Computer Science, Jul 2020, Saarbrücken, Germany. pp.1-14, 10.1145/3373718.3394740 . hal-02548315

HAL Id: hal-02548315
https://inria.hal.science/hal-02548315

Submitted on 20 Apr 2020

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Russian Constructivism in a Prefascist Theory

Pierre-Marie Pédrot
INRIA
France
pierre-marie.pedrot@inria.fr

Abstract

The results from this paper are twofold. First, we give a purely syntactic presheaf model of CIC. Contrarily to similar endeavours, this variant both preserves conversion and interprets full dependent elimination.

Using a particular instance of this model, we show how to extend CIC with Markov's principle, while preserving all good meta-theoretical properties like canonicity and decidability of type-checking. The resulting construction can be seen as a synthetic presentation of Coquand-Hofmann's syntactic model of PRA^{ω} + MP as the composition of Pédrot-Tabareau's exceptional model with our presheaf interpretation.

CCS Concepts: • **Theory of computation → Type theory**; *Control primitives*; *Constructive mathematics.*

Keywords: syntactic model, presheaf, dependent types, strict propositions, Markov's principle, exceptions

ACM Reference Format:
Pierre-Marie Pédrot. 2020. Russian Constructivism in a Prefascist Theory. In *Proceedings of the 35th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS '20), July 8–11, 2020, Saarbrücken, Germany.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3373718.3394740

1 Introduction

The *category of presheaves* over some base category is a bread-and-butter model factory from topos theory. Well-known in the proof theory community as *forcing* or *Kripke semantics*, presheaves also provide models of dependent type theory. Yet, as functors into **Set**, presheaves form a semantic model, which has various drawbacks. The first one is that they are not really computable, even though the target set theory is constructive. Indeed, the constructive existence of a natural number is always up to some extensional equality, which prevents set-theoretical proofs to be effectively run. The second one is that it is hard to control the precise metatheory used to prove the soundness of the model. What part of set theory is really needed for presheaves to model type theory? Third, a more philosophical issue arises. How can we claim that type theory can provide a sane alternative foundation to mathematics, like the HoTT clique does [45], when it lays on the muddy grounds of set theory?

In order to solve this conundrum, there have been several attempts at giving a syntax-conscious presentation of presheaves. One possible way is to describe an extension of usual type theory, justify its consistency into a set-theoretical presheaf model, but show at the same time good properties of the syntax, like strong normalization. This lukewarm compromise position was held for instance by cubical [13], guarded [11], parametric [9] and modal [24] type theories.

The vanguard approach consists instead in replacing set theory with type theory altogether in the presheaf translation, building what is called a *syntactic model* [27]. While there exist indeed reasonable instances of syntactic models [27, 40], presheaves have proven surprisingly subtle to be typified, despite a handful of unsatisfactory partial results [31, 32, 46]. The source of insatisfaction itself is quite hard to pin down for non-experts in type theory, and will be described in detail in Section 2.

In this paper, we give a strict positive answer to the existence of type-theoretic presheaves given by a syntactic translation described at Section 4. By construction, this model ensures all the expected syntactic properties of the corresponding theory such as decidability of type-checking and canonicity. There are two critical ingredients to this recipe. First, a deep understanding of the constraints to respect, driven by an intuition from programming language theory, and the half-successes from the related work, which is also discussed in Section 2. Second, a relatively minor extension of type theory that introduces a tiny bit of extensionality without endangering the meta-theoretical properties of the resulting system, originally introduced in Gilbert et al. [23] and that will be recalled in Section 3.

Because presheaves *per se* are not that interesting, this paper culminates with an example of a mathematically useful extension of type theory. We show how to extend type theory with Markov's principle (MP), a staple semi-classical principle from the Russian school of constructivism, discussed in Section 8. This theory,

LICS '20, July 8–11, 2020, Saarbrücken, Germany
2020. This is the author's version of the work. It is posted here for your personal use. Not for redistribution. The definitive Version of Record was published in *Proceedings of the 35th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS '20), July 8–11, 2020, Saarbrücken, Germany*, https://doi.org/10.1145/3373718.3394740.

once again, enjoys the usual meta-theoretical properties, and furthermore its soundness does not rely on the availability of MP in the meta-theory. It is given not by one, but two syntactic models. To be precise, it uses the combination of the exceptional model [40], which we will recall in Section 9 and use in Section 10, with an instance of the presheaf model introduced before, that we describe in Section 11. From an abstract standpoint, it is both a synthetic presentation and a dependently-typed generalization of the PRA^{ω} + MP model from Coquand-Hofmann [15]. As far as we know, this is a new result.

2 What is to be Done?

In this section, we review the previous attempts at a internal presentation of presheaves in type theory, and explain why they are subtly broken and how to fix them.

Jaber et al. gave a straightforward type-theoretic adaptation of presheaves [32], by merely translating all the set-theoretic constructs. Unfortunately, they realized that for this model to be sound, they needed what amounted to an extensional target theory. In particular, they interpreted conversion in the presheaf model by propositional equality in the target, relying on axioms such as function extensionality and UIP. In practice, it means that the computational content of the model is destroyed in the process, as well as decidability. Performing a β-reduction in a term becomes a non-local transformation that requires an explicit proof. To add insult to injury, this also makes canonicity pointless, as it is not possible to actually evaluate a term, as the axioms get in the way.

In an unpublished document, Xu follows the same path [46], and gives a precise analysis of the extensional principles needed for each feature of this model.

Lemma 1. *In a type-theoretic presheaf model:*

- Π*-types require* funext *in the target.*
- *Universes require* UIP *in the target.*

Thus, to preserve definitional equality, these principles better be computational in the target, otherwise the explicit rewriting nightmare described above appears again. Function extensionality holds in OTT [5] and CubicalTT [13], but only the former interprets UIP. Alas, those theories require deep changes w.r.t. CIC, which limit the applicability of any presheaf model that would target them, if ever this were possible.

A few years later, Jaber et al. [31] went on a radically different approach to the problem. By studying presheaves as an effect-inducing model, they showed that the usual presentation was squaring the circle. Namely, it was a naturally call-by-value interpretation, which required *post-hoc* equational properties to ensure full β-reduction, which in turn were the worm in the apple relying on extensionality. By taking the alternative call-by-name decomposition, all usual equations would become definitional. The key ingredient of this property was that *all monotonocity properties were freely given.* That is, all terms would start with a universal quantification over the set of smaller conditions.

Quite regrettably, the resulting model was broken, again. By excessively favouring the negative fragment, their translation was too naïve, resulting in a breakage of the positive side, i.e. dependent elimination.

As they realized later [41], this was an instance of a much more general phenomenon. The call-by-name forcing translation was introducing effects, which allowed non-standard inhabitants of inductive types contradicting their induction principle. Dually, the categorical presheaf model recovered a call-by-name equational theory thanks to the naturality property, corresponding exactly to Fürmann's notion of purity called *thunkability* [21]. In call-by-name, this corresponds to eliminating troublemakers with parametricity, a technique better known as *realizability*.

Hence, it was deemed possible to build a parametric variant of the call-by-name presheaf construction, which would enjoy both definitional equations *and* dependent elimination, following Boulier's generous variant [12] of Bernardy-Lasson parametricity [10]. Namely, every variable in the source should be mapped to two variables in the target, the first for the effectful term, and the second for the proof that is was pure, e.g. for booleans

$$(x : \mathbb{B}) \quad \mapsto \quad (x_0 : \Pi q\, (\alpha : q \leq p).\, \mathbb{B}), (x_\varepsilon : \mathbb{B}_\varepsilon\ p\ x_0)$$

with $\mathbb{B}_\varepsilon$ an inductive predicate ensuring that its argument is either constantly true or constantly false. This definitely enforces dependent elimination by construction, but we are now back to square one w.r.t. monotonocity. Indeed, one needs to lift the parametricity proof to any lower condition, and this needs to be done definitionally or else we have to reintroduce extensionality. Fair enough, we can apply the same free construction to the parametricity component, which results in

$$\begin{aligned}&(x_0 : \Pi q\, (\alpha : q \leq p).\, \mathbb{B}),\\ &(x_\varepsilon : \Pi q\, (\alpha : q \leq p).\, \mathbb{B}_\varepsilon\ q\ (\alpha \circ x_0))\end{aligned}$$

where $\alpha \circ (-)$ stands for precomposition by α. We recover definitional equations, and by specializing x_ε to p and the identity, one can ensure that indeed x_0 is internally a boolean... except that *this does not suffice to prove dependent elimination!* All functions depend both on the effectful term and on the parametricity proof, which means that not only x_0 should be a boolean *but also* x_ε. Yet, there is no 2-parametricity proof at hand to enforce this on x_ε, which would require adding one more parametricity layer. But adding 2-parametricity

would just lift the problem one level higher, and so on, leading to the infamous *coherence hell*.

3 Strict Propositions

There are two opposite, canonical ways out of the coherence hell, the libertarian one and the authoritarian one. HoTT champions the libertarian approach, which consists in freely adding higher equalities, which are made available via the univalence principle and higher inductive types. The authoritarian tenet, contrastingly, is that there is only one way to be equal, a doctrine embodied by UIP, the principle of uniqueness of identity proofs. The resulting equality type, as found in OTT, is known as a *strict* equality.

While the HoTT stance seems *a priori* more attractive, we argue that this is currently not a tenable ideological position. There has been ongoing and very recent work [16, 28] to show that cubical type theory enjoyed the desirable syntactic properties of strong normalization and canonicity. For now, these results are still rudimentary, as they only consider a small subset of type theory. Worse, all their proofs rely on set-theoretic presheaves, and thus fundamentally take place in the contempted anti-univalent set theory.

This is an aporia. Rather, we advocate for a two-stage cubical revolution. By first implementing presheaves in a minimalistically authoritarian type theory, we will be able to unroot the cubical model from set theory, eventually leading to a setless cubical theory.

We will implement parametric presheaves in 𝔰CIC, a HoTT-inspired extension of CIC with strict propositions and UIP, described in Section 4.4 of Gilbert et al. [23]. We now briefly recall what the extension looks like.

The basic 𝔰CIC theory extends CIC with a hierarchy of sorts[1] $*_i : \square_{i+1}$ s.t. for any type $A : *_i$, if $M, N : A$ then $M \equiv N$. Such types are called definitionally proof-irrelevant. Furthermore, this sort is closed under irrelevant-codomain products, i.e. if $A : \square_i$ and $B : A \to *_j$ then $(\Pi x : A.\, B\, x) : *_{\max(i,j)}$, and it also contains inductive types defined similarly to their relevant counterparts. The tricky problem to figure out is the legality of elimination of those $*$-dwelling inductive types into $\square$. It turns out that the criterion is a restriction of the singleton elimination that applies to Coq's `Prop`. Namely, while the latter only authorizes elimination from inductive types with at most one constructor with only `Prop`-living arguments, the former further mandates that the type is not recursive, and, depending on the legality of UIP, that it only has proof-irrelevant indices. Recursivity is forbidden because it entails non-decidability of type-checking, while allowing proof-relevant indices implies UIP.

We will embrace the single-equality doctrine, and thus allow elimination over proof-relevant indices of strict propositions. In this paper, we will write 𝔰CIC for this theory, implicitly assuming that it validates UIP.

