Internship report M2
Non-singleton Elimination

Soudant Léo, supervised by **Pierre-Marie Pédrot**, Galinette team at LS2N

2025

General context

To efficiently formalise mathematics we need strong permissive theories. When attempting to mechanising proofs, however, strong computational properties, like typing decidability, are also expected. Studying the computational contents of sheaves could allow to emulate forcing directly in type theory, but also to bring a computational content to the axiom which the theory forces. Thierry Coquand published a few papers on the computational aspect of forcing, Vincent Rahli studied effectful, though undecidable, type theory based on sheaves. Martin Baillon used a similar theory to show a continuity result on the cantor space [1].

Research problem

An element of a topological sheaf may be defined over an open set whenever it is defined on a cover of that open set, so long as they are compatible over the intersections. We hope to be able to do something similar in a type theory, where open sets correspond to proof irrelevant propositions. This requires special attention to the compatibility condition : when a disjunction covers a proposition, it could be possible to eliminate from this disjunction to create a term in any type whenever compatible marginal terms are found.

Your contribution

We have proved using Rocq that a system T variant with a part of the wanted features normalises. Since the proof was done using logical relations, a few other results should be within reach. I also have a model in Rocq where types are interpreted as sheaves, although some other variant probably existed before.

Arguments supporting its validity

The type theory developped by Martin Baillon, which we aim to generalise, proved continuity of all functional $(\mathbf{N} \to \mathbf{B}) \to \mathbf{B}$. We hope to find other similar results at term. Forcing has also been a very fruitful method in term of showing consistency of added principle.

Summary and future works

We will refine the type theory further as well as study other instances similar to the one Martin Baillon studied. We should also make a fork of the logrel Rocq project, and adapt it to show the good properties of any such theory.

Contents

Outline

In section 1 we explain what type theory is and detail the basic type theory we will study. Section 2 present the construction we would like to make appear in type theory, and why we think it is possible and useful. Section 3 present work done on models of type theory, though the important part is precisely the model of sheaf, since its theory is what we ar etrying to build syntactically. Section 4 exposes two sheaf theories, one which comes from previous work and is an important inspiration, and the other which is a prototype of a generalisation. The last section, section 5, uses the recipe of logical relation to make a proof of concept using a weaker theory, which normalises.

1 Type theory

Type theory is a manner of formalising mathematics with a very different design from set theory, and quite a bit closer in structure to a purely logical system like natural deduction. They are also more adapted to mechanising proof than ZFC.

Our meta-theory is alike a type theory, in which

- $\mathfrak{N}$ denotes its natural numbers type
- $\mathfrak{B}$, $\mathfrak{tt}$ and $\mathfrak{ff}$, its boolean type and its two constants
- Π, λ and concatenation for its dependent function type
- Σ, $(\_,\_)$, $\mathfrak{p}_1$, $\mathfrak{p}_2$ its dependent product type, pairs and projections, with η-expansion (w convertible to $(\mathfrak{p}_1 w, \mathfrak{p}_2 w)$).
- Type its universe, with levels left implicit.

1.1 MLTT

We present a variant of MLTT (Martin-Löf Type Theory), a very standard type theory.

This will give hints as to how our meta-theory work, give a precise reference point for when talking about MLTT, and most importantly, since one of our end goal is to provide an extension of MLTT, describe the first rules of the extension.

This variant will feature dependent function types, universes, and, as positive types, natural numbers and an empty type.

We give ourself a poset of universe levels ℓ. It may be thought of as $\mathfrak{N}$, however, for any proof in Rocq, we will need to pick out a finite poset, most likely "$0 < 1$".

We add a type $\mathbf{N}$ of natural numbers and an empty type $\bot$ as positive types, instead of the heavy syntax of a general inductive type. They both implement large elimination, which means, in particular, that their induction principle can be used to construct functions returning types, like $\lambda A n, A^n$, in presence of a binary product.

We ignore any issue that could stem from bound variables, and assume different binders use variables different from one another and from free variables.

With terms :

$$M, N ::= x \mid \lambda x.M \mid MN \mid 0 \mid S \mid \mathbf{N}_{\text{rec}} \mid \bot_{\text{rec}} \mid \mathbf{N} \mid \bot \mid \Pi x : A.B \mid \square_\ell$$

Contexts :

$$\Gamma ::= \Gamma, x : A \mid \cdot$$

We finally give rules for conversion and well formation of contexts. Typing is given by self-conversion, a term M is of type A in context Γ whenever $\Gamma \vdash M \equiv M : A$ is derivable.

$$\textsc{Wf-Empty}\frac{}{\cdot \vdash \mathsf{well\text{-}formed}} \qquad \textsc{Wf-Ext}\frac{\Gamma \vdash A \equiv A \qquad \Gamma \vdash \mathsf{well\text{-}formed}}{\Gamma, x : A \vdash \mathsf{well\text{-}formed}}$$

$$\textsc{Int-Typ}\frac{\Gamma \vdash \mathsf{well\text{-}formed}}{\Gamma \vdash \mathbf{N} \equiv \mathbf{N} : \square_\ell} \qquad \textsc{Emp-Typ}\frac{\Gamma \vdash \mathsf{well\text{-}formed}}{\Gamma \vdash \bot \equiv \bot : \square_\ell}$$

$$\textsc{Fun-Typ}\frac{\Gamma \vdash A \equiv A' : \square_\ell \qquad \Gamma, x : A \vdash B \equiv B' : \square_{\ell'} \qquad \ell \leq \ell'' \qquad \ell' \leq \ell''}{\Gamma \vdash \Pi x : A, B \equiv \Pi x : A', B' : \square_{\ell''}} \qquad \textsc{Typ-Typ}\frac{\Gamma \vdash \mathsf{well\text{-}formed} \qquad \ell < \ell'}{\Gamma \vdash \square_\ell \equiv \square_\ell : \square_{\ell'}}$$

$$\textsc{Fun-Intro}\frac{\Gamma, x : A \vdash M \equiv M' : B \qquad \Gamma \vdash A \equiv A : \square_\ell}{\Gamma \vdash \lambda x.M \equiv \lambda x.M' : \Pi x : A, B} \qquad \textsc{Fun-Elim}\frac{\Gamma \vdash M \equiv M' : \Pi x : A, B \qquad \Gamma \vdash N \equiv N' : A}{\Gamma \vdash MN \equiv M'N' : B(N/x)}$$

$$\textsc{Axiom}\frac{\Gamma \vdash \mathsf{well\text{-}formed} \qquad x : A \in \Gamma}{\Gamma \vdash x \equiv x : A} \qquad \textsc{Beta}\frac{\Gamma, x : A \vdash M \equiv M' : B \qquad \Gamma \vdash N \equiv N' : A}{\Gamma \vdash (\lambda x.M)N \equiv M'(N'/x) : B(N/x)}$$

$$\textsc{Int-Zero}\frac{\Gamma \vdash \mathsf{well\text{-}formed}}{\Gamma \vdash 0 \equiv 0 : \mathbf{N}} \qquad \textsc{Int-Succ}\frac{\Gamma \vdash \mathsf{well\text{-}formed}}{\Gamma \vdash S \equiv S : \mathbf{N} \to \mathbf{N}}$$

$$\textsc{Int-Rec}\frac{\Gamma \vdash \mathsf{well\text{-}formed}}{\Gamma \vdash \mathbf{N}_{\mathrm{rec}} \equiv \mathbf{N}_{\mathrm{rec}} : \Pi A : \mathbf{N} \to \square_\ell, A0 \to (\Pi n : \mathbf{N}, An \to A(Sn)) \to \Pi n : \mathbf{N}, An}$$

$$\textsc{Int-Rec-Zero}\frac{\begin{array}{c}\Gamma \vdash A \equiv A : \mathbf{N} \to \square_\ell \qquad \Gamma \vdash N_0 \equiv N_0' \equiv A0 \\ \Gamma \vdash N_S \equiv N_S : \Pi n : \mathbf{N}, An \to A(Sn)\end{array}}{\Gamma \vdash \mathbf{N}_{\mathrm{rec}} A N_0 N_S 0 \equiv N_0' : A0}$$

