Debunking Sheaves

Pierre-Marie Pédrot

February 8, 2021

Abstract

In this note I will give a direct reformulation of sheaves in type theory that I came up with after translating back the topos theoretic notions straight out of the reference book *Sheaves in Geometry and Logic* [MM92]. The main takeaways of this note are the following.

- Despite the name, sheaves have nothing to do with presheaves.
- What it means to be a sheaf can be defined in a basic type theory.
- Sheafification, by contrast, requires advanced machinery and generates monsters.

As a coproduct, this also gives a straightforward explanation for Beth semantics without the need to go through Kripke semantics.

Introduction

Sheaves were always veiled into a shroud of arcane mystery to me. They immediately summon a complex imagery of topology, abstract geometry and abstruse categorical nonsense. Even after having written a paper [Péd20] giving a computational reconstruction of presheaves, sheaves remained more complex. You have this notion of covering, and any coverage of a value can be pasted together to produce a legitimate value. It seems a priori very hard to give a type-theoretic account of this miraculous phenomenon, which looks more like the popsci description of topology where choice-wielding toddlers keep deforming psychedelic playdough in way too many dimensions. This standpoint has always left me beffuzled, since I personally do not believe in graphical proofs.

The categorical standpoint was not helpful either. In the specific case of sheaves, I hold a specific grudge against the nLab wiki, which generally deserves suspicion given the amount of categorical nonsense it can spit out to describe basic structures[1]. The main problem with categorical sheaves is that they are described in a topos-theoretic way, which is very weird from a type-theoretical perspective. As a type theory, elementary topoi are both very weak and very strong. They are weak because they do not feature basic constructs like universes

[1]Before the reader starts complaining about my bad faith, I know this is a wiki, so in theory I should be able to expand it myself. But do you think an article written in Malayalam would be welcome on the German Wikipedia?

and dependent elimination, i.e. they are a only model of HOL in general. Similarly, some other features can only be achieved through low-level elaborate encodings reminiscent of set theory which hide the computational intent which would be obvious with a built-in high-level construct. Conversely, topoi are way too powerful. They gleefully hardwire UIP, funext and propext, which on my scale of acceptability are respectively rated OK-ish, yikes and nuclear holocaust. If one wants to understand sheaves in an intensional setting this is a bad start.

So what I did was simple. Armed with my understanding of presheaves as a model of type theory, I would simply try to state all the sheaf paraphernalia in direct style in the source theory obtained through the presheaf construction. Some of this was already written in an almost type-theoretic way, notwithstanding the baking in of extensionality principles. This was the case for Grothendieck topologies. Other stuff was more tricky, and as far as my knowledge goes, I had never read Definition 2 in such clarity anywhere.

To be fair, in hindsight most of the contents of this note can be extracted from the deep paper about modalities by Rijke, Shulman and Spitters [RSS20]. Yet I believe that the latter does *too much*, insofar as its aim is to treat a much more general situation, with the added complexity hiding the simplicity of sheaves when stated at the right level of abstraction. Also, the fact that they adopt univalence as a founding principle also makes the structures more involved as they need to carefully handle the rich contents of equality.

This note performs the public service of giving a simple explanation of sheaves to people who, like me, have a deficient comprehension of category theory but a clear understanding of type theory[2]. It is not groundbreaking in any way. I wrote it mainly for myself, and should serve as a reference whenever I want to understand what those (*edited*) category terrorists are up to. It might even be useful to somebody else, who knows.

This note is divided in the following way. Section 1 gives a purely type-theoretical description of sheaves without any mention of presheaves whatsoever. Section 2 complains about topos theory. Section 3 is the longest one and does the job of translating bit-by-bit the definitions from [MM92] into an extensional type theory meant to represent **Set**. Section 4 gives a high-level description of Kripke and Beth semantics as degenerate presheaf and sheaf models respectively. Section 5 gives some intuition about the computational content of sheafification.

1 Sheaves

In this section we assume an ambient intensional type theory as a flavour of CIC. It will feature a universe of propositions `Prop` and a hierarchy of universes of types $\texttt{Type}_i$. We will silently use typical ambiguity. Inhabitants of the `Prop` universe are intended to be proof-irrelevant but we do not rely on it for the definitions of this section.

Definition 1. A `Prop`-valued monad is given by the following.

- $\mathsf{J} : \texttt{Prop} \to \texttt{Prop}$

[2] As such, and like all my notes, it is also a rant against category theory.

- $\eta : \Pi(P : \mathtt{Prop}).\, P \to \mathsf{J}\ P$
- $\gg\!= : \Pi(P\, Q : \mathtt{Prop}).\, \mathsf{J}\ P \to (P \to \mathsf{J}\ Q) \to \mathsf{J}\ Q$

Note that we do not require the usual equations for identity and associativity of the combinators for Type-valued monads. If propositions were proof-irrelevant as intended, all equations would indeed hold trivially. We assume in the remainder of this section a fixed Prop-valued monad J.

Definition 2. A type $A : \mathtt{Type}$ is a J-sheaf whenever there exists

- $\mathtt{ask}_A : \Pi(P : \mathtt{Prop}).\, \mathsf{J}\ P \to (P \to A) \to A$
- $\sigma_A : \Pi(P : \mathtt{Prop})\,(j : \mathsf{J}\ P)\,(x : A).\, \mathtt{ask}_A\ P\ j\ (\lambda p : P.\, x) = x$

Assuming the ambient type theory features some form of quotients as QITs or HITs, one can define the sheafification in a concise way.

Definition 3. We define the quotient inductive type $\mathscr{S}_{\mathsf{J}}$ as

$$
\begin{array}{l}
\mathtt{Inductive}\ \mathscr{S}_{\mathsf{J}}\ (A : \mathtt{Type}) : \mathtt{Type} := \\
\mid \mathtt{ret} : \Pi(x : A).\, \mathscr{S}_{\mathsf{J}}\ A \\
\mid \mathtt{ask} : \Pi(P : \mathtt{Prop}).\, \mathsf{J}\ P \to (P \to \mathscr{S}_{\mathsf{J}}\ A) \to \mathscr{S}_{\mathsf{J}}\ A \\
\mid \sigma : \Pi(P : \mathtt{Prop})\,(j : \mathsf{J}\ P)\,(x : \mathscr{S}_{\mathsf{J}}\ A).\, \mathtt{ask}\ P\ j\ (\lambda p : P.\, x) = x
\end{array}
$$

This presentation can be found in [RSS20] in a generalized form. The richer form is needed to handle less degenerate forms of modalities, and furthemore needs to account for higher coherences in the univalent setting.

We liberally assume funext and UIP to prove the theorems below. The former is needed as soon as one phrases monadic structures in type theory by translating them directly from category theory. The latter is needed to get a simple treatment of QITs that does not require full-blown univalence.

Theorem 1. *$\mathscr{S}_{\mathsf{J}}$ is a* Type*-valued monad, with*

$$
\begin{array}{lcl}
\eta\ x & := & \mathtt{ret}\ x \\
(\mathtt{ret}\ x) \gg\!= f & := & f\ x \\
(\mathtt{ask}\ P\ i\ k) \gg\!= f & := & \mathtt{ask}\ P\ i\ (\lambda(p : P).\, (k\ p) \gg\!= f).
\end{array}
$$

Theorem 2. *A type A is a $\mathscr{S}_{\mathsf{J}}$-algebra iff it is a* J*-sheaf.*

Theorem 3. *Sheafification is the call-by-name embedding of $\mathscr{S}_{\mathsf{J}}$.*

Sheafification can thus be understood as lifting a monad on Prop to a monad on Type.

Note that this is not the standard way to describe sheafification, since it usually relies on a more low-level presentation of quotients in topos theory. The standard definition is fairly bad since it needs a strong form of the axiom of choice in the ambient theory to get anything done. For the sake of completeness, we expose below this encoding as well as related concepts.

Definition 4. A type $A : \mathtt{Type}$ is J-separated whenever there is a proof

$$\pi_A : \Pi(P : \mathtt{Prop})\,(j : \mathsf{J}\ P)\,(x\,y : A).\,(P \to x = y) \to x = y.$$

In the presheaf model, J-separated types are unsurprisingly called J-separated presheaves. It is easy to check assuming funext that J-sheaves are in particular J-separated.

The standard sheafification process is typically defined in two steps and critically relies on propext.

Theorem 4. *Let $A : \mathtt{Type}$, we define A^+ as the quotient type*

$$A^+ : \mathtt{Type} := (\Sigma(P : \mathtt{Prop}).\,\mathsf{J}\ P \times (P \to A))/\sim$$

where

$$\begin{aligned}(P, i, \hat{x}) \sim (Q, j, \hat{y}) \quad := \quad & \exists(T : \mathtt{Prop})\,(\varphi : T \to P \wedge Q).\\ & (\mathsf{J}\ T) \wedge (\Pi(e : T).\,\hat{x}\ (\pi_1\ (\varphi\ e)) = \hat{y}\ (\pi_2\ (\varphi\ e)))\end{aligned}$$

Theorem 5. *For any $A : \mathtt{Type}$, A^+ is separated. If furthemore A is already separated then A^+ is a sheaf.*

Theorem 6. *In addition to the full set of extensionality axioms, assuming*

$$\mathtt{choice} : \Pi(A : \mathtt{Type}).\,||A|| \to A$$

where $||\cdot||$ is the Prop*-squash, then A^{++} is isomorphic to $\mathscr{S}_{\mathsf{J}}\ A$.*

There is even quotient-free variant of sheafification that can be defined in an elementary topos. It is literally a set-theoretical encoding of equivalence classes, which is so against the type-theorerical ethos than I will only write it in the dedicated Section 3.4.

2 A Brief Look at Topoi

Better not look at it for too long, we could turn into topologists — or worse.