The major change introduced by this extension is not about conversion, but reduction. Because any two proofs of $\mathsf{eq}\ A\ M\ N$ are convertible, this also means that any proof of $\mathsf{eq}\ A\ M\ M$ is convertible to $\mathtt{refl}\ A\ M$, as their types coincide. Thus, to ensure canonicity, reduction must also be extended for the $\square$-valued operator

$$\mathsf{J} : \Pi(A : \square)\ (x : A)\ (P : \Pi y : A.\, \mathsf{eq}\ A\ x\ y \to \square).\\ P\ x\ (\mathtt{refl}\ A\ x) \to \Pi(y : A)\ (e : \mathsf{eq}\ A\ x\ y).\, P\ y\ e$$

with $\mathsf{J}\ A\ M\ P\ Q\ N\ E \to Q$ whenever $M \equiv N$. Notably, this makes reduction dependent on conversion.

We insist that 𝔰CIC is weaker that OTT, for it features neither function extensionality nor quotients. A representative subset of the UIP-free fragment has been formally showed to enjoy strong normalization and canonicity [22]. There is no such proof for the full 𝔰CIC extension, and worse, 𝔰CIC with an impredicative universe has been shown to break strong normalization while seemingly retaining canonicity [2]. We do not rely on impredicativity, thus we speculate that 𝔰CIC is indeed strongly normalizing.

4 Negative Forcing Translation

In this section, we will define the *prefascist*[2] model for CC_ω, the negative fragment of type theory [31]. It follows the abstract scheme described in Section 2, i.e. it is intuitively the generously parametric restriction of the call-by-name presheaf model from Jaber et al. [31], where crucially, the parametricity predicate is a *strict proposition*. As such, we will use 𝔰CIC as the target theory of this syntactic interpretation. We will assume η-rules for functions and negative products, and we will write the strict equality in $*_i$ as $M = N$, omitting the type argument and the universe level. We highlight the differences with the effectful model when they appear.

Definition 1. A base category is given by

- objects $\mathbb{P} : \square_0$
- morphisms $\mathsf{hom} : \mathbb{P} \to \mathbb{P} \to \square_0$
- identity $\mathsf{id}_p : \mathsf{hom}\ p\ p$
- composition $\circledast_{p,q,r} : \mathsf{hom}\ p\ q \to \mathsf{hom}\ q\ r \to \mathsf{hom}\ p\ r$

whose inferable arguments will be omitted and that will follow the usual infix notations for readability, furthermore subject to the following definitional equalities.

[1] As in the Agda implementation, but Coq features a single impredicative sort instead. We will not rely on impredicativity here.

[2] Sheaves were first introduced in French as *faisceaux*. Owing to this Romance etymology, and noting the lack of a corresponding formal Germanic counterpart, we are forced to conclude that the adjectivization of *sheaf* is *fascist*, fitting their authoritarian meta-theory.

$$
\begin{array}{lcl}
[\Box_i]^\Gamma_p & := & \mathtt{Psh}\ (\lambda(q\,\alpha : p).\, \mathbb{\Box}^i_q) \\
 & & \quad (\lambda(A : \Pi(q\,\alpha : p).\, \mathbb{\Box}^i_q).\, \Pi(q\,\alpha : p).\, ((A\ q\ \alpha).\mathsf{T}\ q\ \mathtt{id}_q) = ((A\ p\ \mathtt{id}_p).\mathsf{T}\ q\ \alpha)) \\
[\Pi x : A.\, B]^\Gamma_p & := & \mathtt{Psh}\ (\lambda(q\,\alpha : p).\, \Pi(x_0 : \alpha \bullet_\Gamma [\![A]\!]^\Gamma_q)\ (x_\varepsilon : (\alpha \bullet_\Gamma \{\!\{A\}\!\}^\Gamma_q)\ x_0).\ (\alpha \bullet_\Gamma [B]^{\Gamma,x}_q).\mathsf{T}\ q\ \mathtt{id}_q) \\
 & & \quad (\lambda(f : \Pi(q\,\alpha : p)\ (x_0 : \alpha \bullet_\Gamma [\![A]\!]^\Gamma_q)\ (x_\varepsilon : (\alpha \bullet_\Gamma \{\!\{A\}\!\}^\Gamma_q)\ x_0).\ (\alpha \bullet_\Gamma [B]^{\Gamma,x}_q).\mathsf{T}\ q\ \mathtt{id}_q). \\
 & & \qquad \Pi(x_0 : [\![A]\!]^\Gamma_p)\ (x_\varepsilon : \{\!\{A\}\!\}^\Gamma_p\ x_0).\ [B]^{\Gamma,x}_p.\mathsf{R}\ (\lambda(q\,\alpha : p).\, \mathtt{rw}\ (\{B\}^{\Gamma,x}_p\ q\ \alpha)\ (f\ q\ \alpha\ (\alpha \circ x_0)\ (\alpha \circ x_\varepsilon)))) \\
[x]^\Gamma_p & := & x_0\ p\ \mathtt{id}_p \\
[\lambda x : A.\, M]^\Gamma_p & := & \lambda(x_0 : [\![A]\!]^\Gamma_p)\ (x_\varepsilon : \{\!\{A\}\!\}^\Gamma_p\ x_0).\ [M]^{\Gamma,x}_p \\
[M\ N]^\Gamma_p & := & [M]^\Gamma_p\ (\lambda(q\,\alpha : p).\, \alpha \bullet_\Gamma [N]^\Gamma_q)\ (\lambda(q\,\alpha : p).\, \alpha \bullet_\Gamma \{N\}^\Gamma_q) \\
\{\Box_i\}^\Gamma_p & := & \lambda(q\,\alpha : p).\, \mathtt{refl} \\
\{\Pi x : A.\, B\}^\Gamma_p & := & \lambda(q\,\alpha : p).\, \mathtt{refl} \\
\{x\}^\Gamma_p & := & x_\varepsilon\ p\ \mathtt{id}_p \\
\{\lambda x : A.\, M\}^\Gamma_p & := & \lambda(x_0 : [\![A]\!]^\Gamma_p)\ (x_\varepsilon : \{\!\{A\}\!\}^\Gamma_p\ x_0).\ \{M\}^{\Gamma,x}_p \\
\{M\ N\}^\Gamma_p & := & \{M\}^\Gamma_p\ (\lambda(q\,\alpha : p).\, \alpha \bullet_\Gamma [N]^\Gamma_q)\ (\lambda(q\,\alpha : p).\, \alpha \bullet_\Gamma \{N\}^\Gamma_q) \\
[\![A]\!]^\Gamma_p & := & \Pi(q\,\alpha : p).\, (\alpha \bullet_\Gamma [A]^\Gamma_q).\mathsf{T}\ q\ \mathtt{id}_q \\
\{\!\{A\}\!\}^\Gamma_p\ x_0 & := & \Pi(q\,\alpha : p).\, (\alpha \bullet_\Gamma [A]^\Gamma_q).\mathsf{R}\ (\lambda(r\,\beta : q).\, \mathtt{rw}\ ((\alpha \bullet_\Gamma \{A\}^\Gamma_q)\ r\ \beta)\ (x_0\ r\ (\alpha \circledast \beta))) \\
[\![\cdot]\!]_p & := & p : \mathbb{P} \\
[\![\Gamma, x : A]\!]_p & := & [\![\Gamma]\!]_p, x_0 : [\![A]\!]^\Gamma_p, x_\varepsilon : \{\!\{A\}\!\}^\Gamma_p\ x_0
\end{array}
$$

Figure 1. Prefascist Translation

$$
\begin{array}{c}
\mathtt{id} \circledast \alpha \equiv \alpha \quad \alpha \circledast \mathtt{id} \equiv \alpha \\
(\alpha \circledast \beta) \circledast \gamma \equiv \alpha \circledast (\beta \circledast \gamma)
\end{array}
$$

By convention, variables p, q, r range over objects, and α, β, γ over morphisms. We use the binding notation $(q\,\alpha : p)$ for $(q : \mathbb{P})\ (\alpha : \mathtt{hom}\ q\ p)$. Given a term $M : \Pi(q\,\alpha : p).\, A\{q, \alpha\}$, and a morphism $\alpha : \mathtt{hom}\ q\ p$, we write $\alpha \circ M$ for the precomposition of M with α, i.e.

$$\alpha \circ M := \lambda(r\,\beta : q).\, M\ r\ (\alpha \circledast \beta).$$

Remark 1. As explained in Jaber et al. [31], the definitional equations on composition are not restrictive, even less so with a strict equality. Indeed, the Yoneda embedding

$$
\begin{array}{l}
\mathrm{Y}_{\mathtt{hom}}\ p\ q := \Sigma\alpha : (\Pi r : \mathbb{P}.\, \mathtt{hom}\ q\ r \to \mathtt{hom}\ p\ r). \\
\qquad\qquad \exists\alpha_0 : \mathtt{hom}\ p\ q.\, (\alpha = \lambda r\, k.\, \alpha_0 \circledast k)
\end{array}
$$

where $\exists$ lives in $*$, is *internally* isomorphic to $\mathtt{hom}$ in $\mathfrak{s}\mathrm{CIC}$, and can be equipped with a definitional composition by precomposition.

Definition 2. Let $p : \mathbb{P}$ and i a universe level, we define $\mathbb{\Box}^i_p$ the type of *pre-presheaves* at p as the negative record

$$\mathbb{\Box}^i_p := \mathtt{Psh} \left\{ \begin{array}{lcl} \mathsf{T} & : & \Pi(q\,\alpha : p).\, \Box_i \\ \mathsf{R} & : & (\Pi(q\,\alpha : p).\, \mathsf{T}\ q\ \alpha) \to *_i \end{array} \right\}$$

The first field is exactly the universe interpretation from Jaber et al. [31]. The main addition is the second field, which is a *strict* parametricity predicate.

As in the parametricity interpretation [10], the translation will duplicate every context variable x into a base variable x_0 and a parametricity variable x_ε, except that it does so both for base and parametricity terms as in Boulier's generous variant [12]. Critically, the sort of the type of a variable x_ε will always be $*$. This introduces a minor difference with usual parametricity, where the parametricity predicate of a type is literally the parametricity translation of that type. Since here the parametricity translation is always a proposition, we have to bundle the predicate with the type itself in the $\mathbb{\Box}$ record.

We will track the set of free variables in the translation to be able to maintain them at the right level. Given a set of variables Γ, a morphism $\alpha : \mathtt{hom}\ q\ p$ and a term M we define $\alpha \bullet_\Gamma M$ the lift of M along Γ as

$$\alpha \bullet_\Gamma M := M\{x_0 := \alpha \circ x_0, x_\varepsilon := \alpha \circ x_\varepsilon \mid x \in \Gamma\}.$$

Lemma 2. *We have the immediate definitional equalities*

$$
\begin{array}{cc}
\mathtt{id} \circ M \equiv M & (\alpha \circledast \beta) \circ M \equiv \beta \circ (\alpha \circ M) \\
\mathtt{id} \bullet_\Gamma M \equiv M & (\alpha \circledast \beta) \bullet_\Gamma M \equiv \beta \bullet_\Gamma (\alpha \bullet_\Gamma M)
\end{array}
$$

We will call $\mathtt{rw} : \Pi(A\,B : \Box).\, A = B \to A \to B$ the transport function, and leave its type arguments implicit.

The prefascist translation of CC_ω is defined at Figure 1. It is quite a mouthful, so we first state the soundness theorem, whose proof is just a variant of Jaber et al. [31], and then we walk through the translation.