$$\textsc{Int-Rec-Succ}\frac{\begin{array}{c}\Gamma \vdash A \equiv A' : \mathbf{N} \to \square_\ell \qquad \Gamma \vdash N_0 \equiv N_0' \equiv A0 \\ \Gamma \vdash N_S \equiv N_S' : \Pi n : \mathbf{N}, An \to A(Sn) \qquad \Gamma \vdash N \equiv N' : \mathbf{N}\end{array}}{\Gamma \vdash \mathbf{N}_{\mathrm{rec}} A N_0 N_S (SN) \equiv N_S' N' (\mathbf{N}_{\mathrm{rec}} A' N_0' N_S' N') : A(SN)}$$

$$\textsc{Emp-Rec}\frac{\Gamma \vdash \mathsf{well\text{-}formed}}{\Gamma \vdash \bot_{\mathrm{rec}} \equiv \bot_{\mathrm{rec}} : \Pi A : \bot \to \square_\ell, \Pi e : \bot, Ae}$$

$$\textsc{Sym}\frac{\Gamma \vdash M \equiv M' : A}{\Gamma \vdash M' \equiv M : A} \qquad \textsc{Trans}\frac{\Gamma \vdash M \equiv M' : A \qquad \Gamma \vdash M' \equiv M'' : A}{\Gamma \vdash M \equiv M'' : A}$$

$$\textsc{Conv}\frac{\Gamma \vdash M \equiv M' : A \qquad \Gamma \vdash A \equiv A' : \square_\ell}{\Gamma M \equiv M' : A'}$$

However, it is useful to consider the extension with a type of booleans $\mathbf{B}$, with new terms :

$$M, N ::= \ldots \mid \mathbf{B} \mid \mathbf{B}_{\mathrm{rec}} \mid \mathsf{tt} \mid \mathsf{ff}$$

And conversion rules

$$\textsc{Bool-True}\frac{\Gamma \vdash \mathsf{well\text{-}formed}}{\Gamma \vdash \mathsf{tt} \equiv \mathsf{tt} : \mathbf{B}} \qquad \textsc{Bool-False}\frac{\Gamma \vdash \mathsf{well\text{-}formed}}{\Gamma \vdash \mathsf{ff} \equiv \mathsf{ff} : \mathbf{B}}$$

$$\textsc{Bool-Rec}\frac{\Gamma \vdash \mathsf{well\text{-}formed}}{\Gamma \vdash \mathbf{B}_{\mathrm{rec}} \equiv \mathbf{B}_{\mathrm{rec}} : \Pi A : \mathbf{B} \to \square_\ell, A\,\mathsf{tt} \to A\,\mathsf{ff} \to \Pi b : \mathbf{B}, Ab}$$

$$\textsc{Bool-Rec-True}\frac{\begin{array}{c}\Gamma \vdash A \equiv A : \mathbf{B} \to \square_\ell \\ \Gamma \vdash M_{\mathsf{tt}} \equiv M_{\mathsf{tt}}' \equiv A\mathsf{tt} \qquad \Gamma \vdash M_{\mathsf{ff}} \equiv M_{\mathsf{ff}} : A\,\mathsf{ff}\end{array}}{\Gamma \vdash \mathbf{B}_{\mathrm{rec}} A M_{\mathsf{tt}} M_{\mathsf{ff}}\,\mathsf{tt} \equiv M_{\mathsf{tt}}' : A\,\mathsf{tt}}$$

$$\textsc{Bool-Rec-False}\frac{\begin{array}{c}\Gamma \vdash A \equiv A : \mathbf{B} \to \square_\ell \\ \Gamma \vdash M_{\mathsf{tt}} \equiv M_{\mathsf{tt}} \equiv A\,\mathsf{tt} \qquad \Gamma \vdash M_{\mathsf{ff}} \equiv M_{\mathsf{ff}}' : A\,\mathsf{ff}\end{array}}{\Gamma \vdash \mathbf{B}_{\mathrm{rec}} A M_{\mathsf{tt}} M_{\mathsf{ff}}\,\mathsf{ff} \equiv M_{\mathsf{ff}}' : A\,\mathsf{ff}}$$

1.2 System T

To verify that the features we would add made some amount of sense, we studied them first in system T. System T has no universes, no dependent function types, and keeps types and terms well-separated. This makes applying the proof technique we used, logical relations, much simpler. We use the following variant :

With types :

$$A ::= A \to A \mid \mathbf{N} \mid \bot$$

Terms :

$$M, N ::= x \mid \lambda x.M \mid MN \mid 0 \mid S \mid \mathbf{N}_{\mathrm{rec}} \mid \bot_{\mathrm{rec}}$$

Contexts :

$$\Gamma ::= \Gamma, x : A \mid \cdot$$

And conversion rules :

$$\textsc{Fun-Intro}\frac{\Gamma, x : A \vdash M \equiv M' : B}{\Gamma \vdash \lambda x.M \equiv \lambda x.M' : A \to B} \qquad \textsc{Fun-Elim}\frac{\Gamma \vdash M \equiv M' : A \to B \quad \Gamma \vdash N \equiv N' : A}{\Gamma \vdash MN \equiv M'N' : B}$$

$$\textsc{Axiom}\frac{x : A \in \Gamma}{\Gamma \vdash x \equiv x : A} \qquad \textsc{Beta}\frac{\Gamma, x : A \vdash M \equiv M' : B \quad \Gamma \vdash N \equiv N' : A}{\Gamma \vdash (\lambda x.M)N \equiv M' : B}$$

$$\textsc{Int-Zero}\frac{}{\Gamma \vdash 0 \equiv 0 : \mathbf{N}} \qquad \textsc{Int-Succ}\frac{}{\Gamma \vdash S \equiv S : \mathbf{N} \to \mathbf{N}}$$

$$\textsc{Int-Rec}\frac{}{\Gamma \vdash \mathbf{N}_{\text{rec}} \equiv \mathbf{N}_{\text{rec}} : A \to (\mathbf{N} \to A \to A) \to \mathbf{N} \to A}$$

$$\textsc{Int-Rec-Zero}\frac{\begin{array}{c}\Gamma \vdash N_0 \equiv N_0' \equiv A \\ \Gamma \vdash N_S \equiv N_S : \mathbf{N} \to A \to A\end{array}}{\Gamma \vdash \mathbf{N}_{\text{rec}} N_0 N_S 0 \equiv N_0' : A}$$

$$\textsc{Int-Rec-Succ}\frac{\begin{array}{c}\Gamma \vdash N_0 \equiv N_0' \equiv A \\ \Gamma \vdash N_S \equiv N_S' : \mathbf{N} \to A \to A \quad \Gamma \vdash N \equiv N' : \mathbf{N}\end{array}}{\Gamma \vdash \mathbf{N}_{\text{rec}} N_0 N_S (SN) \equiv N_S' N' (\mathbf{N}_{\text{rec}} N_0' N_S' N') : A}$$

$$\textsc{Emp-Rec}\frac{}{\Gamma \vdash \bot_{\text{rec}} \equiv \bot_{\text{rec}} : \bot \to A}$$

$$\textsc{Sym}\frac{\Gamma \vdash M \equiv M' : A}{\Gamma \vdash M' \equiv M : A} \qquad \textsc{Trans}\frac{\Gamma \vdash M \equiv M' : A \quad \Gamma \vdash M' \equiv M'' : A}{\Gamma \vdash M \equiv M'' : A}$$

Typing rules are encoded as self-conversion, and difficulties surrounding variables are ignored.

2 Sheaves

Since we refer to some of the more historical notion of sheaves and since topoi are a set theorist approach of type theory, the meta-theory and language of subsections 2.1, 2.2 and 2.3 are more alike set theory. Since the definition are presented by way of example, this should not be an issue.

2.1 Sheaves on topological spaces

While sheaves on topological space are the hardest form of sheaves to connect to the (I, O)-sheaves we study (defined in 2.5), we find that they offer a clearer insight on how sheaves are going to be used than sheaves on a grothendieck topology do, and as such present the first and not the second in detail. Treating proof irrelevant proposition as subset of a space is a notion we like to keep in mind.