Lawvere-Tierney topologies Prop-valued monads are usually called *Lawvere-Tierney topologies* the topos world. Since topoi bake in propext, the usual definition of Lawvere-Tierney topologies doesn't bother with finesse and states the η and $\gg\!=$ combinators as equalities. If we phrase the nLab definition in a slightly more type-friendly manner, this boils down to the following.

Definition 5. A Lawvere-Tierney topology is given by a term $\mathsf{J} : \mathtt{Prop} \to \mathtt{Prop}$ such that:

- $\mathsf{J}\ \top = \top$
- $\Pi(P : \mathtt{Prop}).\,\mathsf{J}\ (\mathsf{J}\ P) = \mathsf{J}\ P$
- $\Pi(P\,Q : \mathtt{Prop}).\,\mathsf{J}\ (P \wedge Q) = \mathsf{J}\ P \wedge \mathsf{J}\ Q$

Up to propext, the first equation corresponds to η, the second equation corresponds to the join operator μ of the monad, and the third equation states that the monad is strong. It is a bizarre way to express the monadic structure that smells of filters. Notably, it is missing functoriality, i.e. the `map` operation, which can be derived from those equations and a copious amount of extensionality. Since `Prop` is proof-irrelevant in topoi, no equations are required as they all hold trivially. I would argue that Definition 1 is more elegant and self-contained, or at least less demanding in the ambient theory.

Canonicity and Monsters There is a surprising claim that can be found in topos-theoretic writings, which is that some form of canonicity holds in Grothendieck topoi. For instance, by taking the double-negation monad $\mathsf{J}\ P := \neg\neg P$, allegedly the sheafification process would preserve booleans. This is unfortunately a typical confusion in model theory akin to "Markov's Principle holds in Kleene realizability". Since models are given semantically in the meta-theory, any crap from your meta can flow down into the model. Sheafification of booleans is literally given by $\mathscr{S}_{\neg\neg}\ \mathbb{B}$, which is some kind of eldritch tree from beyond the chasm of non-euclidean dimensions. It has thus about as much to do with booleans as the Necronomicon has to do with "Peppa Pig Goes Swimming".

But, indeed, if you secretly believe in classical logic in the meta-theory, and you implicitly identify this meta-theory with **Set**, then *clearly* $\mathscr{S}_{\neg\neg}\ \mathbb{B} \cong \mathbb{B}$ because *it is true*[3]. But as soon as you consider syntactic interpretations into explicit derivation systems, this illusion disappears and you are left with sheafified monsters. Since ZFC does not satisfy the witness property, canonicity goes down the drain.

Misleading Advertising As outlined in the introduction and formally observed in Section 1, you do not need presheaves at all to explain sheaves. Since the definition is purely internal, you do not even need to pick a base category or whatever, being a sheaf is something you can describe in a very simple type theory. It is worth repeating in bold in case you are skimming through this note.

Sheaves have nothing to do with presheaves.

I am no algebraic geometer, but my understanding of this fact is that it is a historical accident worsened by a commonplace ignorance of logic. From what I gather, sheaves were defined over topological spaces, so it was unclear at first that just considering presheaves would lead to an internal logic rich enough to formulate anything, let alone sheafness. As the internal definition is irrelevant to their most common uses it is not highlighted out.

Second, since the vast majority of mathematicians do not care about their foundations and work in **Set** fantasy land with full nonconstructive apparatus, I suspect that the internal notion of sheaf would look degenerate to a standard mathematician. In **Set** with classical logic, there are only two Lawvere-Tierney topologies: the identity monad $\mathsf{J}\ A := A$ and the

[3]Anybody phrasing Gödel's incompleteness theorem as "there are unprovable true statements" deserves a good whipping.

trivial monad $\mathsf{J}\ A := \top$. The sheaf category over the identity monad is just **Set**, so this is fairly useless. Sheaves over the trivial monad are just singleton sets, which is even worse.

To fully appreciate sheaves, you thus need an intuitionistic setting. And if you are not doing topos theory, what is the only way to force a generic mathematician to restrict the logic to be intuitionistic? Right, it is through the one complete model of intuitionistic logic, i.e. presheaves. Therefore, my probably incorrect theory is that most mathematicans do not care about a direct style description of sheaves since they will likely only ever use it composed with a presheaf model. This is quite regrettable because this hides the simplicity of the synthetic presentation under a pile of implementation details.

3 Through Hell and Back

In this section we explain how the high-level definitions from Section 1 correspond to the usual topological mumbo-jumbo one can find e.g. in the reference book [MM92]. Instead of explicitly writing set-theoretical constructs, we use the fact that **Set** is a model of type theory, albeit quite degenerated. We abstract away from **Set** by writing it as an extension of CIC called SetTT where additional principles hold.

3.1 Set theory as a type theory

Definition 6. SetTT is defined as CIC extended with extensionality principles.

Extensionality SetTT satisfies the extensionality rule.

$$\frac{\Gamma \vdash_{\mathsf{SetTT}} e : M =_A N}{\Gamma \vdash_{\mathsf{SetTT}} M \equiv N : A}$$

This allows to transparently rewrite propositional equations as if they were conversions. As such, the equations stated below can be equivalently stated as conversions or equalities. Note that in particular extensionality already implies UIP and funext.

Proof-irrelevance We have proof-irrelevance.

$$\vdash_{\mathsf{SetTT}} \mathsf{pi} : \Pi(P : \mathtt{Prop})\,(p\,q : P).\, p = q$$

Propositional extensionality We have propext.

$$\vdash_{\mathsf{SetTT}} \mathsf{propext} : \Pi(P\,Q : \mathtt{Prop}).\, (P \leftrightarrow Q) \to P = Q$$

And that is it! Most notably, we do not rely on classical logic. In fact, more than a type-theoretic reification of **Set**, SetTT may be thought of as a reification of an arbitrary topos rich enough to interpret dependent types.

As intended, SetTT has a straightforward intepretation into ZFC with a countable hierarchy of universes $(\mathcal{U}_i)_{i\in\mathbb{N}}$, where $A : \mathtt{Type}$ is intepreted as a set and $x : A$ is set membership. $\mathtt{Type}_i$ is interpreted as $\mathcal{U}_i$ and $\mathtt{Prop}$ is interpreted as the set of truth values $\mathcal{P}(\{\varnothing\})$ which is classically equal to $\mathbf{2}$. Through Aczel's trace encoding [Acz98], functions in SetTT are literally set-theoretical functions in ZFC between the underlying sets. In particular, predicates over a set A are isomorphic to functions $A \to \mathtt{Prop}$ as in topos theory.

These equivalences allow to readily translate from the usual phrasing in structural set theory to SetTT without much trouble.

3.2 Presheaves in SetTT

First of all, since we are not Bourbakists obsessed with the size of their sets, we don't really care about the difference between small and big sets. It doesn't even make much sense in our setting since smallness is relative to a given universe and we have a countable hierarchy of those. We could just pick an arbitrary base level and bump the level of all of the constructions given in this section. Therefore, we will stick to the standard usage of typical ambiguity and swear to the reader that we are not silently relying on universe inconsistencies.

Definition 7. Let us consider a small category $\mathbb{P}$. A presheaf over $\mathbb{P}$ is a functor $\mathbb{P} \to \mathbf{Set}$.

First, let us reformulate what a (small) category $\mathbb{P}$ is in SetTT.

Proposition 1. *In* SetTT*, a category is given by:*

- $|\mathbb{P}| : \mathtt{Type}$
- $\mathtt{Hom}_{\mathbb{P}} : |\mathbb{P}| \to |\mathbb{P}| \to \mathtt{Type}$
- $\mathtt{id}_{\mathbb{P}} : \Pi p : |\mathbb{P}|.\, \mathtt{Hom}_{\mathbb{P}}\ p\ p$
- $\circ_{\mathbb{P}} : \Pi p\, q\, r : |\mathbb{P}|.\, \mathtt{Hom}_{\mathbb{P}}\ p\ q \to \mathtt{Hom}_{\mathbb{P}}\ q\ r \to \mathtt{Hom}_{\mathbb{P}}\ p\ r$
- $\_ : \Pi(p\, q : |\mathbb{P}|)\, (\alpha : \mathtt{Hom}_{\mathbb{P}}\ p\ q).\ \circ_{\mathbb{P}}\ p\ p\ q\ (\mathtt{id}_{\mathbb{P}}\ p)\ \alpha = \alpha$
- $\_ : \Pi(p\, q : |\mathbb{P}|)\, (\alpha : \mathtt{Hom}_{\mathbb{P}}\ p\ q).\ \circ_{\mathbb{P}}\ p\ q\ q\ \alpha\ (\mathtt{id}_{\mathbb{P}}\ q) = \alpha$
- $\_ : \Pi(p\, q\, r\, s : |\mathbb{P}|)\, (\alpha : \mathtt{Hom}_{\mathbb{P}}\ p\ q)\, (\beta : \mathtt{Hom}_{\mathbb{P}}\ q\ r)\, (\gamma : \mathtt{Hom}_{\mathbb{P}}\ r\ s).$
  $\circ_{\mathbb{P}}\ p\ r\ s\ (\circ_{\mathbb{P}}\ p\ q\ r\ \alpha\ \beta)\ \gamma = \circ_{\mathbb{P}}\ p\ q\ s\ \alpha\ (\circ_{\mathbb{P}}\ q\ r\ s\ \beta\ \gamma)$

For legibility, we will simply write $\mathbb{P}$ for the type of objects, use infix notation for composition and mostly omit implicit arguments. We will write the set of morphisms $\mathtt{Hom}_{\mathbb{P}}\ p\ q$ using a preorder notation $p \leq q$, despite the morphisms being proof relevant. With this notational apparatus, the equations above are simply expressed as

$$\begin{aligned} \alpha \circ \mathtt{id} &= \alpha \\ \mathtt{id} \circ \alpha &= \alpha \\ \gamma \circ (\beta \circ \alpha) &= (\gamma \circ \beta) \circ \alpha. \end{aligned}$$

The above type is a dependent record, even if not written explicitly. Since we are in extensional type theory, the equations defined by the record can be omitted when destructing an instance of this record type, and treated as conversions known to hold. We reuse the same conventions hereafter.