Theorem 1. *If* $\Gamma \vdash_{\mathsf{CC}_\omega} M : A$ *then*

- $[\![\Gamma]\!]_p \vdash [M]^\Gamma_p : [A]^\Gamma_p.\mathsf{T}\ p\ \mathtt{id}_p$
- $[\![\Gamma]\!]_p \vdash \{M\}^\Gamma_p :$
  $[A]^\Gamma_p.\mathsf{R}\ (\lambda(q\,\alpha : p).\ \mathtt{rw}\ (\{A\}^\Gamma_p\ q\ \alpha)\ (\alpha \bullet_\Gamma [M]^\Gamma_q))$

Likewise, if $M \equiv_{\mathsf{CC}_\omega} N$ *then* $[M]^\Gamma_p \equiv_{\mathsf{sCIC}} [N]^\Gamma_p$ *and* $\{M\}^\Gamma_p \equiv_{\mathsf{sCIC}} \{N\}^\Gamma_p$.

Remark 2. In particular, if $\Gamma \vdash_{\mathsf{CC}_\omega} A : \square_i$,

- $[\![\Gamma]\!]_p \vdash [A]^\Gamma_p : \square^i_p$
- $[\![\Gamma]\!]_p \vdash \{A\}^\Gamma_p : \Pi(q\,\alpha : p).$
  $(\alpha \bullet_\Gamma [A]^\Gamma_q).\mathsf{T}\ q\ \mathtt{id}_q = [A]^\Gamma_p.\mathsf{T}\ q\ \alpha$

First, observe that all variables have a thunked type, i.e. are quantified over some $(q\ \alpha : p)$. Therefore, we get definitional monotonocity for free.

Lemma 3 (Free Monotonocity)**.** *Assume that we have* $[\![\Gamma]\!]_p \vdash M_0 : [\![A]\!]^\Gamma_p$ *and* $[\![\Gamma]\!]_p \vdash M_\varepsilon : \{\!\{A\}\!\}^\Gamma_p\ M_0$. *Then we also have for any* $\alpha : \mathtt{hom}\ q\ p$*:*

- $[\![\Gamma]\!]_p \vdash \alpha \circ M_0 : \alpha \bullet_\Gamma [\![A]\!]^\Gamma_q$
- $[\![\Gamma]\!]_p \vdash \alpha \circ M_\varepsilon : (\alpha \bullet_\Gamma \{\!\{A\}\!\}^\Gamma_q)\ (\alpha \circ M_0)$.

The $[-]^\Gamma_p$ part corresponds to the model from Jaber et al. [31] except that types are enriched with a correctness predicate, and that all arguments come with an additional correctness proof. We also use an explicit lift $\alpha \bullet_\Gamma (-)$ every time we thunk w.r.t. α in the style of Miquel [39] instead of accumulating constraints as in Jaber et al. [31], but this is merely cosmetic.

The new part is the $\{-\}^\Gamma_p$ translation, which corresponds to the parametricity proof. There are several points to highlight. First, as already explained, this translation always lives in a proof-irrelevant sort. Second, applying the parametricity predicate of a type A on some thunk x_0 requires a type cast, as exemplified in the type soundness for $\{M\}^\Gamma_p$. There, $\alpha \bullet_\Gamma [M]^\Gamma_q$ has type $(\alpha \bullet_\Gamma [A]^\Gamma_q).\mathsf{T}\ q\ \mathtt{id}_q$, but $[A]^\Gamma_p.\mathsf{R}$ expects a thunk of type $\Pi(q\,\alpha : p).\ [A]^\Gamma_p.\mathsf{T}\ q\ \alpha$. Luckily, the parametricity proof $\{A\}^\Gamma_p$ precisely allows to cast from one type to the other. The bit of magic that makes this work is that this equality is convertible to the universe predicate as in Remark 2.

We do not detail the proof of Theorem 1, it is a simple mutual induction over derivations. A direct consequence though is the following theorem.

Theorem 2. *The prefascist translation is a model of* CC_ω.

The soundness theorem for the negative fragment does not rely on proof-irrelevance. In particular, replacing $*_i$ with $\square_i$ in the definition of $\square^i$ would not have any effect yet.

5 Positive Types

So far, the additional parametricity layer has been but a burden, we did not use it actively. Its only interest is to recover dependent elimination that was lost in the naive presheaf model. We therefore show how by translating inductive types in this section.

Once again the translation is very similar to Jaber et al. [31], up to the fact that we now translate a type $\mathcal{I}$ into a base type $\mathcal{I}_0$ and a strict predicate $\mathcal{I}_\mathsf{R}$ over that type. As in the negative case, the universe parametricity equation will hold definitionally. The translation is straightforward.

Definition 3. Given an inductive type $\mathcal{I}$ with parameters $\Gamma_\mathcal{I}$, indices $\Delta_\mathcal{I}$ and constructors c_i, we define $\mathcal{I}_0$ with non-uniform parameters $[\![\Gamma_\mathcal{I}]\!]_p$, indices $[\![\Delta_\mathcal{I}]\!]_p$ without the leading p, and constructors c_{0i} whose types are the pointwise $[\![-]\!]^{\Gamma_\mathcal{I}}_p$ translation, with $[\mathcal{I}]^\Xi_q.\mathsf{T}\ r\ \beta$ replaced by $\mathcal{I}_0\ r$ and quantifications over $[\mathcal{I}]^\Xi_q.\mathsf{R}$ erased, for any Ξ, q, r and β. We define the $\mathcal{I}_\mathsf{R}$ the same, except that is has one more index $x_0 : \mathcal{I}_0\ \Gamma_\mathcal{I}\ \Delta_\mathcal{I}$, that it lands in $*$ and that its constructors $c_{\mathsf{R}i}$ have type $\{\!\{-\}\!\}^{\Gamma_\mathcal{I}}_p$ applied to the corresponding thunked $\mathcal{I}_0$ constructor, with replacement for both translations of the underlying inductive.

The usual strict positivity condition ensures that the replacements of the self inductive is sound[3]. We give somewhat clearer representative examples of the translation in Figure 2, for booleans, lists and equality. Note e.g. how $\mathtt{cons}_0$ does not mention the parametricity predicate on its recursive call, but $\mathtt{cons}_\mathsf{R}$ does. We extend the translation in Figure 3 using a bit of syntactic sugar for context handling.

It is possible to write recursors for any inductive type, satisfying the expected definitional equations. Nonetheless, the resulting terms are quite hideous and unreadable. Hence, we explain instead at a high level how strict parametricity saves the day on the particular example of booleans. Barring conversion, it suffices to prove that

$$\begin{array}{lcl} P & : & \Pi(x_0 : [\![\mathbb{B}]\!]_p)\ (x_\varepsilon : \{\!\{\mathbb{B}\}\!\}_p\ x_0).\ \square \\ p_\mathtt{t} & : & P\ (\lambda(q\,\alpha : p).\ \mathtt{true}_0\ q)\ (\lambda(q\,\alpha : p).\ \mathtt{true}_\mathsf{R}\ q) \\ p_\mathtt{f} & : & P\ (\lambda(q\,\alpha : p).\ \mathtt{false}_0\ q)\ (\lambda(q\,\alpha : p).\ \mathtt{false}_\mathsf{R}\ q) \\ x_0 & : & \Pi(q\,\alpha : p).\ \mathbb{B}_0\ q \\ x_\varepsilon & : & \Pi(q\,\alpha : p).\ \mathbb{B}_\mathsf{R}\ q\ (\alpha \circ x_0) \end{array}$$

implies $P\ x_0\ x_\varepsilon$. The trick is the following: $x_\varepsilon\ p\ \mathtt{id}_p$ will provide an irrelevant proof that x_0 is indeed a boolean. From this, we get a propositional equality between x_0 and its η-expansion:

$$\mathtt{if}\ x_0\ p\ \mathtt{id}_p\ \mathtt{then}\ (\lambda q\,\alpha.\ \mathtt{true}_0\ q)\ \mathtt{else}\ (\lambda q\,\alpha.\ \mathtt{false}_0\ q)$$

which we use to rewrite in $P\ x_0\ x_\varepsilon$. This step is a critical use of UIP. By dependent elimination over $x_0\ p\ \mathtt{id}_p$, we are almost done. Thanks to the irrelevance of $\mathbb{B}_\mathsf{R}$, any

[3] Nested inductive types would require induction-induction though.

$$\begin{array}{l}
\texttt{Ind}\ \mathbb{B}_0\ (p : \mathbb{P})\ : \square :=\\
\mid \texttt{true}_0 : \mathbb{B}_0\ p\\
\mid \texttt{false}_0 : \mathbb{B}_0\ p\\
\\
\texttt{Ind}\ \texttt{list}_0\ (p : \mathbb{P})\ (A_0 : [\![\square]\!]_p)\ (A_\varepsilon : \{\!\{\square\}\!\}_p\ A_0)\ : \square :=\\
\mid \texttt{nil}_0 : \texttt{list}_0\ p\ A_0\ A_\varepsilon\\
\mid \texttt{cons}_0 : \Pi(x_0 : [\![A]\!]_p)\ (x_\varepsilon : \{\!\{A\}\!\}_p\ x_0).\ (\Pi(q\ \alpha : p).\ \texttt{list}_0\ q\ (\alpha \circ A_0)\ (\alpha \circ A_\varepsilon)) \to \texttt{list}_0\ p\ A_0\ A_\varepsilon\\
\\
\texttt{Ind}\ \texttt{eq}_0\ (p : \mathbb{P})\ (A_0 : [\![\square]\!]_p)\ (A_\varepsilon : \{\!\{\square\}\!\}_p\ A_0)\ (x_0 : [\![A]\!]_p)\ (x_\varepsilon : \{\!\{A\}\!\}_p\ x_0)\ : \Pi(y_0 : [\![A]\!]_p)\ (y_\varepsilon : \{\!\{A\}\!\}_p\ y_0).\ \square :=\\
\mid \texttt{refl}_0 : \texttt{eq}_0\ p\ A_0\ A_\varepsilon\ x_0\ x_\varepsilon\ x_0\ x_\varepsilon
\end{array}$$