We fix a topological space X, in this case presheaves on that space are presheaves on the category underlying its poset of open subsets. More precisely :

Definition 2.1 (Presheaf on a topology)**.** *A* presheaf P *is given by*

- *For every open* $U \in \mathcal{O}(X)$*, a set* $P(U)$ *of* sections
- *For every open* U*, section* $s \in P(U)$*, and open subset* $V \subseteq U$*, a* restriction $s|_V \in P(V)$
- *If* $s \in P(U)$*,* $s|_U = s$
- *If* $s \in P(U)$*,* $W \subseteq V \subseteq U$*,* $(s|_V)|_W = s|_W$

The $\lambda(s \in P(U)), s|_V : P(U) \to P(V)$ function is the function $P(f)$ necessary to construct the functor $P : \mathcal{O}(X)^{op} \to \mathbf{Set}$, justifying the use of the same term « presheaf » here and in the categorical context.

In the context of topologies, sheaves are simply refinements of presheaves :

Definition 2.2 (Sheaf on a topology)**.** *A* sheaf $\mathcal{F}$ *is a presheaf such that given any open* U *and family of open* $(U_i)_i$ *such that* $U = \bigcup_i U_i$*, there is a bijection between :*

- *Sections* s *of* U
- Compatible *families* $(s_i \in \mathcal{F}(U_i))_i$ *of sections where* $s_i|_{U_i \cap U_j} = s_j|_{U_i \cap U_j}$

where the forward direction is given by $s \mapsto (s|_{U_i})_i$*.*

Using natural transformations as arrows, both sheaves and presheaves over a topology form a category, respectively $\widehat{\mathcal{O}(X)}$ and $\mathbf{Sh}(X)$.

Lemma 2.3 (Sheafification)**.** *The inclusion functor of sheaves into presheaves* $\mathbf{Sh}(\mathbf{X}) \to \widehat{\mathcal{O}(X)}$ *has a left* sheafification *adjoint.*

This adjoint can be constructed using the plus construction $P \mapsto P^+$, which gives the presheaf of compatible families up to refinement. It always yields a sheaf when applied twice, and stabilises at this point [2].

Since the definitions only access the open subsets and not the points of the space, they can be extended to *locales*, that is, posets with finite meets and arbitrary joins satisfying the infinite distributive law. As far as the reader is concerned, this only means $\vee, \wedge$ and $\leq$ will be used instead $\cup, \cap$ and $\subseteq$. We will stick with calling their elements opens, however.

2.2 Kripke and Beth semantics

As a first taste of the logical content of presheaves and sheaves, we present Kripke and Beth models for propositional intuitionistic logic.

To construct a Kripke model, we consider a locale (or in general, a small category). We consider a set of propositional variables and to P one of them, associate a downward closed set of opens, or *sieve* S_P.

We can then define $U \Vdash \phi$, meaning U forces ϕ, by induction on the structure of ϕ.

- $U \Vdash P$ iff for all $V \leq U$, $V \in S_P$
- $U \Vdash \phi \wedge \psi$ iff for all $V \leq U$, $V \Vdash \phi$ and $V \Vdash \psi$
- $U \Vdash \phi \Rightarrow \psi$ iff for all $V \leq U$, if $V \Vdash \phi$ then $V \Vdash \psi$
- $U \Vdash \phi \vee \psi$ iff for all $V \leq U$, $V \Vdash \phi$ or $V \Vdash \psi$

We can then prove that for all ϕ, $U \Vdash \phi$ iff for all $V \leq U$, $V \Vdash \phi$, making all cases but $\Rightarrow$ become degenerate. For example, $U \Vdash \phi \wedge \psi$ iff for all $V \leq U$, $V \Vdash \phi$ and $V \Vdash \psi$ iff $U \Vdash \phi$ and $U \Vdash \psi$. This recovers the more common definition.

We note that by using the poset of contexts, Kripke models done with small categories are (only classicaly) complete for propositional intuitionistic logic.

Sheaves themselves yield a more general kind of model, Beth models.

For every propositional variable P, S_P must now be stable under union, since we introduce *covers* of an open, and says $(U_i)_i$ covers U whenever $U = \bigvee U_i$. The forcing statement $U \Vdash \phi$ is now defined as follows :

- $U \Vdash P$ iff for all $V \leq U$ there exists $(V_i)_i$ covering V, such that for all i, $V_i \in S_P$
- $U \Vdash \phi \wedge \psi$ iff for all $V \leq U$ there exists $(V_i)_i$ covering V, such that for all i, $V_i \Vdash \phi$ and $V_i \Vdash \psi$.
- $U \Vdash \phi \vee \psi$ iff for all $V \leq U$ there exists $(V_i)_i$ covering V, such that for all i, $V_i \Vdash \phi$ or $V_i \Vdash \psi$.
- $U \Vdash \phi \Rightarrow \psi$ iff for all $V \leq U$ there exists $(V_i)_i$ covering V, such that for all i, if $V_i \Vdash \phi$ then $U_i \Vdash \psi$.

Now it can also be proven that $U \Vdash \phi$ iff there exists $(U_i)_i$ covering U, such that for all i, $U_i \Vdash \phi$, and the above degenerates into :

- $U \Vdash P$ iff $U \in S_P$
- $U \Vdash \phi \wedge \psi$ iff $U \Vdash \phi$ and $U \Vdash \psi$.
- $U \Vdash \phi \vee \psi$ iff there exists $(U_i)_i$ with $U = \bigvee_i U_i$, such that for all i, $U_i \Vdash \phi$ or $U_i \Vdash \psi$.

- $U \Vdash \phi \Rightarrow \psi$ iff for all $V \leq U$, if $V \Vdash \phi$ then $V \Vdash \psi$.

They are other variant of Beth semantics, relying on different notion of *cover* in small categories.

Both Kripke and Beth model can be extended to interpret quantifiers, and even further, since some of them, depending on their base category and notion of cover, even interpret set theory entirely. Notably the completely degenerate 1 object category, which recovers the standard model (interpreting sets as sets...), and models which negates the axiom of choice or the continuum hypothesis.

2.3 Topoi and sheaves in topoi

Those extended models emerge because presheaves and sheaves form topoi. Looking at topoi allows separation of the notion of sheaves from that of presheaves.

Proofs, results and details for this section can often be found in *Sheaves in geometry and logic: A first introduction to topos theory* by Saunders Maclane and Ieke Moerdijk [2].

Definition 2.4 (Subobject)**.** *In a category, a* subobject *of X is an equivalence class of monomorphism $m : A \rightarrowtail X$, where the equivalence comes from the preorder where $m : A \rightarrowtail X$ is smaller than $m' : A' \rightarrowtail X$ when there is a map $f : A \to A'$ with $m' \circ f = m$.*

We deduce a presheaf **Sub** where $\mathbf{Sub}(X)$ is the the set of subobjects of X, and $\mathbf{Sub}(f) : \mathbf{Sub}(Y) \to \mathbf{Sub}(X)$ sends $m : A \to Y$ to its pullback by $f : X \to Y$.

Definition 2.5 (Topos)**.** *A* topos *is a cartesian closed category with all finite limits and a suboject* classifier Ω *and an isomorphism $\mathbf{Sub}(X) \cong \mathbf{Hom}(X, \Omega)$ natural in X.*

We note that topoi also have finite colimits.

Topoi form a class of models of some kind of type theory. However, this type theory has equality reflection, propositional extensionnality, strict propositions and the axiom of unique choice. This properties form a very strong, but also undecidable, system. In particular they are no known type theory with both unique choice and strict propositions which preserves canonicity and decidability of type-checking. They also have no universe in general, which, on the other hand, makes them very weak.

To give a few examples of topoi, **Set** is a topos, and given a topos $\mathcal{E}$, $\mathcal{E}/X$, the category of maps with codomain X and commuting triangles, as well as $\mathcal{E}^{\mathbf{C}^{op}}$, the category of contravariant functors from a small category $\mathbf{C}$ and natural transformation, are all topoi. In particular categories of presheaves are topoi, and their internal language are extensions of Kripke models. The categories of sheaves over topological spaces also form topoi, and their internal language are extensions of Beth models.

The subobject classifier Ω is equipped with an internal meet-semilattice structure inherited from the meet-semilattice structure on each $\mathbf{Sub}(X)$, which is natural in X.