Proposition 2. *In* SetTT *a presheaf over* $\mathbb{P}$ *at level* i *is given by*

- $A : \mathbb{P} \to \mathtt{Type}_i$
- $\theta_A : \Pi(p\, q : \mathbb{P}).\, q \leq p \to A\ p \to A\ q$
- $\_ : \Pi(p : \mathbb{P})\, (x : A\ p).\, \theta_A\ p\ p\ \mathtt{id}\ x = x$
- $\_ : \Pi(p\, q\, r : \mathbb{P})\, (x : A\ p)\, (\alpha : q \leq p)\, (\beta : r \leq q).\, \theta_A\ p\ r\ (\beta \circ \alpha)\ x = \theta_A\ q\ r\ \beta\ (\theta_A\ p\ q\ \alpha\ x)$

The θ_A function is known as the family of restriction morphisms. As above, we will keep its inferable arguments implicit.

Observe that the record above is parameterized by the universe level of the type family of the presheaf. The usual set-theoretic presentation covers the case of small presheaves, i.e. $i = 0$, but the definition is better given in the general, universe-polymorphic case. For reasons that will become obvious later on we call this record type **Type**.

Since we also have an explicit `Prop` universe in SetTT, it is possible to define a notion of proof-irrelevant presheaf.

Definition 8. A proof-irrelevant presheaf over $\mathbb{P}$ is simply given by

- $A : \mathbb{P} \to \mathtt{Prop}$
- $\theta_A : \Pi(p\, q : \mathbb{P}).\, q \leq p \to A\ p \to A\ q$

Since `Prop` is proof-irrelevant, the equations governing the behaviour of θ_A w.r.t. identity and composition are already true so they do not need to be stated explicitly. As above, we will call this record type **Prop**.

Definition 9. The type of global elements of a presheaf (A, θ_A) is given by

- $x : \Pi(p : \mathbb{P}).\, A\ p$
- $\_ : \Pi(p\, q : \mathbb{P})\, (\alpha : q \leq p).\, \theta_A\ \alpha\ (x\ p) = x\ q.$

We write $\mathsf{El} : \mathbf{Type} \to \mathtt{Type}$ for the function that maps a given presheaf to its type of global elements. There is a similar function $\mathbf{Prop} \to \mathtt{Prop}$ which we will conflate with El. As one may remark, equations over proofs of a propositions are trivial so this function is simply

$$\lambda((A, \theta_A) : \mathbf{Prop}).\, \Pi(p : \mathbb{P}).\, A\ p$$

implicitly using extensionality principles.

3.3 Presheaves as a model of type theory

We now introduce a model of type theory based on presheaves. So we now have to consider *two* type theories:

1. the target type theory, which will be the system in which we will write the model;
2. and the source type theory, which will be what will be validated in the model.

The target will simply be SetTT, since presheaves are usually described from within set theory. To fuel confusion, the source will actually contain SetTT, although we will not describe the whole translation in full detail. Let us call the source theory $\mathsf{PshTT}_{\mathbb{P}}$ for a fixed category $\mathbb{P}$ described in SetTT.

Intuitively, a type $\vdash_{\mathsf{PshTT}_{\mathbb{P}}} A : \mathtt{Type}_i$ is interpreted as a presheaf $[\![A]\!]$ at level i in SetTT. Similarly, propositions in $\mathsf{PshTT}_{\mathbb{P}}$ are interpreted as proof-irrelevant presheaves. A term $\vdash_{\mathsf{PshTT}_{\mathbb{P}}} M : A$ is thus interpreted as a global element $[M]$ of $[\![A]\!]$ in SetTT. Written formally,

$$\begin{array}{lll} \vdash_{\mathsf{PshTT}_{\mathbb{P}}} A : \mathtt{Type} & \text{implies} & \vdash_{\mathsf{SetTT}} [\![A]\!] : \mathbf{Type} \\ \vdash_{\mathsf{PshTT}_{\mathbb{P}}} M : A & \text{implies} & \vdash_{\mathsf{SetTT}} [M] : \mathsf{El}\ [\![A]\!] \end{array}$$

There is a bit of complexity added to the mix because one needs to handle dependent types, and in particular terms living in a context, but it does not change this fundamental intuition.

Note that this model crucially relies on extensionality. We will be stealthily rewriting equations known to hold in the current context without further warning. This allows to guarantee the property that

$$\vdash_{\mathsf{PshTT}_{\mathbb{P}}} M \equiv N : A \qquad \text{implies} \qquad \vdash_{\mathsf{SetTT}} [M] \equiv [N] : [\![A]\!].$$

Getting a presheaf model that works in intensional type theory requires a complete change of model. See the series of articles [JTS12; Jab+16; Péd20] that arguably solve this problem. Since we want to transfer back textbook results, we do not want to change the presheaf interpretation and will just stick to the usual description.

We do not describe the full model, but state the important result.

Theorem 7. $\mathsf{PshTT}_{\mathbb{P}}$ *contains* SetTT.

This allows to work in the presheaf model as if it were an extensional theory, which it actually is. In the remainder of this section we will focus on some basic type formers that will be needed to understand sheaves. For legibility, if $[\![A]\!] := (A, \theta_A)$ we will write $[\![A]\!]_p := A\ p$ and $\theta_A := \theta_{[\![A]\!]}$ with a little abuse of notation. These form the structural data needed to define a presheaf. Equations can be checked separately thanks to extensionality, and we will omit their proofs.

Proposition 3. *Given two presheaves* $[\![A]\!]$ *and* $[\![B]\!]$ *the non-dependent arrow* $[\![A \to B]\!]$ *is defined as*

$$[\![A \to B]\!]_p := \left\{ \begin{array}{l} \mathtt{fun} : \Pi(q : \mathbb{P})\,(\alpha : q \leq p).\, [\![A]\!]_q \to [\![B]\!]_q \\ \_ \quad : \Pi(q\,r : \mathbb{P})\,(\alpha : q \leq p)\,(\beta : r \leq q)\,(x : A\ q). \\ \qquad\qquad \theta_B\ \beta\ (\mathtt{fun}\ q\ \alpha\ x) = \mathtt{fun}\ r\ (\beta \circ \alpha)\ (\theta_A\ \beta\ x) \end{array} \right\}$$

$$\theta_{A \to B}\ (\alpha : q \leq p)\ f := \lambda(r : \mathbb{P})\,(\beta : r \leq q)\,(x : [\![A]\!]_r).\, f\ r\ (\beta \circ \alpha)\ x$$

The underlying type of $[\![A \to B]\!]$ is known as the type of natural transformations, and the equation of the subset type are called naturality equations. If in the above definition B is a proposition, then naturality becomes trivially true.

We now turn to the definition of the presheaf of universes, since it is a key feature of dependent type theories.

Proposition 4.

$$[\![\mathtt{Type}]\!]_p := \left\{ \begin{array}{ll} \mathtt{typ} & : \Pi(q:\mathbb{P})\,(\alpha : q \le p).\,\mathtt{Type} \\ \mathtt{hom} & : \Pi(q\,r:\mathbb{P})\,(\alpha : q \le p)\,(\beta : r \le q). \\ & \quad \mathtt{typ}\ q\ \alpha \to \mathtt{typ}\ r\ (\beta \circ \alpha) \\ \_ & : \Pi(q:\mathbb{P})\,(\alpha : q \le p)(x : \mathtt{typ}\ q\ \alpha). \\ & \quad \mathtt{hom}\ q\ q\ \alpha\ \mathtt{id}\ x = x \\ \_ & : \Pi(q\,r\,s:\mathbb{P})\,(\alpha : q \le p)\,(\beta : r \le q)\,(\gamma : s \le r)\,(x : \mathtt{typ}\ q\ \alpha). \\ & \quad \mathtt{hom}\ q\ s\ \alpha\ (\gamma \circ \beta)\ x = \mathtt{hom}\ r\ s\ (\beta \circ \alpha)\ \gamma\ (\mathtt{hom}\ q\ r\ \alpha\ \beta\ x) \end{array} \right\}$$

$$\theta_{\mathtt{Type}}\ (\alpha : q \le p)\ (A, \theta_A) := \left(\begin{array}{c} \lambda(r:\mathbb{P})\,(\beta : r \le q).\,A\ r\ (\beta \circ \alpha) \quad , \\ \lambda(r\,s:\mathbb{P})\,(\beta : r \le q)\,(\gamma : s \le r)\,(x : A\ r\ (\beta \circ \alpha)). \\ \quad \theta_A\ r\ s\ (\beta \circ \alpha)\ \gamma\ x \end{array} \right)$$

It is quite a mouthful, although it follows a pattern similar to functions. From a high level, it is no more than **Type** relativized to a starting point $p : \mathbb{P}$. In particular it is relatively easy to check that $\mathsf{El}\ [\![\mathtt{Type}]\!] \cong \mathbf{Type}$. Once again, propositions are interpreted similarly but all equations become trivially true.