$$\begin{array}{l}
\texttt{Ind}\ \mathbb{B}_\mathsf{R}\ (p : \mathbb{P})\ : (\Pi(q\ \alpha : p).\ \mathbb{B}_0\ p) \to * :=\\
\mid \texttt{true}_\mathsf{R} : \mathbb{B}_\mathsf{R}\ p\ (\lambda(q\ \alpha : p).\ \texttt{true}_0\ q)\\
\mid \texttt{false}_\mathsf{R} : \mathbb{B}_\mathsf{R}\ p\ (\lambda(q\ \alpha : p).\ \texttt{false}_0\ q)\\
\\
\texttt{Ind}\ \texttt{list}_\mathsf{R}\ (p : \mathbb{P})\ (A_0 : [\![\square]\!]_p)\ (A_\varepsilon : \{\!\{\square\}\!\}_p\ A_0)\ : (\Pi(q\ \alpha : p).\ \texttt{list}_0\ q\ (\alpha \circ A_0)\ (\alpha \circ A_\varepsilon)) \to * :=\\
\mid \texttt{nil}_\mathsf{R} : \texttt{list}_\mathsf{R}\ p\ A_0\ A_\varepsilon\ (\lambda(q\ \alpha : p).\ \texttt{nil}_0\ q\ (\alpha \circ A_0)\ (\alpha \circ A_\varepsilon))\\
\mid \texttt{cons}_\mathsf{R} : \Pi(x_0 : [\![A]\!]_p)\ (x_\varepsilon : \{\!\{A\}\!\}_p\ x_0).\\
\qquad \Pi(l_0 : \Pi(q\ \alpha : p).\ \texttt{list}_0\ q\ (\alpha \circ A_0)\ (\alpha \circ A_\varepsilon))\ (l_\varepsilon : \Pi(q\ \alpha : p).\ \texttt{list}_\mathsf{R}\ q\ (\alpha \circ A_0)\ (\alpha \circ A_\varepsilon)\ (\alpha \circ l_0)).\\
\qquad\quad \texttt{list}_\mathsf{R}\ p\ A_0\ A_\varepsilon\ (\lambda(q\ \alpha : p).\ \texttt{cons}_0\ q\ (\alpha \circ A_0)\ (\alpha \circ A_\varepsilon)\ (\alpha \circ x_0)\ (\alpha \circ x_\varepsilon)\ (\alpha \circ l_0))\\
\\
\texttt{Ind}\ \texttt{eq}_\mathsf{R}\ (p : \mathbb{P})\ (A_0 : [\![\square]\!]_p)\ (A_\varepsilon : \{\!\{\square\}\!\}_p\ A_0)\ (x_0 : [\![A]\!]_p)\ (x_\varepsilon : \{\!\{A\}\!\}_p\ x_0)\ :\\
\quad \Pi(y_0 : [\![A]\!]_p)\ (y_\varepsilon : \{\!\{A\}\!\}_p\ y_0).\ (\Pi(q\ \alpha : p).\ \texttt{eq}_0\ q\ (\alpha \circ A_0)\ (\alpha \circ A_\varepsilon)\ (\alpha \circ x_0)\ (\alpha \circ x_\varepsilon)\ (\alpha \circ y_0)\ (\alpha \circ y_\varepsilon)) \to * :=\\
\mid \texttt{refl}_\mathsf{R} : \texttt{eq}_\mathsf{R}\ p\ A_0\ A_\varepsilon\ x_0\ x_\varepsilon\ x_0\ x_\varepsilon\ (\lambda(q\ \alpha : p).\ \texttt{refl}_0\ q\ (\alpha \circ A_0)\ (\alpha \circ A_\varepsilon)\ (\alpha \circ x_0)\ (\alpha \circ x_\varepsilon))
\end{array}$$

Figure 2. Examples of the Prefascist Inductive Translation

$$\begin{array}{ll}
[\mathcal{I}]^\Gamma_p := \lambda[\![\Gamma_\mathcal{I}, \Delta_\mathcal{I}]\!]_p.\ \mathsf{Psh}\ (\lambda(q\ \alpha : p).\ \mathcal{I}_0\ q\ (\alpha \circ \Gamma_\mathcal{I})\ (\alpha \circ \Delta_\mathcal{I}))\ (\mathcal{I}_\mathsf{R}\ p\ \Gamma_\mathcal{I}\ \Delta_\mathcal{I}) & \{\mathcal{I}\}^\Gamma_p := \lambda[\![\Gamma_\mathcal{I}, \Delta_\mathcal{I}]\!]_p.\ \lambda(q\ \alpha : p).\ \mathsf{refl}\\
[c_i]^\Gamma_p := c_{0i}\ p & \{c_i\}^\Gamma_p := c_{\mathsf{R}i}\ p
\end{array}$$

Figure 3. Translation Extension

proof of the right type is convertible to the parametricity proofs used in p_t and p_f. This definitional irrelevance is also critical, had we not guillotined the higher coherences, we would not have been able to conclude.

The same trick applies to any inductive type. Compared to Jaber et al. [31], the additional parametricity proof is, in essence, internalizing that any inhabitant of an inductive type is strictly equal to its η-expansion and thus that all predicates are linear. Therefore:

Theorem 3. *The prefascist translation is a model of* CIC.

6 Canonicity

Let us call $\mathfrak{Psh}_\mathbb{P}$ the type theory generated by the prefascist translation over the category $\mathbb{P}$. By virtue of the soundness theorem, $\mathfrak{Psh}_\mathbb{P}$ contains CIC. We have a syntactic model, so it is trivial to peek at the whereabouts of canonicity.

Theorem 4. *If $\mathbb{P}$ has a weak terminal element* $\mathbf{1}$*, then $\mathfrak{Psh}_\mathbb{P}$ enjoys canonicity up to irrelevance.*

Proof. Let us look at the case of booleans. A closed term $\vdash_{\mathfrak{Psh}_\mathbb{P}} M : \mathbb{B}$ is given by a pair of terms M_0, M_ε of type

$$\begin{array}{l}
p : \mathbb{P} \vdash_{\mathfrak{s}\mathsf{CIC}} M_0 : \mathbb{B}_0\ p\\
p : \mathbb{P} \vdash_{\mathfrak{s}\mathsf{CIC}} M_\varepsilon : \mathbb{B}_\mathsf{R}\ p\ (\lambda(q\ \alpha : p).\ M_0\{p := q\})
\end{array}$$

By canonicity of $\mathfrak{s}$CIC, $M_\varepsilon\{p := \mathbf{1}\}$ ensures that either $(\lambda(q\ \alpha : \mathbf{1}).\ M_0\{p := q\}) \equiv (\lambda(q\ \alpha : \mathbf{1}).\ \texttt{true}_0\ q)$ or $(\lambda(q\ \alpha : \mathbf{1}).\ M_0\{p := q\}) \equiv (\lambda(q\ \alpha : \mathbf{1}).\ \texttt{false}_0\ q)$. Either way, by applying both sides to the weak terminal morphism $!_q : \mathsf{hom}\ q\ \mathbf{1}$, we get canonicity of M_0.

While there is no proof in sight that M_ε is a constant constructor, it does not matter because by canonicity of M_0, it is necessarily convertible to such a constructor. □

We insist that canonicity is relative to the target theory, and that there is currently no proof that $\mathfrak{s}$CIC enjoys canonicity, although it is believed so [2].

7 A Categorical Equivalence

It so happens that **Set** models strict propositions [23], where they are interpreted by subsingletons. Observe that, by taking **Set** as our target type theory, a base category as in Definition 1 is a (small) category. We can thus apply the prefascist translation and compare the result to usual set-theoretic presheaves. We will ignore size issues.

First, we make explicit the unfolding the prefascist translation on closed expressions.

Lemma 4. *A type* $\mathbf{A}$ *in the prefascist translation is given by, for any* $p : \mathbb{P}$*,*

- *a family of sets* $A_p : \Pi(q\,\alpha : p).\,\mathbf{Set}$
- *a predicate* $(-) \Vdash_p \mathbf{A} \subseteq (\Pi(q\,\alpha : p).\,A_p\ q\ \alpha)$

s.t. for all $p, q, \alpha : \mathtt{hom}\ q\ p$*, we have* $A_q\ q\ \mathtt{id}_q = A_p\ q\ \alpha$*.*

The two components are the set-theoretical intepretation of a universally quantified pre-presheaf, and the equation is given by parametricity of the latter.

Definition 4. If $\mathbf{A}$ is a type and $p : \mathbb{P}$, we define $\mathsf{El}_p\ \mathbf{A}$, the set of partial elements of $\mathbf{A}$ at p, as

$$\{x : \Pi(q\,\alpha : p).\,A_p\ q\ \alpha \mid \forall(q\,\alpha : p).\,(\alpha \circ x) \Vdash_q \mathbf{A}\}.$$

Lemma 5. *Given two types* $\mathbf{A}$ *and* $\mathbf{B}$*, a function* f *of type* $\mathbf{A} \to \mathbf{B}$ *is given by a* $\mathbb{P}$*-indexed family of functions*

$$f_p : \mathsf{El}_p\ \mathbf{A} \to B_p\ p\ \mathtt{id}_p$$

which preserves the realizability predicate, i.e.

$$\forall p : \mathbb{P}.\ \forall x : \mathsf{El}_p\ \mathbf{A}.\ (\lambda(q\,\alpha : p).\,f_q\ (\alpha \circ x)) \Vdash_p \mathbf{B}.$$

Prefascist types and functions between them form a category in a straightforward way, with the categorical equations satisfied *definitionally*. A natural question to ask is its relationship with the usual presheaf category. We will describe a standard presheaf $\mathcal{A}$ as

- a $\mathbb{P}$-indexed family of sets $\mathcal{A}_p : \mathbf{Set}$
- restrictions $\theta_p^{\mathcal{A}} : \Pi(q\,\alpha : p).\,\mathcal{A}_p \to \mathcal{A}_q$

satisfying the usual functoriality equations.

Definition 5. Let $\mathcal{A}$ be a presheaf. We define the prefascist type $\mathfrak{S}(\mathcal{A})$ as

$$\begin{aligned}(\mathfrak{S}(\mathcal{A}))_p &:= \lambda(q\,\alpha : p).\,\mathcal{A}_q\\ \hat{x} \Vdash_p \mathfrak{S}(\mathcal{A}) &:= \forall(q\,\alpha : p).\,\hat{x}\ q\ \alpha = \theta_p^{\mathcal{A}}\ q\ \alpha\ (\hat{x}\ p\ \mathtt{id}_p).\end{aligned}$$

Similarly, let $\mathbf{A}$ be a prefascist type. We define the presheaf $\mathscr{L}(\mathbf{A})$ as

$$\begin{aligned}(\mathscr{L}(\mathbf{A}))_p &:= \mathsf{El}_p\ \mathbf{A}\\ \theta_p^{\mathscr{L}(\mathbf{A})}\ q\ \alpha\ x &:= \alpha \circ x.\end{aligned}$$

Theorem 5. $\mathfrak{S}(-)$ *and* $\mathscr{L}(-)$ *are functors and define an equivalence of categories.*

The proof can be carried in any model of $\mathfrak{s}$CIC additionally featuring function extensionality. This shows that as long as one is willing to accept funext in the metatheory, our new notion of presheaves is virtually the same as the traditional one. But ours shines by the fact that it makes all conversions purely definitional without relying on any extensionality axiom.

8 Russian Constructivism

Russian constructivism is a splinter group from orthodox intuitionism. Its defining feature [8], that puts it apart both from Bishop's minimal intuitionism and Brouwer's radical constructivism[4], is that they decree Markov's principle (MP), usually stated as

$$\Pi(f : \mathbb{N} \to \mathbb{B}).\ \neg\neg(\Sigma n : \mathbb{N}.\,f\ n = \mathtt{true}) \to \Sigma n : \mathbb{N}.\,f\ n = \mathtt{true}.$$

It is equivalent to saying that a Turing machine that does not loop necessarily terminates. While not provable in intuitionistic logic [44], Markov's principle has been long known to be somewhat compatible with it, insofar as it does not break the witness property. Moreover, MP turns out to be necessary for many completeness results [25, 36].

There are various justifications of MP in the literature, with varying degrees of convincing strength. We review the most representative ones in the remainder of this section.

The oldest one is probably Kleene realizability [34], where MP is realized via an unbounded loop . The realizer does not use at all the doubly-negated premise, and its validity critically relies in MP being provable in the meta-theory. As such, it is arguably *cheating*. The requirement of arbitrary fixpoints in the language of realizers is also an issue, as in general it entails undecidability of the realizability relation.