Definition 2.6 (Lawvere-Tierney topology)**.** *A* Lawvere-Tierney topology *is a left exact idempotent monad j on the internal meet-semilattice on Ω.*

- $id_\Omega \leq j$,
- $j \circ j \leq j$
- $j \circ \wedge = \wedge \circ j \times j$

From a topology j we extract a closure operator J_X of $\mathbf{Sub}(X)$ for any X.

Definition 2.7 (Dense subobject)**.** *A suboject U of X is dense if $J_X U = X$*

A topology can be lifted to a left exact idempotent monad on the entirety of the topos, the sheafification monad.

Definition 2.8 (j-Sheaf in topos)**.** *An object F is a* j-sheaf *in a topos if for any dense subobject U of any object X, the morphism $\mathbf{Hom}(X, F) \to \mathbf{Hom}(U, F)$ obtained by precomposition is an isomorphism.*

Just like with topological sheaves, the full sub-category of j-Sheaves in a topos $\mathcal{E}$ forms a topos $\mathbf{Sh}_j(\mathcal{E})$. Note that this definition of sheaves makes no reference to presheaves, however, they do still specialise to topological presheaves.

In the topos of presheaf over X, elements of Ω are downward closed sets of open. To recover topolgical sheaves, the j operator must send such set to the set of sub-opens of their union *i.e* sub-opens which can be covered by them. A subobject A of P is then dense if for every open U and every section s of $P(U)$, there is a family of opens and sections $s_i \in U_i$ which can be glued back into s, *i.e.* $U = \bigvee U_i$ and $s_i = s|_{U_i}$. Although, to recover the earlier sheaf definition, one can simply look at presheaves P where $P(U)$ has at most one element, since morphisms out of these presheaves directly yield compatible families.

And just like with topological presheaves, we can define sheafification.

Lemma 2.9. *The inclusion functor from j-sheaves in a topos $\mathcal{E}$ to the topos $\mathcal{E}$ has a left* sheafification *adjoint.*

As announced, it is an idempotentt monad.

2.4 Geometric formulas

In the litterature around sheaves and forcing, we find the following definitions.

Not-a-Definition 2.10. *A* geometric formula *is a formula built from $\exists, \wedge, \bigvee$, and atomic formulas.*

Not-a-Definition 2.11. *A* geometric implication *is a formula of shape $\forall \overrightarrow{x}, \phi(\overrightarrow{x}) \to \psi(\overrightarrow{x})$, where ϕ and ψ are geometric.*

Not-a-Definition 2.12. *A* geometric theory *is a set of geometric implication.*

We rather use the following definition

Definition 2.13. *A* geometric formula *is a formula of the form*

$$\bigwedge_{i:I} \left(\forall \overrightarrow{x}, \bigwedge_{j:J_i} O_{i,j}(\overrightarrow{x}) \to \bigvee_{k:K_i} \exists \overrightarrow{y}, \bigwedge_{l:L_{i,k}} Q_{i,k,l}(\overrightarrow{x}, \overrightarrow{y}) \right)$$

Where $O_{i,j}$ and $Q_{i,k,l}$ are atomic formulas, and both every J_i and every $L_{i,k}$ are finite.

Every « geometric theory » can be written as a single geometric formula.

We'd like to note that they define covers as they can be read as saying that some intersection of atoms can be covered by a family of intersection of atoms, and that they may be integrated in a deduction system as :

$$\textsc{Geo}\text{-}i \frac{Q_{i,1,1}(\overrightarrow{x}, \overrightarrow{t_1}) \dots Q_{i,1,m_{i,1}}(\overrightarrow{x}, \overrightarrow{t_1}), \Gamma \vdash \mathcal{J} \ \cdots \ Q_{i,p_i,1}(\overrightarrow{x}, \overrightarrow{t_{p_i}}) \dots Q_{i,p_i,m_{i,p_i}}(\overrightarrow{x}, \overrightarrow{t_{p_i}}), \Gamma \vdash \mathcal{J}}{O_{i,1}(\overrightarrow{x}), \dots O_{i,n_i}(\overrightarrow{x}), \Gamma \vdash \mathcal{J}}$$

2.5 Sheaves in type theory

Consider a type theory with a notion of proof irrelevant propositions Prop, *e.g.* book-HoTT with mere propositions, or Rocq with SProp.

In this case, a Lawvere-Tierney topology may be similarily defined, as a monad:

- $\mathsf{J} : \mathsf{Prop} \to \mathsf{Prop}$
- $\eta : \Pi(P : \mathsf{Prop}).P \to \mathsf{J}\ P$
- $\mathsf{bind} : \forall(PQ : \mathsf{Prop}).\mathsf{J}\ P \to (P \to \mathsf{J}\ Q) \to \mathsf{J}\ Q$

Then a sheaf is just a type T with

- A map $\mathsf{ask}_T : \Pi(P : \mathsf{Prop}).\ \mathsf{J}\ P \to (P \to T) \to T$
- A coherence $\varepsilon_T : \Pi(P : \mathsf{Prop})\ (j : \mathsf{J}\ P)\ (x : T).\ \mathsf{ask}_T\ \ P\ j\ (\lambda p : P.x) = x$

In this context, sheafification doesn't exist in general, but if the meta-theory admits quotient inductive types, it can then be defined as follow :

$$\begin{aligned}&\mathsf{Inductive}\ \mathcal{S}_{\mathsf{J}}\ T : \mathsf{Type} :=\\ &\mid \mathsf{ret} : T \to \mathcal{S}_{\mathsf{J}}\ T\\ &\mid \mathsf{ask} : \Pi(P : \mathsf{Prop}).\mathsf{J}\ P \to (P \to \mathcal{S}_{\mathsf{J}}\ T) \to \mathcal{S}_{\mathsf{J}}\ T\\ &\mid \varepsilon : \Pi(P : \mathsf{Prop})\ (j : \mathsf{J}\ P)\ (x : \mathcal{S}_{\mathsf{J}}\ T).\,\mathsf{ask}\ \ P\ j\ (\lambda p : P.x) = x\end{aligned}$$

The induction principle is then about as powerful as an adjunction could be, as such we will chose not to define the amount of category theory necessary to formulate the later.

We note that by taking $I := \Sigma(P : \mathsf{Prop}).\mathsf{J}\ P$ and $O := \lambda(P, j).P$, the definition of a sheaf becomes a special case of the following :

Definition 2.14. *A* sheaf *or* (I, O)-sheaf*, is given by*

- *A type* T
- *A map* $\mathsf{ask}_T : \Pi(i : I), (O\ i \to T) \to T$
- *A coherence map* $\varepsilon_T : \Pi(i : I)\ (x : T), \mathsf{ask}_T\ \ i\ (\lambda o : O\ i.x) = x$

Where I is a type and O a family of propositions.

(I, O)-Sheafification can be defined similarily. This definition is marginally simpler, and make sheaves appear as quotient dialogue trees, hence why we will henceforth consider (I, O)-*sheaves* instead of J-*sheaves*.

It is also possible to see geometric formulas as being of the form $\Pi i : I, Oi$, after hefty uncurrying. As such, we will use them to describe our sheaves. However, not all sheaves can be described this way, in particular $\neg\neg$ is a Lawvere-Tierney topology which has no satisfying geometric formula.

3 Models

3.1 Categories with family

A common mean of constructing models of type theory is through categories with families.