Proposition 5.

$$[\![\mathtt{Prop}]\!]_p := \left\{ \begin{array}{ll} \mathtt{typ} & : \Pi(q:\mathbb{P})\,(\alpha : q \le p).\,\mathtt{Prop} \\ \mathtt{hom} & : \Pi(q\,r:\mathbb{P})\,(\alpha : q \le p)\,(\beta : r \le q). \\ & \quad \mathtt{typ}\ q\ \alpha \to \mathtt{typ}\ r\ (\beta \circ \alpha) \end{array} \right\}$$

$$\theta_{\mathtt{Prop}}\ (\alpha : q \le p)\ (A, \theta_A) := \left(\begin{array}{c} \lambda(r:\mathbb{P})\,(\beta : r \le q).\,A\ r\ (\beta \circ \alpha) \quad , \\ \lambda(r\,s:\mathbb{P})\,(\beta : r \le q)\,(\gamma : s \le r)\,(x : A\ r\ (\beta \circ \alpha)). \\ \quad \theta_A\ r\ s\ (\beta \circ \alpha)\ \gamma\ x \end{array} \right)$$

This is literally a copy-paste of the above where `Type` is replaced by `Prop` and all trivial equations are removed. Again, $\mathsf{El}\ [\![\mathtt{Prop}]\!] \cong \mathbf{Prop}$.

Proposition 6. *We will rely on some isomorphisms to ease the writing of presheaf elements.*

$$\mathsf{El}\ [\![A \to B]\!] \cong \left\{ \begin{array}{lll} f & : & \Pi(p:\mathbb{P}).\,[\![A]\!]_p \to [\![B]\!]_p \\ \_ & : & \Pi(p\,q:\mathbb{P})\,(\alpha : q \le p)\,(x : [\![A]\!]_p).\,\theta_B\ \alpha\ (f\ p\ x) = f\ q\ (\theta_A\ \alpha\ x) \end{array} \right\}$$

As usual the naturality equation can be ignored when $[\![B]\!] : \mathbf{Prop}$.

$$\mathsf{El}\ [\![A \to \mathtt{Type}]\!] \cong \left\{ \begin{array}{lll} B & : & \Pi(p:\mathbb{P}).\,[\![A]\!]_p \to \mathtt{Type} \\ \theta_B & : & \Pi(p\,q:\mathbb{P})\,(\alpha : q \le p)\,(x : [\![A]\!]_p).\,B\ p\ x \to B\ q\ (\theta_A\ \alpha\ x) \\ \_ & : & \Pi(p:\mathbb{P})\,(x : [\![A]\!]_p)\,(y : B\ p\ x).\,\theta_B\ p\ p\ \mathtt{id}\ x\ y = y \\ \_ & : & \Pi(p\,q\,r:\mathbb{P})\,(\alpha : q \le p).\,(\beta : r \le q)\,(x : [\![A]\!]_p)\,(y : B\ p\ x). \\ & & \quad \theta_B\ p\ r\ (\beta \circ \alpha)\ x\ y = \theta_B\ q\ r\ \beta\ (\theta_A\ \alpha\ x)\ (\theta_B\ p\ q\ \alpha\ x\ y) \end{array} \right\}$$

And again, we have the degenerate `Prop` *case.*

$$\mathsf{El}\ [\![A \to \mathtt{Prop}]\!] \cong \left\{ \begin{array}{lcl} B & : & \Pi(p:\mathbb{P}).\,[\![A]\!]_p \to \mathtt{Prop} \\ \theta_B & : & \Pi(p\,q:\mathbb{P})\,(\alpha : q \leq p)\,(x : [\![A]\!]_p).\,B\ p\ x \to B\ q\ (\theta_A\ \alpha\ x) \end{array} \right\}$$

We have now all the material to define dependent products.

Proposition 7. *Given* A : **Type** *and* $(B, \theta_B) : \mathsf{El}\ [\![A \to \mathtt{Type}]\!]$ *as given by the isomorphism from Proposition 6, we define*

$$[\![\Pi(x:A).\,B\ x]\!]_p := \left\{ \begin{array}{lcl} \mathtt{fun} & : & \Pi(q:\mathbb{P})\,(\alpha : q \leq p)\,(x : [\![A]\!]_q).\,B\ q\ x \\ \_ & : & \Pi(q\,r:\mathbb{P})\,(\alpha : q \leq p)\,(\beta : r \leq q)\,(x : [\![A]\!]_q). \\ & & \quad \theta_B\ \beta\ (\theta_A\ \beta\ x)\ (\mathtt{fun}\ q\ \alpha\ x) = \mathtt{fun}\ r\ (\beta \circ \alpha)\ (\theta_A\ \beta\ x) \end{array} \right\}$$

$$\theta_{\Pi(x:A).\,B\ x}\ (\alpha : q \leq p)\ f \quad := \quad \lambda(r:\mathbb{P})\,(\beta : r \leq q)\,(x : [\![A]\!]_r).\,f\ r\ (\beta \circ \alpha)\ x$$

Note that dependent products have the same computational content as non-dependent arrows. Actually, with a slight abuse of notation since we did not formally define types in a non-empty context, $[\![\Pi(x : A).\,B]\!] = [\![A \to B]\!]$. Similar isomorphisms apply to global elements of a dependent product, although we will not make them explicit.

Inductive types are defined pointwise, including when they have indices. We will not give their translation here.

3.4 Sheaves

It is time to dive into the horror of sheaves. We will follow the exposition of the reference book [MM92], quoting the relevant definitions and mention their precise source. The definitions are simply reformulated in SetTT with a type-theoretic flavour, and matched with the corresponding direct style concept through the presheaf intepretation.

Sieves [Section I.4]

> Given an object C in the category $\mathbf{C}$, a *sieve* on C (in French, a "crible" on C) is a set S of arrows with codomain C such that $f \in S$ and the composite fh is defined implies $fh \in S$.

Definition 10. A sieve on $p : \mathbb{P}$ is given as

- A predicate $S : \Pi(q:\mathbb{P})\,(\alpha : q \leq p).\,\mathtt{Prop}$ which corresponds to the set of arrows
- A proof $\theta_S : \Pi(q\,r:\mathbb{P})\,(\alpha : q \leq p)\,(\beta : r \leq q).\,S\ q\ \alpha \to S\ r\ (\beta \circ \alpha)$ which corresponds to closure under composition.

Note that we have merely made explicit the domain of the arrow in the predicate. It is implicit in the set-theoretic definition, but once again instead of using dependent types to define categories this book makes the choice to use partial functions instead.

Proposition 8. *A sieve on p is just a term of type $[\![\mathtt{Prop}]\!]_p$.*

Some constructions on sieves are also considered, which we rephrase as well.

Definition 11. The maximal sieve $\top_p : [\![\mathtt{Prop}]\!]_p$ is the constantly true function, which is trivially monotonous. It corresponds in the model to the true proposition, hence the notation.

Definition 12. Given a sieve $(S, \theta_S) : [\![\mathtt{Prop}]\!]_p$ and $\alpha : q \leq p$,

$$\alpha^*(S, \theta_S) : [\![\mathtt{Prop}]\!]_q := (\lambda(r : \mathbb{P})\,(\beta : r \leq q).\, S\, r\ (\beta \circ \alpha), \_)$$

Otherwise said, $\alpha^*(S, \theta_S)$ is just $\theta_{\mathtt{Prop}}\ \alpha\ (S, \theta_S)$.

For clarity purposes, in what follows we will implicitly coerce a sieve (S, θ_S) to its underlying predicate S.

Grothendieck topologies [Section III.2]

> A (Grothendieck) topology on a category $\mathbf{C}$ is a function J which assigns to each object C of $\mathbf{C}$ a collection $J(C)$ of sieves on C, in such a way that
>
> 1. the maximal sieve $t_C = \{f \mid \mathrm{cod}(f) = C\}$ is in $J(C)$;
> 2. (stability axiom) if $S \in J(C)$, then $h^*(S) \in J(D)$ for any arrow $h : D \to C$;
> 3. (transitivity axiom) if $S \in J(C)$ and R is any sieve on C such that $h^*(R) \in J(D)$ for all $h : D \to C$ in S, then $R \in J(C)$.

Definition 13. A Grothendieck topology is a function

$$J : \Pi(p : \mathbb{P}).\, [\![\mathtt{Prop}]\!]_p \to \mathtt{Prop}$$

together with proofs:

1. $\eta^J : \Pi(p : \mathbb{P}).\, J\ p\ \top_p$
2. $\mathtt{nat}^J : \Pi(p\, q : \mathbb{P})\,(\alpha : q \leq p)\,(S : [\![\mathtt{Prop}]\!]_p).\, J\ p\ S \to J\ q\ (\theta_{\mathtt{Prop}}\ \alpha\ S)$
3. $\gg\!=^J : \Pi(p : \mathbb{P})\,(S\, R : [\![\mathtt{Prop}]\!]_p).$
   $J\ p\ S \to (\Pi(q : \mathbb{P})\,(\alpha : q \leq p).\, S\ q\ \alpha \to J\ q\ (\theta_{\mathtt{Prop}}\ \alpha\ R)) \to J\ p\ R$

Exploiting the isomorphisms we have described before, it is clear that J and $\mathtt{nat}^J$ correspond to a term $\vdash_{\mathsf{PshTT}} J : \mathtt{Prop} \to \mathtt{Prop}$. Since all the types we are considering here are propositions, naturality conditions are trivially true. In particular,

$$\Pi(q : \mathbb{P})\,(\alpha : q \leq p).\, S\ q\ \alpha \to J\ q\ (\theta_{\mathtt{Prop}}\ \alpha\ S) \cong [\![S \to J\ R]\!]_p$$

so the type of $\gg\!=^J$ is isomorphic to

$$\mathsf{El}\ [\![\Pi(S\, R : \mathtt{Prop}).\, J\ S \to (S \to J\ R) \to J\ R]\!].$$

Similarly, the type of η^J is isomorphic to $\mathsf{El}\ [\![J\ \top]\!]$. The following theorem is now obvious.