Dialectica [6] also inteprets MP, in a more intricate way. Contrarily to Kleene realizability, the Dialectica realizer of a negation contains exploitable data [35]. MP is realized by extracting the integer witness from its double-negated premise, allowing to be at the same time more efficient than a clueless loop, and less demanding logically as this does not require meta-MP.

Coquand and Hofmann [15] gave a syntactic model of PRA^{ω} + MP into PRA^{ω} itself, which was allegedly a variant of Kripke semantics enriched with an additional layer looking like Friedman's A-translation [20].

More recently, Herbelin [25] and Ilik [29] described how to implement MP using a weak form of delimited continuations. Their realizer creates a fresh, statically-bound exception and uses it to feed the doubly-negated premise with a failing term. The exception carries an integer as an argument, which is the witness of the existential being proved. The realizer thus catches the exception ensuring it does not escape, and returns its carried value. This is valid because the double negation

[4] Members of the Markovian school also insist that all proofs should correspond to a realizer. In first-order logic this often leads to the acceptance of some form of Church's thesis [38]. We argue that according to a lax interpretation of this requirement, dependent types fulfill it *by construction*, since in CIC proofs are literally programs.

is applied to a hereditarily positive type, which guarantees that the fresh exception cannot be involuntarily caught in a closure and escape its handler.

So far, all known models of a dependent type theory extended with MP relied on classical logic at the meta-level, and in particular broke decidability of typechecking and computational canonicity. As a matter of fact, there were two distinct countermodels to the derivability of MP in CIC [17, 40].

In the remainder of this paper, we give a generalization of the Coquand-Hofmann model to CIC, and make the proof more synthetic. The resulting model is the composition of two simpler model transformations, namely an instance of the prefascist interpretation described above, followed by an exceptional interpretation [40]. We will call the resulting system CCCP, the *Calculus of Constructions with Completeness Principles*, which will be an extension of CIC validating MP and enjoying by construction canonicity and the like. The whole process can be pictorially summarized as

$$\mathrm{CCCP} \xrightarrow{\ \mathsf{Exn}\ } \mathrm{CIC} + \mathcal{E} \xrightarrow{\ \mathsf{Psh}\ } \mathfrak{s}\mathrm{CIC}$$

where $\mathcal{E}$ is an *exotic* type introduced by the presheaf intepretation together with a few combinators, and the exceptional translation is performed with $\mathcal{E}$ as the type of exceptions.

9 Exception is the Rule

We briefly summarize the exceptional model [40] in this section. We will not dwell too much on the details for lack of space, instead we invite the reader to refer to the cited paper for a more in-depth exposition.

Intuitively, given a type theory $\mathcal{T}$ containing CIC, together with a type $\vdash_{\mathcal{T}} \mathbb{E} : \square$ that will stand for the type of exceptions, the exceptional translation builds a new type theory $\mathcal{T}_{\mathbb{E}}$ where failure is allowed, under the form of terms

$$\vdash_{\mathcal{T}_{\mathbb{E}}} \mathbf{E} : \square$$
$$\vdash_{\mathcal{T}_{\mathbb{E}}} \mathtt{raise} : \Pi A : \square.\, \mathbf{E} \to A$$

where $\mathbf{E}$ reflects $\mathbb{E}$, and subject to equations such as

$$\begin{aligned} \mathtt{raise}\ (\Pi x : A.\, B)\ e &\equiv \lambda x : A.\, \mathtt{raise}\ B\ e \\ \mathtt{match}\ (\mathtt{raise}\ \mathcal{I}\ e)\ \mathtt{return}\ P\ \mathtt{with}\ \vec{p} &\equiv \mathtt{raise}\ P\ e. \end{aligned}$$

Said otherwise, we have a primitive that allows to escape from the surrounding context, i.e. exceptions. They obey a call-by-name equational theory, so they are quite distinct from their realworld counterparts. In particular, it is not possible to catch exceptions at an arbitrary type. They can only be caught over inductive types, in which case the `catch` primitive is a generalization of the usual dependent eliminator. For instance, exception-catching on booleans is given by the primitive

$$\begin{aligned} &\vdash_{\mathcal{T}_{\mathbb{E}}} \mathtt{catch}_{\mathbb{B}} : \Pi P : \mathbb{B} \to \square. \\ &\quad P\ \mathtt{true} \to P\ \mathtt{false} \to (\Pi e : \mathbf{E}.\, P\ (\mathtt{raise}\ \mathbb{B}\ e)) \to \\ &\qquad \Pi b : \mathbb{B}.\, P\ b \end{aligned}$$

which enjoys the obvious reduction rules. Alternatively, this can be presented as a pattern-matching extension where branches may optionally handle an exception.

In what follows we fix a theory $\mathcal{T}$ containing CIC, a type $\vdash_{\mathcal{T}} \mathbb{E} : \square_0$ and an arbitrary family of terms of type $\vdash_{\mathcal{T}} \Omega_i : \mathbb{E} \to \mathbb{\square}_i$, where $\mathbb{\square}_i := \Sigma A : \square_i.\, (\mathbb{E} \to A)$[5]. This family of terms is used to fix the meaning of judgments of the shape $\vdash_{\mathcal{T}_{\mathbb{E}}} M : \mathtt{raise}\ \square\ N$. Note that the type of Ω_i is always inhabited, hence such a term can always be constructed regardless of the actual value of $\mathbb{E}$. Once again, $\mathcal{T}_{\mathbb{E}}$ is given by a syntactic translation into $\mathcal{T}$. The basic idea is that we enrich types through the interpretation with their own `raise` function. In particular,

$$\begin{aligned} \vdash_{\mathcal{T}_{\mathbb{E}}} A : \square \quad &\mapsto \quad \vdash_{\mathcal{T}} [A] : \mathbb{\square} \\ \vdash_{\mathcal{T}_{\mathbb{E}}} M : A \quad &\mapsto \quad \vdash_{\mathcal{T}} [M] : [A].\pi_1. \end{aligned}$$

Translation of the negative fragment is formally given at Figure 4.

Inductive types can be added straightforwardly, we sketch their translation here. Given an inductive type $\mathcal{I}$, we set $[\mathcal{I}] := (\mathcal{I}_{\mathbb{E}}, \mathcal{I}_{\varnothing})$ where $\mathcal{I}_{\mathbb{E}}$ is an inductive type in $\mathcal{T}$ whose constructors have as type the pointwise translation of those of $\mathcal{I}$, plus one additional constructor $\mathcal{I}_{\varnothing} : \mathbb{E} \to \mathcal{I}_{\mathbb{E}}$. Constructors in the source theory are then interpreted by their counterpart from the extended inductive in the target theory. This interpretation is readily adapted to parameters and indices. As an example, we show the translation on lists below.

$$\begin{aligned} &\mathtt{Inductive}\ \mathtt{list}_{\mathbb{E}}\ (A : \mathbb{\square}) : \square := \\ &\mid \mathtt{nil}_{\mathbb{E}} : \mathtt{list}_{\mathbb{E}}\ A \\ &\mid \mathtt{cons}_{\mathbb{E}} : A.\pi_1 \to \mathtt{list}_{\mathbb{E}}\ A \to \mathtt{list}_{\mathbb{E}}\ A \\ &\mid \mathtt{list}_{\varnothing} : \mathbb{E} \to \mathtt{list}_{\mathbb{E}}\ A \end{aligned}$$

This fully defines the introduction rules for inductive types. As usual, the subtle part lies in the interpretation of dependent elimination. Essentially, we interpret pattern-matching pointwise. This almost works thanks to the preservation of typing, except for the additional constructor which is not accounted for. For this one constructor, we use instead the raising primitive given by the return type of the pattern-matching, which corresponds computationally with reraising the exception in the error case.

Theorem 6. *$\mathcal{T}_{\mathbb{E}}$ is a model of* CIC.

[5] We will reuse in this section the same kind of notations as for the forcing translation, because all these syntactic models follow the same pattern. They should not be confused though.

$$
\begin{array}{lcllcl}
[\Box_i] & := & (\mathbb{\Box}_i, \Omega_i) & \mathbb{\Box}_i & := & \Sigma A : \Box_i.\ (\mathbb{E} \to A) \\
[x] & := & x & [\![A]\!] & := & [A].\pi_1 \\
[\lambda x : A.\, M] & := & \lambda x : [\![A]\!].\ [M] & [A]_\varnothing & := & [A].\pi_2 \\
[M\ N] & := & [M]\ [N] & [\![\cdot]\!] & := & \cdot \\
[\Pi x : A.\, B] & := & (\Pi x : [\![A]\!].\ [\![B]\!], (\lambda (e : \mathbb{E})\ (x : [\![A]\!]).\ [B]_\varnothing\ e)) & [\![\Gamma, x : A]\!] & := & [\![\Gamma]\!], x : [\![A]\!] \\
[\mathbf{E}] & := & (\mathbb{E}, (\lambda e : \mathbb{E}.\, e)) &
\end{array}
$$

Figure 4. Exceptional Translation

We do not further detail the implementation of the aforementioned exceptional combinators, although it is quite direct. Instead, we turn to the metatheoretical properties of the translation. To clarify the vocabulary, we say that a type A is inhabited in a theory $\mathcal{T}$ when there is a closed term $\vdash_{\mathcal{T}} M : A$.

It is immediate to make the following observation.

Lemma 6. *$\mathcal{T}_{\mathbb{E}}$ is consistent iff $\mathbb{E}$ is not inhabited in $\mathcal{T}$.*

Proof. We have $\bot_{\mathbb{E}} \cong \mathbb{E}$ in $\mathcal{T}$. □

It is more generally possible to show a weak form of canonicity for $\mathcal{T}_{\mathbb{E}}$, which is obtained directly from the constructor-adding interpretation of inductive types.

Lemma 7. *Assuming $\mathcal{T}$ enjoys canonicity, if $\mathcal{I}$ is an inductive type in $\mathcal{T}_{\mathbb{E}}$ with constructors $\vec{c}$ and $\vdash_{\mathcal{T}_{\mathbb{E}}} M : \mathcal{I}$, then either $M \equiv c_i\ \vec{N}$ for some i or $M \equiv \mathtt{raise}\ \mathcal{I}\ e$ for some $\vdash_{\mathcal{T}_{\mathbb{E}}} e : \mathbf{E}$.*

Note that a direct consequence of this property is that if $\mathbb{E}$ is not inhabited in $\mathcal{T}$, then $\mathcal{T}_{\mathbb{E}}$ does enjoy usual canonicity.

Interestingly, this model can already be used to show a very weak form of MP. Rather than the full fledged internal principle, it is possible to show the admissibility of *Markov's rule* in CIC. This derivation rule is defined as

$$\frac{\vdash_{\mathsf{CIC}} M : \neg\neg P}{\vdash_{\mathsf{CIC}} \langle M \rangle : P}$$

where P is some hereditarily positive type, e.g. $P := \Sigma n : \mathbb{N}.\ f\ n = \mathtt{true}$. Note that this rule crucially requires the proof environment to be empty, thus forbidding to derive MP.

Lemma 8 (Friedman's trick)**.** *Markov's rule is admissible in* CIC.