They are constituted of :

- A category of contexts and substitutions :

$$\begin{aligned}\mathsf{Ctx} :&\ \mathsf{Type}\\ \mathsf{Sub} :&\ \mathsf{Ctx} \to \mathsf{Ctx} \to \mathsf{Type}\\ \mathsf{Id} :&\ \Pi(\Gamma : \mathsf{Ctx}), \mathsf{Sub}\ \Gamma\ \Gamma\\ &\ldots\end{aligned}$$

- For every context a type of inner types, and for every context and inner type in that context, a type of terms, both accompanied by the action of the substitution upon those, and the corresponding rules.

$$\begin{aligned}\mathsf{Typ} :&\ \mathsf{Ctx} \to \mathsf{Type}\\ \mathsf{Trm} :&\ \Pi(\Gamma : \mathsf{Ctx}), \mathsf{Typ}\,\Gamma \to \mathsf{Type}\\ A[\sigma] :&\ \mathsf{Typ}\ \ \Gamma \text{ whenever } A : \mathsf{Typ}\,\Delta \text{ and } \sigma : \mathsf{Sub}\ \Gamma\ \Delta\\ t[\sigma] :&\ \mathsf{Trm}\ \ \Gamma\ A[\sigma] \text{ whenever } t : \mathsf{Trm}\ \ \Delta\ A \text{ and } \sigma : \mathsf{Sub}\ \Gamma\ \Delta\\ \_ :&\ A[\mathsf{Id}] = A\\ &\ldots\end{aligned}$$

- A way to append a type to a context, substitution weakening, lifting and extension.

$$\begin{aligned}\Gamma; A :&\ \mathsf{Ctx} \text{ whenever } A : \mathsf{Typ}\,\Gamma\\ &\ldots\end{aligned}$$

They may be complemented by :

- Dependent function inner types, with their constructor, eliminator, and their substitution laws.

$$\begin{aligned}
\mathsf{Pi}\ A\ B :&\ \mathsf{Typ}\,\Gamma \text{ whenever } A : \mathsf{Typ}\,\Gamma \text{ and } B : \mathsf{Typ}(\Gamma; A)\\
\mathsf{Abs}\ t :&\ \mathsf{Trm}\ \Gamma\ (\mathsf{Pi}\ A\ B) \text{ whenever } t : \mathsf{Trm}\ (\Gamma; A)\ B\\
\mathsf{App}\ t\ u :&\ \mathsf{Trm}\ \Gamma\ B[u] \text{ whenever } t : \mathsf{Trm}\ \Gamma\ (\mathsf{Pi}\ A\ B) \text{ and } u : \mathsf{Trm}\ (\Gamma; A)\ B\\
&\ldots
\end{aligned}$$

- A universe inner type, and its element function and their substitution laws.

$$\begin{aligned}
\mathsf{U} :&\ \mathsf{Typ}\,\Gamma\\
\mathsf{El[A]} :&\ \mathsf{Typ}\,\Gamma \text{ whenever } A : \mathsf{Trm}\ \Gamma\ \mathsf{U}\\
&\ldots
\end{aligned}$$

  In their presence, some things may be presented as elements of $\mathsf{Trm}\ \Gamma\ \mathsf{U}$ instead of $\mathsf{Typ}\,\Gamma$.

- Some positive types, like a boolean type, its eliminator and constructors, and their substitution laws.

$$\begin{aligned}
\mathbf{B} :&\ \mathsf{Typ}\,\Gamma\\
\mathsf{tt} :&\ \mathsf{Trm}\ \Gamma\ \mathbf{B}\\
\mathsf{ff} :&\ \mathsf{Trm}\ \Gamma\ \mathbf{B}\\
\mathbf{B}_{\mathrm{rec}}\ P\ p_{\mathsf{tt}}\ p_{\mathsf{ff}} :&\ \mathsf{Trm}\ \Gamma\ (\mathsf{Pi}\ \mathbf{B}\ P) \text{ whenever } p_{\mathsf{tt}} : \mathsf{Trm}\ \Gamma\ P[\mathsf{tt}] \text{ and } p_{\mathsf{ff}} : \mathsf{Trm}\ \Gamma\ P[\mathsf{ff}]\\
&\ldots
\end{aligned}$$

- Dependent pair inner types, eliminators, constructors, and laws.

$$\begin{aligned}
\mathsf{Sig}\ A\ B :&\ \mathsf{Typ}\,\Gamma \text{ whenever } A : \mathsf{Typ}\,\Gamma \text{ and } B : \mathsf{Typ}(\Gamma; A)\\
(a, b) :&\ \mathsf{Trm}\ \Gamma\ (\mathsf{Sig}\ A\ B) \text{ whenever } a : \mathsf{Trm}\ \Gamma\ A \text{ and } b : \mathsf{Trm}\ \Gamma\ B[a]\\
\mathsf{Sig}_{\mathrm{rec}}\ P\ p :&\ \mathsf{Trm}\ \Gamma\ (\mathsf{Pi}\ (\mathsf{Sig}\ A\ B)\ P) \text{ whenever } p : \mathsf{Trm}\ \Gamma\ P[(a, b)]\\
&\ldots
\end{aligned}$$

A significant part of my internship was dedicated to contructing models of type theory in Rocq, using categories with families.

The most notable being :

1. A model using dialogue trees.
2. An exceptional model.
3. A model using presheaves.
4. A model using (I, O)-sheaves, which requires univalence, and quotient inductive types to model positive types.

None of them completely implement all the feature listed above, but only the model using presheaves has no dependent function type.

Rocq served as both the meta-theory and the target.

3.2 Dialogue trees

We proceed with introducing these models and their associated structure, starting with dialogue trees.

Given a $I : \mathsf{Type}$ and $O : I \to \mathsf{Type}$, an (I, O)-dialogue tree, is a type A together with a map $\Pi i : I, (O\ i \to A) \to A$. More important, maybe, are how free dialogue tree are defined :

$$\begin{aligned}
&\mathsf{Inductive}\ \mathcal{D}_{I,O}\ T : \mathsf{Type} :=\\
&\mid \mathsf{ret} : T \to \mathcal{D}_{I,O}\ T\\
&\mid \mathsf{ask} : \Pi(i : I), (O\ i \to \mathcal{D}_{I,O}\ T) \to \mathcal{D}_{I,O}\ T
\end{aligned}$$

They encode a fairly large family of monadsThey are quite similar to sheaves, however, unlike sheaves, a model interpreting types as dialogue trees do not require no univalence nor HITs, but cannot implement booleans (or in general, inductive types) as usual.

Instead, they implement them in the way of Baclofen TT, meaning that, using booleans as an example, the eliminator for types $\mathbf{B}_{\text{ind}} : \Pi P : \Box_\ell, P \to P \to \mathbf{B} \to P$ is separate from the eliminator for type families $\mathbf{B}_{\text{rec}} : \Pi P : \mathbf{B} \to \Box_\ell, P\,\mathsf{tt} \to P\,\mathsf{ff} \to \Pi b : \mathbf{B}, \theta_{\mathbf{B}} P b$.

$\mathbf{B}_{\text{ind}}$ behaves like the usual eliminator applied to a constant predicate. $\mathbf{B}_{\text{rec}}$ involves $\theta_{\mathbf{B}} P$, the linearisation of P defined by $\theta_{\mathbf{B}} := \lambda P n, \mathbf{B}_{\text{ind}}((\mathbf{B} \to \Box_\ell) \to \Box_\ell)(\lambda Q, Q\,\mathsf{tt})(\lambda Q, Q\,\mathsf{ff}) b P$, or, more clearly, $\theta_{\mathbf{B}} P\,\mathsf{tt} := P\,\mathsf{tt}$ and $\theta_{\mathbf{B}} P\,\mathsf{ff} := P\,\mathsf{ff}$.

While our model doesn't model integers, it should also be possible, with the eliminator for types as $\mathbf{N}_{\text{ind}} : \Pi P : \Box_\ell, P \to (\mathbf{N} \to P \to P) \to \mathbf{N} \to P$ and the eliminator for type families as $\mathbf{N}_{\text{rec}} : \Pi P : \mathbf{N} \to \Box_\ell, P0 \to (\Pi n : \mathbf{N}, Pn \to P(Sn)) \to \Pi n : \mathbf{N}, \theta_{\mathbf{N}} P n$.

$\mathbf{N}_{\text{ind}}$ also behaves like the usual eliminator applied to a constant predicate, and $\mathbf{N}_{\text{rec}}$ also involves a linearisation of P named $\theta_{\mathbf{N}} P$, defined as $\theta_{\mathbf{N}} := \lambda P n, \mathbf{N}_{\text{ind}}((\mathbf{N} \to \Box_\ell) \to \Box_\ell)(\lambda Q, Q0)(\lambda n \theta' Q, \theta'(Q \circ S)) n P$, or, more clearly, $\theta_{\mathbf{N}} P0 := P0$ and $\theta_{\mathbf{N}} P(Sn) := \theta_{\mathbf{N}}(P \circ S) n$. In particular, they coincide on integers of the form $S(\cdots(S0)\cdots)$.

Other writers may swap ind and rec.