Theorem 8. *Grothendieck topologies are the same as* `Prop`*-valued monads in the presheaf interpretation.*

Interestingly, even within the same book, this definition of Grothendieck topology is much closer to the monadic one than the definition of Lawvere-Tierney topologies. The major difference is that η^J is written with `propext` in mind. Meanwhile, Lawvere-Tierney topologies are defined in Section V.1 following the same extensional presentation as the nLab. I find this fascinating that the low-level notion fits more naturally in type theory while the topos-theoretic presentation hardwires some unnatural conventions because when you have an elementary topos everything looks like a monomorphism.

Matching families [Section III.4]

> If P is such a presheaf and the sieve S is a cover of an object C of $\mathbf{C}$, a *matching family* for S of elements of P is a function which assigns to each element $f : D \to C$ of S an element $x_f \in P(D)$, in such a way that
>
> $$x_f \cdot g = x_{fg} \qquad \text{for all } g : E \to D \text{ in } \mathbf{C}$$

First, note that through our interpretation a cover of $p : \mathbb{P}$ in the topology J is just a dependent pair of type $\Sigma S : [\![\texttt{Prop}]\!]_p.\ J\ p\ S$.

Definition 14. Let $(P, \theta_P) : \mathbf{Type}$, $(S, \theta_S) : [\![\texttt{Prop}]\!]_p$ and $i : J\ p\ S$. A matching family for S of elements of P is a function

$$x : \Pi(q : \mathbb{P})\,(\alpha : q \leq p).\ S\ q\ \alpha \to P\ q$$

together with a proof

$$\_ : \Pi(q\,r : \mathbb{P})\,(\alpha : q \leq p)\,(\beta : r \leq q)\,(e : S\ q\ \alpha).\ \theta_P\ \beta\ (x\ q\ \alpha\ e) = x\ r\ (\beta \circ \alpha)\ (\theta_S\ \beta\ e)$$

Despite the book definition leaving proof-irrelevant content from the e argument maliciously hidden, we cannot be fooled anymore. The side-condition is a naturality equation, so this is standing for a function.

Theorem 9. *Matching families for $S : [\![\texttt{Prop}]\!]_p$ of elements of P are isomorphic to $[\![S \to P]\!]_p$.*

The definition surreptitiously requires a proof $i : [\![J\ S]\!]_p$ by requiring a cover but it is actually not needed. As we will see it is a somewhat misplaced requirement for sheaves.

Amalgamation [Section III.4]

> An *amalgamation* of such a matching family is a single element $x \in P(C)$ with
>
> $$x \cdot f = x_f \qquad \text{for all } f \in S.$$

Definition 15. Let $\hat{x} : [\![S \to P]\!]_p$. An amalgamation of $\hat{x}$ is an element $x : [\![P]\!]_p$ s.t.

$$\Pi(q : \mathbb{P})\,(\alpha : q \leq p)\,(e : S\ q\ \alpha).\,\theta_P\ \alpha\ x = \hat{x}\ q\ \alpha\ e$$

The equation has a trivial direct style explanation through the presheaf interpretation.

Theorem 10. *An amalgamation of $\hat{x} : [\![S \to P]\!]_p$ is an element $x : [\![P]\!]_p$ s.t.*

$$[\![\Pi(e : S).\,x = \hat{x}\ e]\!]_p$$

Sheaves [Section III.4]

> Then P is a sheaf (for J) precisely then, when every matching family for any cover of any object of $\mathbf{C}$ has a unique amalgamation.

This is getting boring by now.

Definition 16. $(P, \theta_P) : \mathbf{Type}$ is a sheaf for J when there is an amalgamation candidate

$$h : \Pi(p : \mathbb{P})\,(S : [\![\mathtt{Prop}]\!]_p)\,(i : J\ p\ S).\,[\![S \to P]\!]_p \to [\![P]\!]_p$$

together with proofs that it is indeed an amalgamation and that it is unique. I am too lazy to spell out these equations but they are obtained directly using the previous expansions.

The S and i arguments correspond to the cover and the $[\![S \to P]\!]_p$ argument to the matching family. In any case, unicity of the amalgamation ensures the naturality of h, so it actually corresponds to a term

$$\vdash_{\mathsf{PshTT}} h : \Pi(S : \mathtt{Prop})\,(i : J\ S).\,(S \to P) \to P.$$

The amalgamation property is equivalent to a proof

$$\vdash_{\mathsf{PshTT}} \_ : \Pi(S : \mathtt{Prop})\,(i : J\ S)\,(\hat{x} : S \to P)\,(e : S).\,h\ S\ i\ \hat{x} =_P \hat{x}\ e.$$

Finally, unicity of the amalgamation is equivalent to a proof that

$$\begin{aligned}\vdash_{\mathsf{PshTT}} \_ :\ & \Pi(S : \mathtt{Prop})\,(i : J\ S)\,(\hat{x} : S \to P)\,(x : P).\\ & (\Pi(e : S).\,\hat{x}\ e = x) \to h\ S\ i\ \hat{x} =_P x.\end{aligned}$$

This justifies the claims from Section 1. Since we have a bit of leeway in the exact phrasing, we went for a slightly more compact definition there. Yet given that PshTT is a perfectly fine type theory, it is easy to check directly in it that the structure given in direct style above is isomorphic to the definitions from Section 1, gladly using extensionality principles.

Separated Presheaves [Section III.5]

> Call a presheaf P *separated* if a matching family can have *at most one* amalgamation. In other words, P is separated if for any $x, y \in P(C)$ and any cover $S \in J(C)$, $x \cdot f = y \cdot f$ for all $f \in S$ implies $x = y$.

Theorem 11. *A type* $\vdash_{\mathsf{PshTT}} P : \mathtt{Type}$ *is separated exactly when*

$$\vdash_{\mathsf{PshTT}} \_ : \Pi(S : \mathtt{Prop})\,(x\,y : P).\, J\ S \to (S \to x = y) \to x = y.$$

Since we are now skilled in presheaf-speak, it is easy to check on sight that the theorem is exactly the direct style formulation of the formulation in the model. We do not have to bother with naturality anywhere since all the functions considered are propositional.

Let us highlight that for our purposes the elementary reformulation is even better than the abstract unicity statement since with the former we do not need to quantify over matching families, i.e. functions of type $S \to P$, and simply work under the assumption S which amounts to considering the trivial matching families for two elements $x, y : P$. Obviously the equivalence between the two statements relies on funext but this is implicit in SetTT, and for that matter, ZFC.

Sheafification [Section III.5]

> Given an arbitrary presheaf P, we might therefore try to define a new presheaf P^+ by
>
> $$P^+(C) = \varinjlim_{R \in J(C)} \mathsf{Match}(R, P)$$
>
> where $\mathsf{Match}(R, P)$ denotes the set of matching families for the cover R of C, and the colimit is taken over all covering sieves of C, ordered by reverse inclusion. In other words, an element of $P^+(C)$ is an equivalence class of matching families
>
> $$\mathbf{x} = \{x_f \mid f : D \to C \in R\}, \quad x_f \in P(D), \quad \text{and} \quad x_f \cdot k = x_{fk},$$
>
> for all $k : E \to D$, where two such families $\mathbf{x} = \{x_f \mid f \in R\}$ and $\mathbf{y} = \{y_g \mid g \in S\}$ are equivalent when there is a common refinement $T \subseteq R \cap S$ with $T \in J(C)$ such that $x_f = y_f$ for all f in T.

We need quotients in SetTT, which we did not define. Since **Set** features them, it is legitimate to assume them in SetTT. Quotients are interpreted through the presheaf model in a similar way to inductive types, thus we readily get the following result.

Theorem 12. *Given a type* $\vdash_{\mathsf{PshTT}} P : \mathtt{Type}$*, P^+ is the quotient type*

$$\vdash_{\mathsf{PshTT}} P^+ : \mathtt{Type} := (\Sigma(S : \mathtt{Prop}).\, J\ S \times (S \to P))/\sim$$

where

$$\begin{aligned}(S, i, \hat{x}) \sim (R, j, \hat{y}) \quad := \quad & \exists(T : \mathtt{Prop})\,(\varphi : T \to S \wedge R). \\ & (J\ T) \wedge (\Pi(e : T).\, \hat{x}\ (\pi_1\ (\varphi\ e)) = \hat{y}\ (\pi_2\ (\varphi\ e)))\end{aligned}$$

I cannot stress how bad this construction is. The equivalence relation needs to land in `Prop`, so the existential over $T : \texttt{Prop}$ prominently featured in its definition is proof-irrelevant. This means that every time we will want to eliminate from this quotient to perfom some reasoning in `Type` we basically need to use a strong form of choice like the global choice function from Section 1.

Under such strong hypotheses, if P is already separated then P^+ is isomorphic to the QIT $\mathscr{S}_\jmath\ P$ described in Section 1, but the latter is much more well-behaved when one cares about computation, even assuming a very extensional ambient theory. Thanks to Santa Claus' choice, the isomorphism can be easily sketched:

- to $(S, i, \hat{x}) : P^+$ associate
$$\texttt{ask}\ S\ i\ (\lambda(e : S).\, \texttt{ret}\ (\hat{x}\ e)) : \mathscr{S}_\jmath\ P$$
- to $\texttt{ask}\ S_1\ i_1\ (\lambda(s_1 : S_1).\ \ldots (\texttt{ask}\ S_n\ i_n\ (\lambda(s_n : S_n).\, \texttt{ret}\ x))) : \mathscr{S}_\jmath\ P$ associate
$$(\exists(s_1 : S_1).\ \ldots\ (s_n : S_n).\, \top,\ \bigwedge_{k=1}^{n} i_k,\ \lambda(s_1, \ldots, s_n).\, x) : P^+$$
where $\bigwedge$ is defined using the monadic structure of J.