Proof. The proof goes as follows. Let $\vdash_{\mathsf{CIC}} M : \neg\neg P$. By soundness of the exceptional model, for any parameter $\mathbb{E}$ we get a proof $\vdash_{\mathsf{CIC}} [M] : [\![\neg\neg P]\!]$. Unfolding the definitions, we get in CIC an isomorphism $[\![\neg\neg P]\!] \cong ([\![P]\!] \to \mathbb{E}) \to \mathbb{E}$. Now, the crux of the proof lies in the observation that if P is hereditarily positive, then there is actually a closed CIC term $\theta_P : [\![P]\!] \to P + \mathbb{E}$. Such a term is called a *storage operator* [37], and is defined by recursively matching over its argument and reconstructing a purified proof, and propagating the exception in case of error. The ability to do so is fundamentally tied to the positivity of P, which allows to locally force exceptions on its inhabitants, and intuitively corresponds to enforcing a delimited form of call-by-value.

We now have all the required ingredients. Let us set $\mathbb{E} := P$, in which case $[M]$ provides us with a term $([\![P]\!] \to P) \to P$, and the storage operator has type $\theta_P : [\![P]\!] \to P + P$. By plugging both together with get a closed proof of P in CIC, hence concluding the proof. □

10 From the Rule to the Principle

We argue that implementing Markov's rule is *almost* sufficient to get the full Markov's principle. Obviously, it is not enough *per se*. We can actually show that $\mathcal{T}_{\mathbb{E}}$ disproves MP when $\mathbb{E}$ is inhabited [40], which is the case for the instance used in the extraction of Markov's rule. Yet, the key point is that this rule already features the necessary basic computational extension to implement MP with a little bit of tweaking. The rationale is the following. Any finite number of closed uses of Markov's rule can be eliminated via the above trick. What prevents us from generalizing to MP is the requirement to pick $\mathbb{E}$ beforehand, as it will be fixed once and for all inside the exceptional translation.

What if we were able to dynamically change the value of $\mathbb{E}$ to track the set of Markov's rules being applied?

Definition 6 (Dynamic prompt)**.** We now assume that the target theory of the exceptional translation is an extension of CIC dubbed CIC + $\mathcal{E}$ which contains the data described at Figure 5. We furthermore assume that $\mathcal{E}$ is not inhabited.

This extension is intuitively defined in a theory where there is an ambient global cell p of type $\mathbb{N} \to \mathbb{B}$, which

$$
\begin{array}{lcl}
\mathcal{E} & : & \square_0 \\
\mathtt{local} & : & (\mathbb{N} \to \mathbb{B}) \to \square_i \to \square_i \\
\mathtt{return} & : & \Pi(A : \square)\,(\varphi : \mathbb{N} \to \mathbb{B}).\, A \to \mathtt{local}\ \varphi\ A \\
\mathtt{of}_{\to} & : & \Pi(A\,B : \square)\,(\varphi : \mathbb{N} \to \mathbb{B}).\, \mathtt{local}\ \varphi\ (A \to B) \to \mathtt{local}\ \varphi\ A \to \mathtt{local}\ \varphi\ B \\
\mathtt{to}_{\to} & : & \Pi(A\,B : \square)\,(\varphi : \mathbb{N} \to \mathbb{B}).\, (\mathtt{local}\ \varphi\ A \to \mathtt{local}\ \varphi\ B) \to \mathtt{local}\ \varphi\ (A \to B) \\
\mathtt{of}_{\Sigma} & : & \Pi(P : \mathbb{N} \to \square)\,(\varphi : \mathbb{N} \to \mathbb{B}).\, \mathtt{local}\ \varphi\ (\Sigma n : \mathbb{N}.\, P\ n) \to \Sigma n : \mathbb{N}.\, \mathtt{local}\ \varphi\ (P\ n) \\
\mathtt{to}_{\Sigma} & : & \Pi(P : \mathbb{N} \to \square)\,(\varphi : \mathbb{N} \to \mathbb{B}).\, (\Sigma n : \mathbb{N}.\, \mathtt{local}\ \varphi\ (P\ n)) \to \mathtt{local}\ \varphi\ (\Sigma n : \mathbb{N}.\, P\ n) \\
\mathtt{of}_{=} & : & \Pi(M\,N : \mathbb{N})\,(\varphi : \mathbb{N} \to \mathbb{B}).\, \mathtt{local}\ \varphi\ (M = N) \to M = N \\
\mathtt{to}_{=} & : & \Pi(M\,N : \mathbb{N})\,(\varphi : \mathbb{N} \to \mathbb{B}).\, M = N \to \mathtt{local}\ \varphi\ (M = N) \\
\mathtt{of}_{\mathcal{E}} & : & \Pi(\varphi : \mathbb{N} \to \mathbb{B}).\, \mathtt{local}\ \varphi\ \mathcal{E} \to (\Sigma n : \mathbb{N}.\, \varphi\ n = \mathtt{true}) + \mathcal{E} \\
\mathtt{to}_{\mathcal{E}} & : & \Pi(\varphi : \mathbb{N} \to \mathbb{B}).\, ((\Sigma n : \mathbb{N}.\, \varphi\ n = \mathtt{true}) + \mathcal{E}) \to \mathtt{local}\ \varphi\ \mathcal{E}
\end{array}
$$

Figure 5. Dynamic Prompt API

can be locally mutated. Only types are allowed to really observe this cell, otherwise this would break the behaviour of the pure CIC fragment. A typical example is given by the `local` modality, which allows evaluating its type argument inside a cell set to the pointwise boolean or between the current cell value and the function argument. Likewise, $\mathcal{E}$ is inhabited if there is some $n : \mathbb{N}$ s.t. $\mathsf{p}\ n = \mathtt{true}$. Its commutation principles with `local` reflect this semantics. Finally, `return` is the unit of the modality, and captures the monotonocity of the pure fragment.

Let CCCP be the theory obtained by unrolling the model construction of the previous section by setting $\mathcal{T} := \mathsf{CIC} + \mathcal{E}$ and $\mathbb{E} := \mathcal{E}$. We immediately get the following results.

Lemma 9. *We have the following.*

- CCCP *contains* CIC.
- *If* $\mathsf{CIC} + \mathcal{E}$ *is consistent then so is* CCCP.
- *If* $\mathsf{CIC} + \mathcal{E}$ *enjoys canonicity then so does* CCCP.

Proof. The first property follows from soundness, and the two last ones from the uninhabitedness of $\mathcal{E}$. □

Note that we are walking a tight rope here. Consistency of CCCP only requires the absence of a term M s.t. $\vdash_{\mathsf{CIC}+\mathcal{E}} M : \mathcal{E}$, which is a much weaker property than internally having a term N such that $\vdash_{\mathsf{CIC}+\mathcal{E}} N : \neg\mathcal{E}$. The latter would have been enough to show the degeneracy of the construction, i.e. every type would be isomorphic to its exceptional translation.

Thanks to our fancily uninhabited type of exceptions, we actually get a non-trivial extension of CIC.

Theorem 7. CCCP *validates Russian constructivism.*

Proof. The proof is actually quite easy. MP is translated by the exceptional translation in $\mathsf{CIC} + \mathcal{E}$ as

$$
\begin{array}{l}
\Pi(\varphi : \mathbb{N}_{\mathcal{E}} \to \mathbb{B}_{\mathcal{E}}). \\
\quad (((\Sigma_{\mathcal{E}}(n : \mathbb{N}_{\mathcal{E}}).\, \varphi\ n =_{\mathcal{E}} \mathtt{true}_{\mathcal{E}}) \to \bot_{\mathcal{E}}) \to \bot_{\mathcal{E}}) \to \\
\quad\quad \Sigma_{\mathcal{E}}(n : \mathbb{N}_{\mathcal{E}}).\, \varphi\ n =_{\mathcal{E}} \mathtt{true}_{\mathcal{E}}.
\end{array}
$$

Recall that in $\mathsf{CIC} + \mathcal{E}$,

$$
\begin{array}{lcl}
\bot_{\mathcal{E}} & \cong & \mathcal{E} \\
M =_{\mathcal{E}} N & \cong & (M = N) + \mathcal{E} \\
\Sigma_{\mathcal{E}}(x : A).\, B & \cong & (\Sigma x : A.\, B) + \mathcal{E}
\end{array}
$$

By intuitionistic symbol pushing, it is thus logically equivalent to prove the simpler statement

$$
\begin{array}{l}
\Pi(\varphi : \mathbb{N}_{\mathcal{E}} \to \mathbb{B}_{\mathcal{E}}). \\
\quad (((\Sigma n : \mathbb{N}_{\mathcal{E}}.\, \varphi\ n = \mathtt{true}_{\mathcal{E}}) \to \mathcal{E}) \to \mathcal{E}) \to \\
\quad\quad (\Sigma n : \mathbb{N}_{\mathcal{E}}.\, \varphi\ n = \mathtt{true}_{\mathcal{E}}) + \mathcal{E}.
\end{array}
$$

The first part of the proof is to get rid of the exceptional encoding on φ. Let us assume for now some $\varphi : \mathbb{N}_{\mathcal{E}} \to \mathbb{B}_{\mathcal{E}}$. Let $\Uparrow_{\mathbb{N}} : \mathbb{N} \to \mathbb{N}_{\mathcal{E}}$ be the injective function that recursively maps every pure constructor to its exceptional counterpart. Remark now we can easily build a function $\hat{\varphi} : \mathbb{N} \to \mathbb{B}$ by pattern-matching as

$$
\begin{array}{lcll}
\hat{\varphi}\ n & := & \mathtt{true} & \text{if } \varphi\ (\Uparrow_{\mathbb{N}} n) \equiv \mathtt{true}_{\mathcal{E}} \\
\hat{\varphi}\ n & := & \mathtt{false} & \text{if } \varphi\ (\Uparrow_{\mathbb{N}} n) \equiv \mathtt{false}_{\mathcal{E}} \\
\hat{\varphi}\ n & := & \mathtt{false} & \text{if } \varphi\ (\Uparrow_{\mathbb{N}} n) \equiv \mathbb{B}_{\varnothing}\ e.
\end{array}
$$

Moreover, we have both

$$
\begin{array}{l}
(\Sigma n : \mathbb{N}_{\mathcal{E}}.\, \varphi\ n = \mathtt{true}_{\mathcal{E}}) \to (\Sigma n : \mathbb{N}.\, \hat{\varphi}\ n = \mathtt{true}) + \mathcal{E} \\
(\Sigma n : \mathbb{N}.\, \hat{\varphi}\ n = \mathtt{true}) \quad\quad \to (\Sigma n : \mathbb{N}_{\mathcal{E}}.\, \varphi\ n = \mathtt{true}_{\mathcal{E}})
\end{array}
$$

the first one being proved by recursively forcing the integer witness, and the second by trivial injection. By combining these results together, $[\![\mathsf{MP}]\!]$ is reducible to

$$
\begin{array}{l}
\Pi(\varphi : \mathbb{N} \to \mathbb{B}). \\
\quad (((\Sigma n : \mathbb{N}.\, \varphi\ n = \mathtt{true}) \to \mathcal{E}) \to \mathcal{E}) \to \\
\quad\quad (\Sigma n : \mathbb{N}.\, \varphi\ n = \mathtt{true}) + \mathcal{E}.
\end{array}
$$

So far we have not used the additional structure on $\mathcal{E}$ and have only reasoned intuitionistically. The second part of the proof is precisely about the use of dynamic prompt to show the above statement.