The model took

$$\begin{aligned} \mathsf{Ctx} &:= \mathsf{Type} \\ \mathsf{Typ}\,\Gamma &:= \Gamma \to (\Sigma(A : \mathsf{Type}), \Pi(i : I), (O\ i \to A) \to A) \\ \mathsf{Trm}\ \Gamma\ A &:= \Pi(\gamma : \Gamma), \mathfrak{p}_1(A\ \gamma) \end{aligned}$$

The rest follows. As is the case with the other models here besides presheaves, contexts are just types.

This models contrasts with how sheaves do not need to weaken the theory down to Baclofen TT to add effects

3.3 Exceptional

The exceptional model (for exception type E) interprets a type as a pair of a type and a map $E \to A$. It is inconsistent, at least when E is inhabited, however, together with parametricity, a consistent model can be recovered.

This models highlights the importance of computations, since an inconsitent type theory can still be used to state a few thing as long as it computes properly.

3.4 Presheaves

The most time consuming model, where neither universes nor dependant function types could be implemented. The main issue is the straightforward implementation is not strict, meaning equalities have to be proven, and transport along them must be handled.

To give the broad strokes of the model, we use the following definition of a category, mostly to name the fields. It must be understood as a record (a fancy Σ), so the projection are named according to the fields.

$$\mathsf{Cat} := \left\{ \begin{array}{rl} \mathsf{obj} : & \mathsf{Type} \\ \mathsf{hom} : & \mathsf{obj} \to \mathsf{obj} \to \mathsf{Type} \\ \mathsf{id} : & \Pi(X : \mathsf{obj}), \mathsf{hom}\ X\ X \\ \mathsf{comp} : & \Pi(XYZ : \mathsf{obj}), \mathsf{hom}\ Y\ Z \to \mathsf{hom}\ X\ Y \to \mathsf{hom}\ X\ Z \\ & \cdots \end{array} \right\}$$

We fix a category $\mathbf{C} : \mathsf{Cat}$, so the first fields of the model are given by :

$$\mathsf{Ctx} := \left\{ \begin{array}{rl} \mathsf{Ctx\_obj} : & \mathsf{obj}\ \mathbf{C} \to \mathsf{Type} \\ \mathsf{Ctx\_hom} : & \Pi(XY : \mathsf{obj}\ \mathbf{C}), \mathsf{hom}\ \mathbf{C}\ X\ Y \to \mathsf{Ctx\_obj}\ Y \to \mathsf{Ctx\_obj}\ X \\ & \cdots \end{array} \right\}$$

$$\mathsf{Typ}\,\Gamma := \left\{ \begin{array}{rl} \mathsf{Typ\_obj} : & \Pi(X : \mathsf{obj})\ \mathsf{Ctx\_obj}\ \Gamma\ X \to \mathsf{Type} \\ \mathsf{Typ\_hom} : & \Pi(XY : \mathsf{obj}\ \mathbf{C})(f : \mathsf{hom}\ \mathbf{C}\ X\ Y)(y : \mathsf{Ctx\_obj}\ \Gamma\ Y), \\ & \quad \mathsf{Typ\_obj}\ Y\ y \to \mathsf{Typ\_obj}\ X\ (\mathsf{Ctx\_hom}\ \Gamma\ X\ Y\ f\ y) \\ & \cdots \end{array} \right\}$$

Notably, context are presheaves and not just types, and, strictly speaking, types are only presheaves in an empty context.

This model highlights why studying sheaves separately from presheaves is interesting.

3.5 Sheaves

The most important one is of course the model using sheaves.

The model requires univalence and used rewrite rules to simulate HIT, which are absent in Rocq9.0.0.

Types were interpreted as sheaves as described in definition 2.14. Product types are relatively easy as an aribtray product of sheaves is still a sheaf. Positives types, in my case boolean, where contructed as the QIT with the same constructor as the original (true and false), together with ask and ε. The resulting type is equivalent to the sheafification of booleans but does not compute exactly the same.

One of the interest of type theory is that it yield meaningful computations from proof, so we were careful about that.

4 ShTT

4.1 Martin Baillon's ShTT

Our work was meant to generalise the work of Pierre-Marie Pédrot and his past PhD student, Martin Baillon, did in the thesis of the latter [1] and afterwards, which itself follows up on a note from Jaber and Coquand [3]. He worked on a roughly similar MLTT extended with boolean and a generic cohen real $\alpha : \mathbf{N} \to \mathbf{B}$, where the context was extended with finite knowledge of this function, which allowed α to compute on closed form integers.

The sheaf structure is the one appearing in this geometric formula :

$$\left(\bigwedge_{n:\mathfrak{N}} \top \to (n \mapsto_\alpha \mathfrak{tt}) \vee (n \mapsto_\alpha \mathfrak{ff})\right) \wedge \left(\bigwedge_{n:\mathfrak{N}} (n \mapsto_\alpha \mathfrak{tt}) \wedge (n \mapsto_\alpha \mathfrak{ff}) \to \bot\right)$$

With atoms $n \mapsto_\alpha b$ for all $n : \mathfrak{N}$ and $b : \mathfrak{B}$.

The term then extends those of MLTT with only α :

$$M, N ::= \ldots \mid \alpha$$

Together with the usual context, judgments depends on a *forcing context* containing the finite approximation of α, given by the following syntax, when n and b an integer and boolean respectively

$$\mathcal{L} ::= \mathcal{L}, n \mapsto b \mid \cdot$$

We write $n \mapsto_{\mathcal{L}} b$ when $n \mapsto b$ appears in $\mathcal{L}$, and $n \not\mapsto_{\mathcal{L}}$ when neither $n \mapsto \mathfrak{tt}$ nor $n \mapsto \mathfrak{ff}$ do.

The type theory has rules :

$$\mathrm{n}\frac{\mathcal{L}, \Gamma \vdash \mathcal{J}_0 \quad \cdots \quad \mathcal{L}, \Gamma \vdash \mathcal{J}_n}{\mathcal{L}, \Gamma \vdash \mathcal{J}}$$

whenever the following is a rule of MLTT (with booleans)

$$\mathrm{n}\frac{\Gamma \vdash \mathcal{J}_0 \quad \cdots \quad \Gamma \vdash \mathcal{J}_n}{\Gamma \vdash \mathcal{J}}$$

With $\mathcal{J}$ a judgment of conversion or well formation, and n the name of the rule.

An exception is made for Wf-Empty, which becomes

$$\textsc{Wf-Empty}\frac{}{\cdot, \cdot \vdash \mathsf{well\text{-}formed}}$$

The new rules, using overlining to turn meta-theoretical integers and booleans into the corresponding terms, are :

$$\textsc{Wf-Ext-Forc}\frac{\mathcal{L}, \Gamma \vdash \mathsf{well\text{-}formed} \quad n \not\mapsto_{\mathcal{L}}}{\mathcal{L}, n \mapsto b, \Gamma \vdash \mathsf{well\text{-}formed}}$$

$$\textsc{Gen}\frac{\mathcal{L}, \Gamma \vdash \mathsf{well\text{-}formed}}{\mathcal{L}, \Gamma \vdash \alpha \equiv \alpha : \mathbf{N} \to \mathbf{B}} \qquad \textsc{Ask}\frac{\mathcal{L}, \Gamma \vdash \mathsf{well\text{-}formed} \quad n \mapsto_{\mathcal{L}} b}{\mathcal{L}, \Gamma \vdash \alpha\overline{n} \equiv \overline{b}}$$

$$\textsc{Split}\frac{\mathcal{L}, n \mapsto \mathfrak{tt}, \Gamma \vdash M \equiv M' : A \quad \mathcal{L}, n \mapsto \mathfrak{ff}, \Gamma \vdash M \equiv M' : A \quad n \not\mapsto_{\mathcal{L}}}{\mathcal{L}, \Gamma \vdash M \equiv M' : A}$$

Amongst other thing, this theory can be used to show that from any term $\cdot \vdash M \equiv M : (\mathbf{N} \to \mathbf{B}) \to \mathbf{N}$ of MLTT, a term $\{M\}$ of MLTT proving its continuity can be constructed, *i.e.* $\cdot \vdash \{M\} \equiv \{M\} : \mathsf{continuous}\ M$ can be derived.

4.2 ShTT

We sought to create a type theory based on general sheaves, based on a simple geometric formula. As of our latest draft, we use $\bigwedge_{i:I}\left(\top \to \bigvee_{\alpha:A\ i} O_{i,\alpha}\right)$. It is the simplest one involving branching.