Separatedness of P is needed to ensure that the quotients are preserved. Needless to say that none of this makes sense without propext.

Elementary Sheafification [Section V.3] The definition of sheafification in the above section relies on having a somewhat rich type theory with quotients. There is an alternative definition for elementary topoi in Section V.3 of the reference book that boils down to handcrafting quotient types as done in set theory, i.e. by considering equivalence classes. For the sake of completeness we give here the definition in direct style in type theory. We will not give the relevant quotes as we did previously because the book takes great pleasure in hiding the computational content of their constructions under existential statements and diagrammatic equations. This is what you get when you have a deficient syntax to write your types, dear categorists.

Definition 17. The set of modal propositions is defined as
$$\texttt{Prop}_J := \{P : \texttt{Prop} \mid J\ P \to P\}.$$

Definition 18. Let $A : \texttt{Type}$. We define $A^+ : \texttt{Type}$ as
$$A^+ := \{P : A \to \texttt{Prop}_J \mid J\ (\exists(x : A).\, \Pi(y : A).\, \pi_1\ (P\ y) \leftrightarrow J\ (x = y))\}.$$

This type is isomorphic to the other A^+ type defined before. It is defined in an even worse way since now we have to explicitly pick elements out of equivalence classes instead of merely trying to preserve a (nasty) equivalence relation. Since we already assumed global choice to write combinators for the other one it does not really matter, but this one needs choice virtually everywhere.

Theorem 13. *For any $A : \texttt{Type}$, A^+ is separated. If furthemore A is already separated then A^+ is a sheaf.*

4 Kripke Semantics and Beth Semantics

Kripke and Beth semantics are similar models of intuitionistic logic. They can be understood as a model of the FOL fragment of respectively presheaves and sheaves. In particular, they are proof-irrelevant, i.e. apart from first-order variables, all proof terms live in sort Prop. This simplifies the model quite a bit. For Kripke models, this means that we do not have to care for naturality conditions on functions. For Beth models, this means we do not have to care for unicity of the amalgamation of a matching family. The former is not very important assuming we have UIP in our target theory, but the latter means that we can go for a much more direct model construction that does not rely on sheafification.

We now recall the interpretations of FOL for both models. Traditionally, rather than a category, Kripke and Beth models are defined w.r.t. a preorder $(\mathbb{P}, \leq)$. In our setting this means that morphisms have type $\mathtt{Hom}_{\mathbb{P}} : \mathbb{P} \to \mathbb{P} \to \mathtt{Prop}$, i.e. they are propositions. This does not matter in practice. All of what follows is folklore.

First, we define our FOL language. We fix a presheaf ι : **Type** for terms, together with atomic predicates $X : \mathsf{El}\ [\![\iota^n \to \mathtt{Prop}]\!]$. As usual we will silently rely on the isomorphisms with natural transformations. Note that morphisms $\alpha : q \leq p$ can act pointwise by θ_ι on a substitution $\rho : \mathsf{Var} \to [\![\iota]\!]_p$, which we write for simplicity $\alpha \bullet \rho : \mathsf{Var} \to [\![\iota]\!]_q$.

4.1 Kripke Semantics

Definition 19. Given $p : \mathbb{P}$, $\rho : \mathsf{Var} \to [\![\iota]\!]_p$ and A a FOL formula, we define the Kripke interpretation $p \Vdash_\rho A : \mathtt{Prop}$ in SetTT by induction on A.

$$
\begin{array}{lcl}
p \Vdash_\rho X(t_1, \ldots, t_n) & := & X\ p\ [t_1]_\rho\ \ldots\ [t_n]_\rho \\
p \Vdash_\rho A \to B & := & \forall(q : \mathbb{P})\,(\alpha : q \leq p).\, q \Vdash_{\alpha\bullet\rho} A \to q \Vdash_{\alpha\bullet\rho} B \\
p \Vdash_\rho \bot & := & \bot \\
p \Vdash_\rho A \vee B & := & (p \Vdash_\rho A) \vee (p \Vdash_\rho B) \\
p \Vdash_\rho \top & := & \top \\
p \Vdash_\rho A \wedge B & := & (p \Vdash_\rho A) \wedge (p \Vdash_\rho B) \\
p \Vdash_\rho \forall x.\, A & := & \forall(q : \mathbb{P})\,(\alpha : q \leq p)\,(x : [\![\iota]\!]_q).\, q \Vdash_{\alpha\bullet\rho, x} A \\
p \Vdash_\rho \exists x.\, A & := & \exists(x : [\![\iota]\!]_p).\, p \Vdash_{\rho,x} A
\end{array}
$$

In the above definition, we write $\forall(x : A).\, B$ to insist that $B : \mathtt{Prop}$, but these types are really dependent products.

Proposition 9. *As is well-known, the Kripke relation forms a proof-irrelevant presheaf, i.e.*

$$\Pi(p\, q : \mathbb{P})\,(\alpha : q \leq p).\, p \Vdash_\rho A \to q \Vdash_{\alpha\bullet\rho} A.$$

Proposition 10. *The Kripke model interprets intuitionistic* FOL*, i.e. if A is a closed* FOL *formula we have*

$$(\vdash_{\mathsf{iFOL}} A) \to \Pi(p : \mathbb{P}).\, p \Vdash_{(\cdot)} A.$$

Obviously Kripke models are the degenerate case of a presheaf interpretation where we only consider propositions, as we claimed above. Thanks to proof-irrelevance, we do not have to deal with naturality. Interestingly, we can also define it as an interpretation into `Type` *without naturality*. In this case we break conversion in the source theory, i.e. it is not a model of CIC anymore. Yet, it still makes sense from a programming point of view, since the resulting theory enjoys a *call-by-value* semantics, e.g. on function types conversion is generated by the equation

$$(\lambda(x : A).\, t)\ V \equiv_{\mathsf{cbv}} t\{x := V\}$$

where V is restricted to be a value. See my other note on the topic for more details.

4.2 Beth Semantics

Beth semantics are usually defined through the notion of *bar*. Bars are a staple notion from the philosophical school of intuitionistic mathematics. There are various formulations that are not equivalent without a sufficiently strong Fan Theorem, and a great deal of effort has been put in the understanding of the relationship between them. The standard notion of bars requires the underlying preorder to be a tree, i.e. it has a well-defined notion of successor of a node. Since I do not care about these Brouwerian shenanigans I will pick a variant that is simple and works for an arbitrary preorder, at the cost of being stronger than the historical definition without the right Fan Theorem.

Definition 20. We will conspicuously write

$$\mathsf{J}\ p\ P := \forall(q : \mathbb{P})\,(\alpha : q \leq p).\, \exists(r : \mathbb{P})\,(\beta : r \leq q).\, P\ r\ (\beta \circ \alpha)$$

for any $P : \Pi(q : \mathbb{P})\,(\alpha : q \leq p).\, \mathtt{Prop}$.

Intuitively, seeing the preorder as a DAG, $\mathsf{J}\ p\ P$ means that P is eventually true following all possible paths starting from p. Beth semantics is then given as the following variant of Kripke semantics.

Definition 21. Given $p : \mathbb{P}$, $\rho : \mathsf{Var} \to [\![\iota]\!]_p$ and A a FOL formula, we define the Beth interpretation $p \Vdash_\rho A : \mathtt{Prop}$ in SetTT by induction on A.

$$\begin{array}{lcl}
p \Vdash_\rho X(t_1, \ldots, t_n) & := & \mathsf{J}\ p\ (\lambda(q : \mathbb{P})\,(\alpha : q \leq p).\, X\ q\ [t_1]_\rho\ \ldots\ [t_n]_\rho) \\
p \Vdash_\rho A \to B & := & \forall(q : \mathbb{P})\,(\alpha : q \leq p).\, q \Vdash_{\alpha \bullet \rho} A \to q \Vdash_{\alpha \bullet \rho} B \\
p \Vdash_\rho \bot & := & \mathsf{J}\ p\ (\lambda(q : \mathbb{P})\,(\alpha : q \leq p).\, \bot) \\
p \Vdash_\rho A \vee B & := & \mathsf{J}\ p\ (\lambda(q : \mathbb{P})\,(\alpha : q \leq p).\, (q \Vdash_{\alpha \bullet \rho} A) \vee (q \Vdash_{\alpha \bullet \rho} B)) \\
p \Vdash_\rho \top & := & \top \\
p \Vdash_\rho A \wedge B & := & (p \Vdash_\rho A) \wedge (p \Vdash_\rho B) \\
p \Vdash_\rho \forall x.\, A & := & \forall(q : \mathbb{P})\,(\alpha : q \leq p)\,(x : [\![\iota]\!]_q).\, q \Vdash_{\alpha \bullet \rho, x} A \\
p \Vdash_\rho \exists x.\, A & := & \mathsf{J}\ p\ (\lambda(q : \mathbb{P})\,(\alpha : q \leq p).\, \exists(x : [\![\iota]\!]_q).\, q \Vdash_{\alpha \bullet \rho, x} A)
\end{array}$$

The same results as Krikpe hold for Beth.

Proposition 11. *The Beth relation forms a proof-irrelevant presheaf, i.e.*

$$\Pi(p\, q : \mathbb{P})\, (\alpha : q \leq p).\, p \Vdash_\rho A \to q \Vdash_{\alpha \bullet \rho} A.$$

Proposition 12. *The Beth model interprets intuitionistic* FOL*, i.e. if A is a closed* FOL *formula we have*

$$(\vdash_{\mathsf{iFOL}} A) \to \Pi(p : \mathbb{P}).\, p \Vdash_{(\cdot)} A.$$

Obviously something else is happening. I already spilled the beans, so guess what? Beth semantics is a (degenerate) sheaf model.