Let us assume now some $\varphi : \mathbb{N} \to \mathbb{B}$ and a proof $e : ((\Sigma n : \mathbb{N}.\, \varphi\ n = \mathtt{true}) \to \mathcal{E}) \to \mathcal{E}$, we need to show $(\Sigma n : \mathbb{N}.\, \varphi\ n = \mathtt{true}) + \mathcal{E}$. By applying `return` to e we get a proof of

$$\mathtt{local}\ \varphi\ (((\Sigma n : \mathbb{N}.\, \varphi\ n = \mathtt{true}) \to \mathcal{E}) \to \mathcal{E}).$$

Now, by applying the various distribution lemmas from Figure 5, we get a proof of

$$((\Sigma n : \mathbb{N}.\, \varphi\ n = \mathtt{true}) \to \mathtt{local}\ \varphi\ \mathcal{E}) \to \mathtt{local}\ \varphi\ \mathcal{E}.$$

But remember that

$$\mathtt{local}\ \varphi\ \mathcal{E} \leftrightarrow (\Sigma n : \mathbb{N}.\, \varphi\ n = \mathtt{true}) + \mathcal{E}$$

so by writing

$$\begin{array}{rcl} \mathcal{E}' & := & \mathtt{local}\ \varphi\ \mathcal{E} \\ P & := & \Sigma n : \mathbb{N}.\, \varphi\ n = \mathtt{true} \end{array}$$

we only have to prove

$$(\mathcal{E}' \leftrightarrow P + \mathcal{E}) \to ((P \to \mathcal{E}') \to \mathcal{E}') \to P + \mathcal{E}'$$

which is an intuitionistic tautology, regardless of $\mathcal{E}$, $\mathcal{E}'$ and P.

□

11 The Prompt Dialectics

In this section, we justify CIC + $\mathcal{E}$ via a specific instance of the prefascist construction. In contrast with the previous section, CIC + $\mathcal{E}$ will now be the *source* theory, while the target will be 𝔰CIC. Similarly, any syntactic translation will refer to the prefascist model, *not* the exceptional one.

As a minimal departure on the definition from Section 4, we will use a strict preorder rather than a category. That is, `hom` will live in $*_0$ rather than $\square_0$, and we will evocatively write $p \le q$ rather than `hom` $p\ q$. We stress out that this does not affect neither the translation nor the soundness theorems in any way.

Without much surprise, the preorder we will consider is the one from Coquand-Hofmann [15].

Definition 7. We define in 𝔰CIC the preorder $\mathbb{F}$ as

- $\mathbb{F} := \mathbb{N} \to \mathbb{B}$.
- $p \le q := \Pi n : \mathbb{N}.\, q\ n = \mathtt{true} \to p\ n = \mathtt{true}$.

Going down in the preorder means becoming truer. This preorder has a meet semi-lattice structure, which we will put at good use, with

- $\top := \lambda n : \mathbb{N}.\, \mathtt{false}$
- $p \wedge q := \lambda n : \mathbb{N}.\, \mathtt{or}\ (p\ n)\ (q\ n)$.

We will write CIC + $\mathcal{E} := \mathfrak{Psh}_{\mathbb{F}}$. By Theorem 3, it is clear that CIC + $\mathcal{E}$ contains CIC. Remark that by Theorem 4 the existence of the maximal element $\top$ ensures that $\mathfrak{Psh}_{\mathbb{F}}$ enjoys canonicity, and thus the premises of Lemma 9 hold.

Let us now implement in the extensions from Figure 5 inside $\mathfrak{Psh}_{\mathbb{F}}$ in Figure 6. We focus on the computational parts in the paper, the commutation lemmas are proved by mere symbol pushing. We need a bit of boilerplate, made explicit below.

Because we work in a strict preorder, for simplicity we will not explicitly write the corresponding proofs, and write $p \triangleleft q$ for some proof of type $p \le q$ derivable from the current context. We write `True` for the one-constructor $*$ inductive type.

By induction it is easy to define a pair of lift functions $\Uparrow_{\mathbb{N}_0} : \mathbb{N} \to [\![\mathbb{N}]\!]_p$ and $\Uparrow_{\mathbb{N}_\varepsilon} : \Pi(n : \mathbb{N}).\, \{\!|\mathbb{N}|\!\}_p\ (\Uparrow_{\mathbb{N}_0} n)$ that map source constructors to their prefascist counterparts. From it, we can derive a function

$$\chi : (\Pi(n_0 : [\![\mathbb{N}]\!]_p)\ (n_\varepsilon : \{\!|\mathbb{N}|\!\}_p\ n_0).\, \mathbb{B}_0\ p) \to \mathbb{N} \to \mathbb{B}$$

that we use to cast prefascist functions $[\![\mathbb{N} \to \mathbb{B}]\!]_p$ to 𝔰CIC functions $\mathbb{N} \to \mathbb{B}$.

There is a bit of noise, but the implementation corresponds literally to the intuitive description from the previous section. We have to freeify the presentation of `local` by adding a spurious quantification, although this is just a technical detail.

Lemma 10. *The $\mathcal{E}$ type is not inhabited.*

Proof. This would require that for any $p : \mathbb{F}$,

$$[\mathcal{E}]_p.\mathsf{T}\ p\ \mathtt{id}_p \equiv (\Sigma n : \mathbb{N}.\, p\ n = \mathtt{true})$$

is inhabited in 𝔰CIC. But it is obvious that this type is provably empty for $p := \top$, and thus uninhabited by consistency of 𝔰CIC.

□

Note that such an exotic type is typical of presheaf models, as its contents depends on the modal value of the global state.

Theorem 8. *The prompt API is realized in $\mathfrak{Psh}_{\mathbb{F}}$.*

By gathering the results from the previous sections, we readily get the following.

Theorem 9. *There is a syntactic model of* CIC + MP *enjoying normalization, canonicity and decidability of type-checking, as long as* 𝔰CIC *also enjoys those properties.*

12 Related Work

The usefulness of strict propositions in the construction of syntactic models was first observed by Altenkirch [4] more than twenty years ago, although this particular instance did not rely on UIP. We find it hard to believe that it took so long for this feature to make it into mainstream proof assistants. It is undeniable that the HoTT

$$
\begin{aligned}
[\mathcal{E}]^{\Gamma}_{p} &:= \mathtt{Psh}\ (\lambda(q\,\alpha : p).\, \Sigma n : \mathbb{N}.\, q\ n = \mathtt{true})\ (\lambda_.\, \mathtt{True}) \\
\{\mathcal{E}\}^{\Gamma}_{p} &:= \lambda(q\,\alpha : p).\, \mathtt{refl} \\
[\mathtt{local}\ \varphi\ A]^{\Gamma}_{p} &:= \mathtt{Psh}\ (\lambda(q\,\alpha : p).\, (q \wedge \chi\,[\varphi]^{\Gamma}_{q} \triangleleft p) \bullet_{\Gamma} [\![A]\!]^{\Gamma}_{q \wedge \chi\,[\varphi]^{\Gamma}_{q}}) \\
&\qquad (\lambda(x_0 : \Pi(q\,\alpha : p).\, (q \wedge \chi\,[\varphi]^{\Gamma}_{q} \triangleleft p) \bullet_{\Gamma} [\![A]\!]^{\Gamma}_{q \wedge \chi\,[\varphi]^{\Gamma}_{q}}).\, (p \wedge \chi\,[\varphi]^{\Gamma}_{p} \triangleleft p) \bullet_{\Gamma} \{\!\{A\}\!\}^{\Gamma}_{p \wedge \chi\,[\varphi]^{\Gamma}_{p}}\ x_0) \\
\{\mathtt{local}\ \varphi\ A\}^{\Gamma}_{p} &:= \lambda(q\,\alpha : p).\, \mathtt{refl} \\
[\mathtt{return}\ A\ \varphi\ M]^{\Gamma}_{p} &:= \lambda(q\,\alpha : p \wedge \chi\,[\varphi]^{\Gamma}_{p}).\, (q \triangleleft p) \bullet_{\Gamma} [M]^{\Gamma}_{q} \\
\{\mathtt{return}\ A\ \varphi\ M\}^{\Gamma}_{p} &:= \lambda(q\,\alpha : p \wedge \chi\,[\varphi]^{\Gamma}_{p}).\, (q \triangleleft p) \bullet_{\Gamma} \{M\}^{\Gamma}_{q}
\end{aligned}
$$

Figure 6. Dynamic Prompt Implementation

trend triggered a revival of the activity around equality that had gone into dormancy at the beginning of the twenty-first century.

When unfolding the composition of the exceptional and prefascist models proving MP, the resulting syntactic translation is, up to implementation details, a dependently-typed variant of the one given by Coquand-Hofmann [15] for PRA^{ω}. This is not surprising, as they describe their model as a mix between the Friedman's A-translation [20] and Krikpe semantics [19]. The former is an ad-hoc first-order variant of the exceptional model, while the latter is a non-dependent variant of the presheaf model. The main contribution of our paper is thus the extension of Coquand-Hofmann's model to CIC, together with its two-stage decomposition that makes it easier to grasp.

We will argue that Herbelin's implementation of MP using a weak form of delimited continuations [25] is, from the computational point of view, essentially the same as the Coquand-Hofmann model. Herbelin's system features statically bound exceptions, that can be introduced through a `try – with` clause and used as a static jump. In Ilik's presentation [29], this system can be typed with an ordered stack of prompts, where failure is allowed to return at any point in this stack. Thus, the intended meaning of a term of type A in a stack Σ is a sum $A + \Sigma$ where the right case corresponds to an exception. Furthermore, given a stack Ξ extending Σ, it is always possible to lower a value A in Σ to a value A in Ξ, mimicking the presheaf monotonocity. The ability to create a new prompt to prove MP is thus tantamount to the action of moving along the order in the Kripke construction. Said otherwise, the prompt stack indexing follows a presheaf discipline.

This is even more obvious in Herbelin-Ghilezan presentation of call-by-name delimited continuations [26], where types that would correspond to a call-by-push-value thunk are indexed by the prompt stack. The CPS they give makes it explicit that a type A_{Σ} can be lowered to A_{Ξ} by weakening, provided Ξ extends Σ. This suggests a strong relationship between typed delimited continuations and presheaves.

An extremely similar phenomenon occurs in normalization-by-evaluation (NbE) proofs for λ-calculi featuring positive types [1, 3, 14, 18, 30, 33, 43]. While proving NbE of the negative fragment can be performed in the syntactic model of presheaves over contexts, there is a fundamental problem as soon as one considers positive types, which require an involved modification of the model, typically in two different ways. The first one [18, 30] is to add MP to the meta-language, either in direct style or by precompositon with a CPS. The second one consists in extending the interpretation of positive types to include neutral terms in the ambient context [1, 14]. With enough squinting, it becomes apparent that those neutral terms behave like statically bound exceptions, an observation already made by Barral [7]. They are made available each time the context is extended with a bound variable, and raised when a term has not enough information to reduce to a normal form. In particular, at toplevel there are no neutral terms because the context is empty, which implies that the model is consistent. This bears a clear kinship with our model of MP.

Let us also mention that our prompt mechanism is reminiscent of a technique recently used by Rahli and Bickford [42] to implement Brouwer's continuity principle based on the generation of fresh exceptions. Their work suggests new uses for our exceptional presheaf model.