While in 4.1 conversion rules could be added, it is imposible without specialising the system and deciding what $O_{i,\alpha}$ means. We needed to add term by which the sheaf nature of our types would appear. Since the model of each type was a sheaf, it contained a map ask, which we sought to add to the syntax under the name $\digamma$. Such map could be obtained in 4.1 as $\mathsf{ask}_n\ T\ t_{\mathsf{tt}}\ t_{\mathsf{ff}} := \mathbf{B}_{\mathrm{rec}}\ (\lambda\_.T)\ t_{\mathsf{tt}}\ t_{\mathsf{ff}}\ (\alpha\ \overline{n})$.

We suppose we have a type Ω of *atoms*. We then set a type I with decidable equality, and a (morally pointwise finite) type family $A : I \to \mathsf{Type}$ standing for arities, and finally a family $O : \Pi i : I, A\ i \to \Omega$, writing $O_{i,\alpha}$ instead of $O\ i\ \alpha$ for space concerns.

Those are also the types we use to give the shape of the geometric formula.

The terms extend those of MLTT as follows :

$$M, N ::= \ldots \mid \digamma_i(M_\alpha)_{\alpha:A\ i}$$

When instanciating with finite Ai, it would rather be $\digamma_i M_1 \cdots M_n$.

The forcing contexts $\mathcal{L}$ are now subset of Ω (lists accessed only through whether they include some elements), and never cause ill-formation.

The conversion rules also copy those from MLTT by adding a forcing context as in 4.1, without the WF-Empty exception.

The new conversion rules are as follows :

$$\textsc{Dig-}i\frac{\forall\alpha\alpha'.\mathcal{L}, O_{i,\alpha}, O_{i,\alpha'}, \Gamma \vdash M_\alpha \equiv M'_{\alpha'} : A}{\mathcal{L}, \Gamma \vdash \digamma_i(M_\alpha)_\alpha \equiv \digamma_i(M'_\alpha)_\alpha : A}$$

The above rule is both the expected congruence rule for conversion and also a compatibility rule for typing that would be stated separately in a system with a pure typing judgement.

$$\textsc{Ask-}i\frac{\forall\alpha\alpha'.\mathcal{L}, O_{i,\alpha}, O_{i,\alpha'}, \Gamma \vdash M_\alpha \equiv M'_{\alpha'} : A}{\mathcal{L}, O_{i,\alpha}, \Gamma \vdash \digamma_i(M_{\alpha'})_{\alpha'} \equiv M'_\alpha : A}$$

The above rule is meant to hardwire the coherence ε of the sheaf.

$$\textsc{Dig-Ev}\frac{\mathcal{L}, \Gamma \vdash N \equiv N' : A \qquad \forall\alpha\alpha', \mathcal{L}, O_{i,\alpha}, O_{i,\alpha}, \Gamma \vdash M_\alpha \equiv M_{\alpha'} : \Pi x : A, B}{\mathcal{L}, \Gamma \vdash (\digamma_i(M_\alpha)_\alpha)N \equiv \digamma(M_\alpha N)_\alpha : B(N/x)}$$

$$\textsc{Int-Rec-Dig}\frac{\begin{array}{c}\mathcal{L}, \Gamma \vdash A \equiv A : \mathbf{N} \to \square_{\mathbf{s}} \qquad \mathcal{L}, \Gamma \vdash M_0 \equiv M'_0 : \mathbf{N} \qquad \mathcal{L}, \Gamma \vdash M_S \equiv M'_S : \Pi n : \mathbf{N}, An \to A(Sn) \\ \forall\alpha\alpha', \mathcal{L}, O_{i,\alpha}, O_{i,\alpha}, \Gamma \vdash M_\alpha \equiv M_{\alpha'} : \mathbf{N}\end{array}}{\mathcal{L}, \Gamma \vdash \mathbf{N}_{\mathrm{rec}} A M_0 M_S (\digamma_i(M_\alpha)_\alpha) \equiv \digamma_i(\mathbf{N}_{\mathrm{rec}} A M_0 M_S M_\alpha)_\alpha : A\digamma_i(M_\alpha)_\alpha}$$

$$\textsc{Emp-Rec-Dig}\frac{\mathcal{L}, \Gamma \vdash A \equiv A : \bot \to \square_{\mathbf{s}} \qquad \forall\alpha\alpha', \mathcal{L}, O_{i,\alpha}, O_{i,\alpha}, \Gamma \vdash M_\alpha \equiv M_{\alpha'} : \bot}{\mathcal{L}, \Gamma \vdash \bot_{\mathrm{rec}} A(\digamma_i(M_\alpha)_\alpha) \equiv \digamma_i(\bot_{\mathrm{rec}} A M_\alpha)_\alpha : A\digamma_i(M_\alpha)_\alpha}$$

This last few rules allows for pursuing computations on $\digamma$.

As such, terms of type $\mathbf{N}$ should always be convertible to a term $\overline{n'}$, where n' is of type $\mathfrak{N}'$

$$\begin{array}{l}\mathsf{Inductive}\ \ \mathfrak{N}' : \mathsf{Type} := \\ \mid 0' : \mathfrak{N}' \\ \mid S' : \mathfrak{N}' \to \mathfrak{N}' \\ \mid \mathsf{ask}' : \Pi(i : I), (A\ i \to \mathfrak{N}') \to \mathfrak{N}'\end{array}$$

With $\overline{0'} := 0$, $\overline{S'\ n'} := S\ \overline{n'}$, and $\overline{\mathsf{ask}\ i\ k} := \digamma_i(\overline{k}\ \alpha)_\alpha$. Furthermore, the same index i should not occur twice in a branch.

This theory is unsatisfying with regards to the goal we set. For example, to instantiate it with 4.1, even adding the necessary generic function α, and setting $I := \mathbf{N}$, $Ai = \mathbf{B}$ and $O_{n,b} := n \mapsto_\alpha b$ so that Dig-i may play the role of Split, we have no way to provide the compatibility proof $n \mapsto \mathfrak{tt}, n \mapsto \mathfrak{ff} \vdash M_{\mathsf{tt}} \equiv M_{\mathsf{ff}}$, which must be some form of ex-falso.

The Ask rule itself cannot be emulated, although this might be more an issue of level of generality.

5 Logical relations

A rather standard manner of studying the properties of a type theory is through logical relations. We roughly followed the recipe of Abel [4], which, since it is meant for a system closer to MLTT, which would ease later scaling. In pratice, we used LogRel Rocqas a reference point [5]. We use « logical relations » in a rather precise manner to designate this recipe. Roughly speaking, we establish a reduction relation and interpret terms by their reduction to a normal form.

5.1 System T extension

To get a hang of F and logical relations we started by studying a extension of system T with F

Again the terms simply extend those of system T :

$$M, N ::= \dots \mid F_i(M_\alpha)_\alpha$$

Rules from system T are then appropriately completed with a forcing context, and the same final rules are appended.

$$\textsc{Dig-}i\frac{\forall \alpha\alpha'.\mathcal{L}, O_{i,\alpha}, O_{i,\alpha'}, \Gamma \vdash M_\alpha \equiv M'_{\alpha'} : A}{\mathcal{L}, \Gamma \vdash F_i(M_\alpha)_\alpha \equiv F_i(M'_\alpha)_\alpha : A}$$

$$\textsc{Ask-}i\frac{\forall \alpha\alpha'.\mathcal{L}, O_{i,\alpha}, O_{i,\alpha'}, \Gamma \vdash M_\alpha \equiv M'_{\alpha'} : A}{\mathcal{L}, O_{i,\alpha}, \Gamma \vdash F_i(M_{\alpha'})_{\alpha'} \equiv M'_\alpha : A}$$

$$\textsc{Dig-Ev}\frac{\mathcal{L}, \Gamma \vdash N \equiv N' : A \qquad \forall \alpha\alpha', \mathcal{L}, O_{i,\alpha}, O_{i,\alpha}, \Gamma \vdash M_\alpha \equiv M_{\alpha'} : A \to B}{\mathcal{L}, \Gamma \vdash (F_i(M_\alpha)_\alpha)N \equiv F(M_\alpha N)_\alpha : B}$$