Proposition 13. J *is clearly a Lawvere-Tierney topology.*

Proposition 14. *The Beth relation is a* J*-algebra in the presheaf model, i.e. there is a proof*

$$\Pi(p : \mathbb{P}).\, \mathsf{J}\ p\ (\lambda(q : \mathbb{P})\, (\alpha : q \leq p).\, q \Vdash_{\alpha \bullet \rho} A) \to p \Vdash_\rho A$$

for every formula A.

The proof does not rely on anything else than J satsifying the topology axioms. Hence, the traditional definition in terms of bars would also go through immediately.

It is quite obvious that Beth models are a form of sheafification, but there are two major differences with the general case.

- All types considered are propositions.
- In particular, we only consider first-order logic quantifications.

This is why the interpretation is much simpler, we did not have to rely on a horrendous quotient inductive type. The only difference with Krikpe models lies in the fact that we freely added the J combinator for the $\bot, \vee, \exists$ and atomic cases. It is therefore immediate to anybody who knows about programming language theory that this corresponds to a call-by-name embedding of the monad J, i.e. a mapping into the Eilenberg-Moore category. This observation leads to a synthetic presentation of Beth models.

Theorem 14. *Given a preorder $\mathbb{P}$ and a* `Prop`*-valued monad J in the presheaf intepretation over $\mathbb{P}$, Beth models are given synthetically as the composition*

$$\mathfrak{B} \xrightarrow{\mathsf{J}^N} \mathfrak{K} \xrightarrow{\mathsf{Psh}(\mathbb{P})} \mathbf{Set}$$

where

- $\mathfrak{B}$ *is the Beth model*
- $\mathfrak{K}$ *is the Kripke model*
- J^N *is the call-by-name embedding of the monad* J
- $\mathsf{Psh}(\mathbb{P})$ *is the (degenerate) presheaf interpretation.*

All the models above are degenerate from a computational point of view, since they are all preorders dealing with provability rather than actual proofs.

5 An Intensional Analysis of Sheaves

We look at the abstract definitions of Section 1 with a eye towards computation.

5.1 Dialogue Trees

We first highlight the relationship of sheaves with a structure known as dialogue trees.

Definition 22. Given a input type $I : \mathtt{Type}$ and an output type $O : \Pi(i : I).\, \mathtt{Type}$, the type of dialogue trees is defined as the inductive type

$$\begin{aligned}&\mathtt{Inductive}\ \mathscr{D}\ (A : \mathtt{Type}) : \mathtt{Type} :=\\ &\mid \mathtt{ret} : \Pi(x : A).\, \mathscr{D}\ A\\ &\mid \mathtt{ask} : \Pi(i : I).\, (O\ i \to \mathscr{D}\ A) \to \mathscr{D}\ A.\end{aligned}$$

Dialogue trees form a monad, and a free one at that. A dialogue tree can be seen as a way to implement an oracle, where I is the type of questions and O the type of answers. While `ret` means that enough information is available to produce a value, `ask` corresponds to a question to the oracle, and the function to the continuation to apply when the oracle has answered.

Since $\mathscr{D}$ is a free monad, it is somewhat easy to give a syntactic model of Baclofen Type Theory [PT17] where all types are $\mathscr{D}$-algebras through the weaning interpretation. Indeed, there is a straightforward description of $\mathscr{D}$-algebras that is compatible with intensionality, namely

$$\mathbf{Type}^{\mathscr{D}} := \{A : \mathtt{Type} \mid \Pi(i : I).\, (O\ i \to A) \to A\}.$$

We will call a function $\Pi(i : I).\, (O\ i \to A) \to A$ an oracle function for A. Assuming funext, $\mathbf{Type}^{\mathscr{D}}$ is isomorphic to the usual definition of $\mathscr{D}$-algebras. Furthermore, even without funext, $\mathbf{Type}^{\mathscr{D}}$ is closed under dependent products, in the sense that if $A : \mathtt{Type}$ and $B : A \to \mathbf{Type}^{\mathscr{D}}$ then $\Pi(x : A).\, \pi_1\ (B\ x)$ has an oracle function defined pointwise. This allows to easily define the weaning translation, which can be described as a high-level point of view by replacing `Type` with $\mathbf{Type}^{\mathscr{D}}$ everywhere. Closure by dependent product codomain allows to lift everything to an arbitrary context. Inductive types are freely turned into $\mathscr{D}$-algebras by adding a constructor for the oracle function, e.g. booleans are interpreted as

$$\begin{aligned}&\mathtt{Inductive}\ \mathtt{bool}^{\mathscr{D}} : \mathtt{Type} :=\\ &\mid \mathtt{true}^{\mathscr{D}} : \mathtt{bool}^{\mathscr{D}}\\ &\mid \mathtt{false}^{\mathscr{D}} : \mathtt{bool}^{\mathscr{D}}\\ &\mid \mathtt{ask}_{\mathtt{bool}} : \Pi(i : I).\, (O\ i \to \mathtt{bool}^{\mathscr{D}}) \to \mathtt{bool}^{\mathscr{D}}.\end{aligned}$$

The crux of the weaning translation is to equip $\mathbf{Type}^{\mathscr{D}}$ with an oracle function. The simplest way to do that is also to turn it into a free algebra as well, which incurs some technicality w.r.t. Tarski universes but can be performed nonetheless. See the aforementioned paper for the full construction.

Theorem 15. *The weaning translation provides a model of* BTT *where types are interpreted as intensional $\mathscr{D}$-algebras. In particular this model can be defined in vanilla* CIC*, no need for any extensionality axiom.*

The main difference between BTT and CIC is that the former only satisfies a restricted form of dependent elimination, where the elimination predicate needs to be *linear*, i.e. strict. Failure of unrestricted dependent elimination is caused by the existence of non-standard or effectful inductive terms, similarly to what happens in effectful call-by-name languages. For instance, non-termination forces to interpret booleans as $\mathbb{B}_\perp$ in Scott domains, which has three different values classically.

We emphasize that BTT is stronger than HOL, since it has universes. Furthermore in presence of the extensionality rule, CIC can be embedded into BTT in a way similar to the embedding of LK into LJ, since there is a canonical way to linearize a predicate which is (propositionally) the identity in CIC.

5.2 Sheaves as Dialogue Trees

It is clear from Definition 2 that sheaves are a special case of $\mathscr{D}$-algebras. Indeed, defining

$$\begin{array}{lcl} I & := & \Sigma P : \mathtt{Prop}.\, \mathsf{J}\ P \\ O\ (P, i) & := & P \end{array}$$

it becomes apparent that sheaves are an instance of dialogue trees where $O : I \to \mathtt{Prop}$ is a propositional predicate, and thus in particular proof-irrelevant, rather than living in `Type`.

Assuming funext and UIP, dialogue trees are very simple in this degenerate setting. Since there is at most one possible answer to the query, the branching structure collapses, and dialogue trees are isomorphic to a list of queries to the oracle.

With these notations, a sheaf is just a type A equipped with an oracle function

$$\mathtt{ask}_A : \Pi(i : I).\, (O\ i \to A) \to A$$

satisfying in addition the uniqueness axiom

$$\sigma_A : \Pi(i : I)\, (x : A).\, \mathtt{ask}_A\ i\ (\lambda(p : O\ i).\, x) = x.$$

The function corresponds to A being a $\mathscr{D}$-algebra, while uniqueness further mandates that a list of queries that returns a value is not distinguishible from that value. In some sense, it hides those queries from the outside, as if they were some kind of silent transitions. Hence, $\mathscr{S}_\mathsf{J}$-algebras are a special kind of quotiented $\mathscr{D}$-algebras.

This requirement is highly problematic in an intensional setting, because it is stated as an equality. In particular, without funext sheaves are not closed under product codomain, since it results in an equality between two functions that are not definitionally equal. Contrarily to $\mathscr{D}$-algebras, I currently do not know of a good intensional definition of $\mathscr{S}_\mathsf{J}$-algebras, and I actually doubt there is any.

The same issue arises with inductively generated types. They need to be interpreted as free $\mathscr{S}_\mathsf{J}$-algebras, but this requires the existence of QITs in the ambient theory. As is folklore with any generic construction that remotely relies on propositional equality, there are only two stable solutions to make quotients work: either require UIP or go full univalence. As a hardcore compatibilist, I do not like either one, although if I had to choose I would go for UIP.

It is actually even worse than that, since in general the type of sheaves is not a sheaf itself, just as the type of algebras is not an algebra. Thus there are no universes in sheaves. In particular, contrarily to BTT it is not even possible to *state* dependent elimination there. What is even the point to preserve all that structure, for Martin-Löf's sake?

This is actually a well-known problem, and the standard solution seems to generalize sheaves to *stacks* [CMR17]. I have zero understanding of what a stack is, although what I understand from the cited article seems to indicate it is the natural thing to do, i.e. relax the equality in the uniqueness condition into higher path types. But then it looks like we are back to requiring univalence in the ambient theory, which is a no-no.

The only sane solution I can think of is the laxification trick, namely replacing this equality by a proof-relevant parametricity-like predicate. This is a poor man's univalence, a.k.a. Sattler's obfuscation by syntax. Proof that it works left as an exercise to the reader.

5.3 Computational Sheaves

Thierry Coquand is the co-author of a series of interesting papers on sheaf models of systems that grow increasingly closer to dependent type theories. Due to the choice of concrete sites for the sheaf construction, the definition of covers always follow a similar structure that closely matches the inductive dialogue tree definition. Typically, the notion of covering is defined as an inductive tree with two cases, either a leaf or a node that performs a step in the DAG followed by subtrees. The monadic structure arises then automatically from the dialogue structure, which was actually already the case for tree in direct style.