References

[1] Andreas Abel. 2013. *Normalization by Evaluation: Dependent Types and Impredicativity.* Habilitation thesis. Universität München.

[2] Andreas Abel and Thierry Coquand. 2019. Failure of Normalization in Impredicative Type Theory with Proof-Irrelevant Propositional Equality. arXiv:1911.08174 [cs.LO]

[3] Andreas Abel and Christian Sattler. 2019. Normalization by Evaluation for Call-By-Push-Value and Polarized Lambda Calculus. In *Proceedings of the 21st International Symposium on Principles*

and Practice of Programming Languages, PPDP 2019, Porto, Portugal, October 7-9, 2019, Ekaterina Komendantskaya (Ed.). ACM, 3:1–3:12. https://doi.org/10.1145/3354166.3354168

[4] Thorsten Altenkirch. 1999. Extensional Equality in Intensional Type Theory. In *14th Annual IEEE Symposium on Logic in Computer Science, Trento, Italy, July 2-5, 1999*. IEEE Computer Society, 412–420. https://doi.org/10.1109/LICS.1999.782636

[5] Thorsten Altenkirch and Conor Mcbride. 2006. Towards observational type theory.

[6] Jeremy Avigad and Solomon Feferman. 1999. *The Handbook of Proof Theory*. North-Holland, Chapter Gödel's functional ("Dialectica") interpretation, 337–405.

[7] Freiric Barral. 2008. *Decidability for non-standard conversions in typed lambda-calculi*. Ph.D. Dissertation. Ludwig Maximilian University of Munich, Germany. http://edoc.ub.uni-muenchen.de/9761/

[8] Michael Beeson. 1985. *Foundations of Constructive Mathematics: Metamathematical Studies*. Number 6 in Ergebnisse der Mathematik und ihrer Grenzgebiete. Springer, Berlin Heidelberg New York Tokyo.

[9] Jean-Philippe Bernardy, Thierry Coquand, and Guilhem Moulin. 2015. A Presheaf Model of Parametric Type Theory. *Electr. Notes Theor. Comput. Sci.* 319 (2015), 67–82. https://doi.org/10.1016/j.entcs.2015.12.006

[10] Jean-Philippe Bernardy and Marc Lasson. 2011. Realizability and Parametricity in Pure Type Systems. In *Foundations of Software Science and Computational Structures*, Vol. 6604. Saarbrücken, Germany, 108–122. https://doi.org/10.1007/978-3-642-19805-2

[11] Ales Bizjak, Hans Bugge Grathwohl, Ranald Clouston, Rasmus Ejlers Møgelberg, and Lars Birkedal. 2016. Guarded Dependent Type Theory with Coinductive Types. In *Foundations of Software Science and Computation Structures - 19th International Conference, FOSSACS 2016, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2016, Eindhoven, The Netherlands, April 2-8, 2016, Proceedings*. 20–35. https://doi.org/10.1007/978-3-662-49630-5_2

[12] Simon Pierre Boulier. 2018. *Extending type theory with syntactic models. (Etendre la théorie des types à l'aide de modèles syntaxiques)*. Ph.D. Dissertation. Ecole nationale supérieure Mines-Télécom Atlantique Bretagne Pays de la Loire, France. https://tel.archives-ouvertes.fr/tel-02007839

[13] Cyril Cohen, Thierry Coquand, Simon Huber, and Anders Mörtberg. 2017. Cubical Type Theory: A Constructive Interpretation of the Univalence Axiom. *FLAP* 4, 10 (2017), 3127–3170. http://collegepublications.co.uk/ifcolog/?00019

[14] Thierry Coquand. 2019. Canonicity and normalization for dependent type theory. *Theor. Comput. Sci.* 777 (2019), 184–191. https://doi.org/10.1016/j.tcs.2019.01.015

[15] Thierry Coquand and Martin Hofmann. 1999. A new method for establishing conservativity of classical systems over their intuitionistic version. *Mathematical Structures in Computer Science* 9, 4 (1999), 323–333. http://journals.cambridge.org/action/displayAbstract?aid=44835

[16] Thierry Coquand, Simon Huber, and Christian Sattler. 2019. Homotopy Canonicity for Cubical Type Theory. In *4th International Conference on Formal Structures for Computation and Deduction, FSCD 2019, June 24-30, 2019, Dortmund, Germany*. 11:1–11:23. https://doi.org/10.4230/LIPIcs.FSCD.2019.11

[17] Thierry Coquand and Bassel Mannaa. 2016. The Independence of Markov's Principle in Type Theory. In *1st International Conference on Formal Structures for Computation and Deduction, FSCD 2016, June 22-26, 2016, Porto, Portugal*. 17:1–17:18. https://doi.org/10.4230/LIPIcs.FSCD.2016.17

[18] Pierre-Évariste Dagand, Lionel Rieg, and Gabriel Scherer. 2019. Dependent Pearl: Normalization by realizability. *CoRR* abs/1908.09123 (2019). arXiv:1908.09123 http://arxiv.org/abs/1908.09123

[19] Michael Dummett. 1977. *Elements of Intuitionism*. Oxford University Press.

[20] Harvey Friedman. 1978. *Classically and intuitionistically provably recursive functions*. Springer Berlin Heidelberg, Berlin, Heidelberg, 21–27. https://doi.org/10.1007/BFb0103100

[21] Carsten Führmann. 1999. Direct Models for the Computational Lambda Calculus. *Electr. Notes Theor. Comput. Sci.* 20 (1999), 245–292. https://doi.org/10.1016/S1571-0661(04)80078-1

[22] Gaëtan Gilbert. 2019. *A type theory with definitional proof-irrelevance*. Ph.D. Dissertation. University of Nantes.

[23] Gaëtan Gilbert, Jesper Cockx, Matthieu Sozeau, and Nicolas Tabareau. 2019. Definitional proof-irrelevance without K. *PACMPL* 3, POPL (2019), 3:1–3:28. https://doi.org/10.1145/3290316

[24] Daniel Gratzer, Jonathan Sterling, and Lars Birkedal. 2019. Implementing a modal dependent type theory. *PACMPL* 3, ICFP (2019), 107:1–107:29. https://doi.org/10.1145/3341711

[25] Hugo Herbelin. 2010. An Intuitionistic Logic that Proves Markov's Principle. In *Proceedings of the 25th Annual Symposium on Logic in Computer Science, LICS 2010, 11-14 July 2010, Edinburgh, United Kingdom*. 50–56. https://doi.org/10.1109/LICS.2010.49

[26] Hugo Herbelin and Silvia Ghilezan. 2008. An approach to call-by-name delimited continuations. In *Proceedings of the 35th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2008, San Francisco, California, USA, January 7-12, 2008*. 383–394. https://doi.org/10.1145/1328438.1328484

[27] Martin Hofmann. 1997. *Extensional constructs in intensional type theory*. Springer.

[28] Simon Huber. 2019. Canonicity for Cubical Type Theory. *J. Autom. Reasoning* 63, 2 (2019), 173–210. https://doi.org/10.1007/s10817-018-9469-1

[29] Danko Ilik. 2010. *Constructive Completeness Proofs and Delimited Control. (Preuves constructives de complétude et contrôle délimité)*. Ph.D. Dissertation. École Polytechnique, Palaiseau, France. https://tel.archives-ouvertes.fr/tel-00529021

[30] Danko Ilik. 2013. Continuation-passing style models complete for intuitionistic logic. *Ann. Pure Appl. Logic* 164, 6 (2013), 651–662. https://doi.org/10.1016/j.apal.2012.05.003

[31] Guilhem Jaber, Gabriel Lewertowski, Pierre-Marie Pédrot, Matthieu Sozeau, and Nicolas Tabareau. 2016. The Definitional Side of the Forcing. In *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science, LICS '16, New York, NY, USA, July 5-8, 2016*. 367–376. https://doi.org/10.1145/2933575.2935320

[32] Guilhem Jaber, Nicolas Tabareau, and Matthieu Sozeau. 2012. Extending Type Theory with Forcing. In *Proceedings of the 27th Annual IEEE Symposium on Logic in Computer Science, LICS 2012, Dubrovnik, Croatia, June 25-28, 2012*. IEEE Computer Society, 395–404. https://doi.org/10.1109/LICS.2012.49

[33] Ambrus Kaposi, Simon Huber, and Christian Sattler. 2019. Gluing for Type Theory. In *4th International Conference on Formal Structures for Computation and Deduction, FSCD 2019, June 24-30, 2019, Dortmund, Germany*. 25:1–25:19. https://doi.org/10.4230/LIPIcs.FSCD.2019.25

[34] Stephen Cole Kleene. 1945. On the Interpretation of Intuitionistic Number Theory. *J. Symb. Log.* 10, 4 (1945), 109–124. https://doi.org/10.2307/2269016

[35] Ulrich Kohlenbach. 2008. *Applied Proof Theory - Proof Interpretations and their Use in Mathematics*. Springer. https://doi.org/10.1007/978-3-540-77533-1

[36] Georg Kreisel. 1962. On Weak Completeness of Intuitionistic Predicate Logic. *J. Symb. Log.* 27, 2 (1962), 139–158. https://doi.org/10.2307/2964110

[37] Jean-Louis Krivine. 1994. Classical Logic, Storage Operators and Second-Order lambda-Calculus. *Ann. Pure Appl. Logic* 68, 1 (1994), 53–78. https://doi.org/10.1016/0168-0072(94)90047-7

[38] Charles McCarty. 2009. Constructivism in Mathematics. In *Philosophy of mathematics*, Andrew Irvine, Dov M. Gabbay, Paul Thagard, and John Woods (Eds.). North Holland, 311–343.

[39] Alexandre Miquel. 2011. Forcing as a Program Transformation. In *Proceedings of the 26th Annual IEEE Symposium on Logic in Computer Science, LICS 2011, June 21-24, 2011, Toronto, Ontario, Canada*. IEEE Computer Society, 197–206. https://doi.org/10.1109/LICS.2011.47

[40] Pierre-Marie Pédrot and Nicolas Tabareau. 2018. Failure is Not an Option - An Exceptional Type Theory. In *Programming Languages and Systems - 27th European Symposium on Programming, ESOP 2018, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2018, Thessaloniki, Greece, April 14-20, 2018, Proceedings (Lecture Notes in Computer Science)*, Amal Ahmed (Ed.), Vol. 10801. Springer, 245–271. https://doi.org/10.1007/978-3-319-89884-1_9

[41] Pierre-Marie Pédrot and Nicolas Tabareau. 2020. The Fire Triangle. In *Proceedings of the 47st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL '20)*. ACM.

[42] Vincent Rahli and Mark Bickford. 2018. Validating Brouwer's continuity principle for numbers using named exceptions. *Mathematical Structures in Computer Science* 28, 6 (2018), 942–990. https://doi.org/10.1017/S0960129517000172

[43] Jonathan Sterling and Bas Spitters. 2018. Normalization by gluing for free λ-theories.

[44] A.S. Troelstra (Ed.). 1973. *Metamathematical Investigation of Intuitionistic Arithmetic and Analysis*. Springer.

[45] The Univalent Foundations Program. 2013. *Homotopy Type Theory: Univalent Foundations of Mathematics*. https://homotopytypetheory.org/book, Institute for Advanced Study.

[46] Chuangjie Xu. 2016. Sheaf models of type theory in type theory. Unpublished.