$$\textsc{Int-Rec-Dig}\frac{\begin{array}{c}\mathcal{L}, \Gamma \vdash M_0 \equiv M'_0 : \mathbf{N} \qquad \mathcal{L}, \Gamma \vdash M_S \equiv M'_S : \mathbf{N} \to A \to A \\ \forall \alpha\alpha', \mathcal{L}, O_{i,\alpha}, O_{i,\alpha}, \Gamma \vdash M_\alpha \equiv M_{\alpha'} : \mathbf{N}\end{array}}{\mathcal{L}, \Gamma \vdash \mathbf{N}_{\mathrm{rec}} M_0 M_S (F_i(M_\alpha)_\alpha) \equiv F_i(\mathbf{N}_{\mathrm{rec}} M_0 M_S M_\alpha)_\alpha : A}$$

$$\textsc{Emp-Rec-Dig}\frac{\forall \alpha\alpha', \mathcal{L}, O_{i,\alpha}, O_{i,\alpha}, \Gamma \vdash M_\alpha \equiv M_{\alpha'} : \bot}{\mathcal{L}, \Gamma \vdash \bot_{\mathrm{rec}}(F_i(M_\alpha)_\alpha) \equiv F_i(\bot_{\mathrm{rec}} M_\alpha)_\alpha : A}$$

5.2 Reducibility

We then applied logical relations to our system T variant.

We define our reduction relation as follows :

$$\frac{M \rightsquigarrow M'}{MN \rightsquigarrow M'N} \qquad \frac{}{(\lambda x.M)N \rightsquigarrow M(N/x)}$$

$$\frac{M \rightsquigarrow M'}{\mathbf{N}_{\mathrm{rec}} M_0 M_S M \rightsquigarrow \mathbf{N}_{\mathrm{rec}} M_0 M_S M'} \qquad \frac{}{\mathbf{N}_{\mathrm{rec}} M_0 M_S O \rightsquigarrow M_0} \qquad \frac{}{\mathbf{N}_{\mathrm{rec}} M_0 M_S (SM) \rightsquigarrow M_S M (\mathbf{N}_{\mathrm{rec}} M_0 M_S M)}$$

$$\frac{M \rightsquigarrow M'}{\bot_{\mathrm{rec}} M \rightsquigarrow \mathbf{N}_{\mathrm{rec}} M'}$$

And our neutrals as follows, through the predicate Ne :

$$\frac{}{\mathrm{Ne}\, x} \qquad \frac{\mathrm{Ne}\, n}{\mathrm{Ne}\, nM} \qquad \frac{\mathrm{Ne}\, n}{\mathrm{Ne}\, \mathbf{N}_{\mathrm{rec}} M_0 M_S n} \qquad \frac{\mathrm{Ne}\, n}{\mathrm{Ne}\, \bot_{\mathrm{rec}} n}$$

We can then define a reducibility predicate $\Vdash$ as follows, by induction on the type.

- For type $A \to B$, $\mathcal{L}, \Gamma \Vdash M \equiv M' : A \to B$, if for all $\mathcal{L}', \Gamma'$ such that $\mathcal{L} \subset \mathcal{L}'$, and Γ is a prefix of Γ', if $\mathcal{L}', \Gamma' \Vdash N \equiv N' : A$, then $\mathcal{L}', \Gamma' \Vdash MN \equiv M'N'$.
- For type $\mathbf{N}$, we define inductively :

$$\frac{M \rightsquigarrow^* 0 \qquad M' \rightsquigarrow^* 0}{\mathcal{L}, \Gamma \Vdash M \equiv M' : \mathbf{N}} \qquad \frac{M \rightsquigarrow^* SN \qquad M' \rightsquigarrow^* SN' \qquad \mathcal{L}, \Gamma \Vdash N \equiv N' : \mathbf{N}}{\mathcal{L}, \Gamma \Vdash M \equiv M' : \mathbf{N}}$$

$$\frac{M \rightsquigarrow^* n \qquad M' \rightsquigarrow^* n' \qquad \mathrm{Ne}\, n \qquad \mathrm{Ne}\, n'}{\mathcal{L}, \Gamma \Vdash M \equiv M' : \mathbf{N}}$$

$$\frac{M \rightsquigarrow^* \digamma_i(M_\alpha)_\alpha \quad M' \rightsquigarrow^* \digamma_i(M'_\alpha)_\alpha \quad \forall \alpha, \alpha' \Vdash M_\alpha \equiv M'_{\alpha'} : \mathbf{N}}{\mathcal{L}, \Gamma \Vdash M \equiv M' : \mathbf{N}}$$

$$\frac{M \rightsquigarrow^* \digamma_i(M_\alpha)_\alpha \quad M' \rightsquigarrow^* M'_\alpha \quad O_{i,\alpha} \in \mathcal{L} \quad \forall \alpha, \alpha', \mathcal{L}, \Gamma \Vdash M_\alpha \equiv M'_{\alpha'} : \mathbf{N}}{\mathcal{L}, \Gamma \Vdash M \equiv M' : \mathbf{N}}$$

$$\frac{M \rightsquigarrow^* M_\alpha \quad M' \rightsquigarrow^* \digamma_i(M'_\alpha)_\alpha \quad O_{i,\alpha} \in \mathcal{L} \quad \forall \alpha, \alpha', \mathcal{L}, \Gamma \Vdash M_\alpha \equiv M'_{\alpha'} : \mathbf{N}}{\mathcal{L}, \Gamma \Vdash M \equiv M' : \mathbf{N}}$$

- For type $\bot$, we use a similar definition as above excluding the case mentioning 0 and the case mentioning S

We can then show the usual results, all proven in Rocq :

Lemma 5.1 (reflection)**.** *If* $\mathrm{Ne}\, n$ *and* $\mathrm{Ne}\, n'$*, then* $\mathcal{L}, \Gamma \Vdash n \equiv n' : A$*.*

Lemma 5.2 (reification)**.** *If* $\mathcal{L}, \Gamma \Vdash M \equiv M' : A$*, then* M *and* M' *normalise.*

Theorem 5.3 (soundness)**.** *If* $\mathcal{L}, \Gamma \vdash M \equiv M' : A$*, then if* $\mathcal{L} \subset \mathcal{L}'$*,* $\mathcal{L}', \Gamma' \Vdash \sigma \equiv \sigma' : \Gamma$*,* $\mathcal{L}', \Gamma' \Vdash M\sigma \equiv M'\sigma' : A$

Where substitution are encoded as lists of terms and reducibility of substitution is reducibility of each of these terms. Transitivity (Trans) made up the bulk of the proof.

We can then deduce that our theory normalises.

A canonicity result could easily follow as well, once hereditary reduction has been defined, with normal form of booleans some variant of dialogue trees instead of sheaves, since ε has no corresponding reduction rule.

6 Conclusion

This constitutes a very incomplete work. I don't think attempting to prove normalisation of ShTT is the next step, same for any theories at the same level of generality. It might be more interesting to either look at continuity for a more general α than in Martin Baillon's work, or find a simple enough theory with non-empty intersections.

References

[1] Martin Baillon. "Continuity in Type Theory". Theses. Nantes Université, Dec. 2023. URL: https://theses.hal.science/tel-04617881.

[2] Saunders MacLane and Ieke Moerdijk. *Sheaves in geometry and logic: A first introduction to topos theory.* Springer Science & Business Media, 2012.

[3] Thierry Coquand and Guilhem Jaber. "A Note on Forcing and Type Theory". In: *Fundam. Inf.* 100.1–4 (Jan. 2010), pp. 43–52. ISSN: 0169-2968.

[4] Andreas Abel, Joakim Öhman, and Andrea Vezzosi. "Decidability of conversion for type theory in type theory". In: *Proc. ACM Program. Lang.* 2.POPL (Dec. 2017). DOI: 10.1145/3158111. URL: https://doi.org/10.1145/3158111.

[5] Arthur Adjedj et al. "Martin-Löf à la Coq". In: *Proceedings of the 13th ACM SIGPLAN International Conference on Certified Programs and Proofs, CPP 2024, London, UK, January 15-16, 2024*. Ed. by Amin Timany et al. ACM, 2024, pp. 230–245. DOI: 10.1145/3636501.3636951. URL: https://doi.org/10.1145/3636501.3636951.