Geometrical Logic The paper [Coq05] provides a proof of completeness of *dynamical proofs* for geometric theories. A bit of squinting shows that the very notion of dynamical proof is literally a dialogue tree, and that covers can be defined as a dialogue tree directly *inside* the presheaf model. For simplicity, if we only consider propositional logic, a geometric theory is a set of formulae

$$\mathcal{T} := \left\{ \Phi_i \vdash \bigvee_{k \in O_i} \Psi_{i,k} \right\}_{i \in I}$$

where Φ, Ψ_k are conjunctions of atoms. The base category considered has objects finite sets of atomic formulae, and order given by reverse set inclusion. The covering definition $X \lhd \mathcal{Y}$ between an object X and a set of objects $\mathcal{Y}$ is inductively defined as

$$\frac{}{X \lhd \{X\}} \qquad \frac{(\Phi \vdash \bigvee_{k \in O} \Psi_k) \in \mathcal{T} \qquad X \Vdash \Phi \qquad (\forall k \in O. \quad X, \Psi_k \lhd \mathcal{Y}_k)}{X \lhd \bigcup_{k \in O} \mathcal{Y}_k}$$

which mimicks the dialogue tree inductive definition. In this simplified setting, a query is a formula $(\Phi \vdash \bigvee_{k \in O} \Psi_k) \in \mathcal{T}$ together with a proof that Φ holds in the current state, and an answer is a proof that one of the disjuncts Ψ_k holds. It is clear that the set of answers depends on the query since the size of the disjunction depends on the axiom of the theory under consideration. Similarly, the higher-order nature of the continuation for the query is obvious when written as above. Handling FOL is a simple generalization of the above.

The theorem proved in the paper is confusingly labelled completeness despite being a proof of soundness of the corresponding Beth model. We highlight the fact that axioms from the geometrical theory are interpreted as queries to the oracle, resulting in the dynamical structure of the proof tree since it is unknown a priori how these axioms are implemented. This is a clever example of side-effects reminiscent of thread forking.

Cohen Reals The papers [CJ12] and [CM17] give a sheaf model of a barebone dependent type theory, which is shown to negate Markov's Principle in the second one. The presheaf model is given over the preorder of finitely supported partial functions $\mathbb{N} \to \mathbb{B}$ ordered by reverse inclusion. For simplicity, we explain it here instead in terms of lists of booleans, where $q \leq p$ if p is a suffix of q. Covering of $p : \mathtt{list}\ \mathbb{B}$ by a set of lists $\mathcal{L}$ is inductively defined as

$$\frac{}{p \lhd \{p\}} \qquad \frac{\mathtt{tt} :: p \lhd \mathcal{L}_{\mathtt{tt}} \qquad \mathtt{ff} :: p \lhd \mathcal{L}_{\mathtt{ff}}}{p \lhd \mathcal{L}_{\mathtt{tt}} \cup \mathcal{L}_{\mathtt{ff}}}$$

Here, because the set of answers is finite it is less clear that this is a dialogue tree, as it is given as what amounts to a binary tree. But by exploiting the fact that $P\ \mathtt{tt} \times P\ \mathtt{ff} \cong \Pi(b : \mathbb{P}).\ P\ b$ the dialoguing nature of this covering relation appears.

Algebraic Closure The paper [MC14] gives a constructive sheaf model of the algeraic closure of a field. The presheaf model is given over a special kind of $\mathbb{K}$-algebras for some field $\mathbb{K}$. Covering of an algebra A is once again inductively defined as a more concrete form of

$$\frac{}{A \lhd \{A\}} \qquad \frac{a \in A \qquad A/\langle a \rangle \lhd \mathcal{L} \qquad A\left[\tfrac{1}{a}\right] \lhd \mathcal{R}}{A \lhd \mathcal{L} \cup \mathcal{R}}$$

so that the queries to our oracle are given as an element a of the current global algebra, and the oracle decides whether this element is irreducible.

5.4 Partiality Monads

For completeness, we mention the fact that synthetic sheaves are strikingly similar to the various kinds of partiality monads described e.g. by Uustalu et al. [UV17]. These monads are known as lifting monads in the category theory literature.

Partiality monads are given by a set of truth values $\Omega : \mathtt{Type}$ together with a function $\mathsf{El} : \Omega \to \mathtt{Prop}$ interpreting it as a proposition. The partiality monad is then defined as

$$\mathfrak{P}\ A := \Sigma P : \Omega.\, \mathsf{El}\ \Omega \to A.$$

They give three representative examples.

- The error monad $\Omega := \mathbb{B}$ and $\mathsf{El}\ b := b = \mathtt{tt}$. In this case $\mathfrak{P}\ A \cong 1 + A$.
- The non-termination monad $\Omega := \mathsf{S}$ and $\mathsf{El}\ s := s \sim \top$ where S is the Sierpinski set or Rosolini's dominance.
- The full partial map classifier $\Omega := \mathtt{Prop}$ and $\mathsf{El}\ P := P$

It is remarkable how these monads are similar to sheafification, up to the quotient construction that ensures the uniqueness property.

References

[Acz98] Peter Aczel. "On Relating Type Theories and Set Theories". In: *Types for Proofs and Programs, International Workshop TYPES '98, Kloster Irsee, Germany, March 27-31, 1998, Selected Papers*. Ed. by Thorsten Altenkirch, Wolfgang Naraschewski, and Bernhard Reus. Vol. 1657. Lecture Notes in Computer Science. Springer, 1998, pp. 1–18. DOI: 10.1007/3-540-48167-2_1. URL: https://doi.org/10.1007/3-540-48167-2%5C_1.

[CJ12] Thierry Coquand and Guilhem Jaber. "A Computational Interpretation of Forcing in Type Theory". In: *Epistemology versus Ontology - Essays on the Philosophy and Foundations of Mathematics in Honour of Per Martin-Löf*. Ed. by Peter Dybjer et al. Vol. 27. Logic, Epistemology, and the Unity of Science. Springer, 2012, pp. 203–213. DOI: 10.1007/978-94-007-4435-6_10. URL: https://doi.org/10.1007/978-94-007-4435-6%5C_10.

[CM17] Thierry Coquand and Bassel Mannaa. "The Independence of Markov's Principle in Type Theory". In: *Log. Methods Comput. Sci.* 13.3 (2017). DOI: 10.23638/LMCS-13(3:10)2017. URL: https://doi.org/10.23638/LMCS-13(3:10)2017.

[CMR17] Thierry Coquand, Bassel Mannaa, and Fabian Ruch. "Stack semantics of type theory". In: *32nd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2017, Reykjavik, Iceland, June 20-23, 2017*. IEEE Computer Society, 2017, pp. 1–11. DOI: 10.1109/LICS.2017.8005130. URL: https://doi.org/10.1109/LICS.2017.8005130.

[Coq05] Thierry Coquand. "A Completeness Proof for Geometrical Logic". In: *Logic, Methodology and Philosophy of Sciences*. 2005. URL: http://www.cse.chalmers.se/~coquand/site.pdf.

[Jab+16] Guilhem Jaber et al. “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*. Ed. by Martin Grohe, Eric Koskinen, and Natarajan Shankar. ACM, 2016, pp. 367–376. DOI: 10.1145/2933575.2935320. URL: https://doi.org/10.1145/2933575.2935320.

[JTS12] Guilhem Jaber, Nicolas Tabareau, and Matthieu Sozeau. “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, 2012, pp. 395–404. DOI: 10.1109/LICS.2012.49. URL: https://doi.org/10.1109/LICS.2012.49.

[MC14] Bassel Mannaa and Thierry Coquand. “A Sheaf Model of the Algebraic Closure”. In: *Proceedings Fifth International Workshop on Classical Logic and Computation, CL&C 2014, Vienna, Austria, July 13, 2014*. Ed. by Paulo Oliva. Vol. 164. EPTCS. 2014, pp. 18–32. DOI: 10.4204/EPTCS.164.2.

[MM92] Saunders MacLane and Ieke Moerdijk. *Sheaves in geometry and logic: a first introduction to topos theory*. Universitext. Berlin: Springer, 1992. DOI: 10.1007/978-1-4612-0927-0. URL: https://cds.cern.ch/record/824105.

[Péd20] Pierre-Marie Pédrot. “Russian Constructivism in a Prefascist Theory”. In: *LICS '20: 35th Annual ACM/IEEE Symposium on Logic in Computer Science, Saarbrücken, Germany, July 8-11, 2020*. Ed. by Holger Hermanns et al. ACM, 2020, pp. 782–794. DOI: 10.1145/3373718.3394740. URL: https://doi.org/10.1145/3373718.3394740.

[PT17] Pierre-Marie Pédrot and Nicolas Tabareau. “An effectful way to eliminate addiction to dependence”. In: *32nd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2017, Reykjavik, Iceland, June 20-23, 2017*. IEEE Computer Society, 2017, pp. 1–12. DOI: 10.1109/LICS.2017.8005113. URL: https://doi.org/10.1109/LICS.2017.8005113.

[RSS20] Egbert Rijke, Michael Shulman, and Bas Spitters. “Modalities in homotopy type theory”. In: *Logical Methods in Computer Science* Volume 16, Issue 1 (Jan. 2020). DOI: 10.23638/LMCS-16(1:2)2020. URL: https://lmcs.episciences.org/6015.

[UV17] Tarmo Uustalu and Niccolò Veltri. “Partiality and Container Monads”. In: *Programming Languages and Systems, APLAS 2017, Suzhou, China, November 27-29, 2017*. Ed. by Bor-Yuh Evan Chang. Vol. 10695. Lecture Notes in Computer Science. Springer, 2017, pp. 406–425. DOI: 10.1007/978-3-319-71237-6_20.