Kripke-Joyal Forcing as a Modality in Lean 4

Wojciech Nawrocki

(This note describes ongoing work. The most up-to-date version can be found at voidma.in/phd_proposal.pdf.)

April 25, 2024

Contents

Abstract

Propositions about categories are sometimes better viewed from a *synthetic* perspective, namely as statements in an internal language $\mathcal{L}$ that axiomatizes the relevant domain of inquiry. A well-designed internal language can clarify the essence of a class of problems, and constructions carried out therein can be more general (by applying to all the models of $\mathcal{L}$). However, unfolding an internal statement into its *external* interpretation is often burdensome on paper, at least because interpretation functions can be highly non-trivial. This presents a barrier to the adoption of synthetic approaches. Our working hypothesis is that a proof assistant can take care of this bureaucracy, granting the user more confidence in their results.

To date, no framework mixing internal and external reasoning in toposes has been implemented in a proof assistant. I propose to carry out a thesis project developing the theory and implementation of an embedding of *Kripke-Joyal forcing* in Lean 4. Kripke-Joyal forcing provides a convenient notation for the interpretation of higher-order logic in a (1-)topos, useful in particular for reasoning about *local* data.

Jointly with Gabriel Ebner I present preliminary work searching for an embedding of Kripke-Joyal notation in an interactive theorem prover. A design goal of the embedding is to be *ergonomic* and feel natural to users of proof assistants based on dependent type theory (for instance by allowing the use of existing tactics while reasoning internally). To this end, we formulate Kripke-Joyal notation as a *modality* in the host type theory. We view the added modality as a conservative extension of the original language which ought to be *eliminable* by means of the interpretation function.

1. Background

> If you wish to make an apple pie from scratch, you must first invent the universe.
>
> — Carl Sagan

In this chapter I motivate the fragments of mathematics and computer science that are relevant to the proposed project. In Section 1.1 I motivate *synthetic mathematics*, a collection of proof techniques that this project aims to contribute to. My proposed approach mechanizes Kripke-Joyal forcing notation as a *modality* in the host type theory. Modal type theory, and systems of natural deduction for it, are motivated in Section 1.2. Next, we move on to recall a few properties of elementary toposes in Section 1.3. Finally, the interpretation of higher-order logic in elementary toposes and its formulation as Kripke-Joyal forcing is sketched in Section 1.4. Some background in proof theory (e.g. Avigad [2]) and category theory (e.g. Awodey [3] or Riehl [63]) is assumed.

Notation for category theory.

- The collection of objects in a category $\mathcal{C}$ is written $\mathrm{Ob}(\mathcal{C})$.
- For $A, B \in \mathrm{Ob}(\mathcal{C})$, the collection of arrows $A \to B$ is written $\mathcal{C}(A, B)$.
- Composition of arrows is written *in-order* using ';', as in $f; g = X \xrightarrow{\mathrm{f}} Y \xrightarrow{\mathrm{g}} Z$.

1.1. Synthetic mathematics

The distinction between **synthesis** and **analysis** has a long and confounding history in philosophy.

Luckily, in mathematical contexts the two terms have been used in a fairly consistent manner, reflecting roughly the distinction between an axiomatic theory and its models.[1] The dichotomy can be formalized in multiple ways, depending on a choice of what sort of thing counts as an axiomatization and what counts as a model.

Consider geometry. In Euclid's foundational text, a point is defined as "that which has position, but not magnitude" and a line as "length without breadth" [15]. These definitions are subsequently never used: all theorems are proven, and constructions made, purely by reference to the basic axioms and previously established facts. Points and lines are undefined *things* that obey certain relations, the attempted definitions notwithstanding.

An early attempt at making this precise is found in Hilbert's *Foundations of Geometry* [34]. In modern terms, we could say that the book declares points, lines, and planes to be the sorts of a multisorted second-order theory, and specifies a family of five sets of axioms (on incidence, ordering, congruence, parallels, completeness). Models are then constructed for various subfamilies, demonstrating for instance that the axioms of continuity do not follow from the other ones, being refuted by non-Archimedean geometries.

In Hilbert's setting, one could call a definition, theorem statement, or proof *synthetic* when it can be stated or proven using just the available sorts and reasoning from geometric axioms. On

[1] It can be useful to coopt an existing word to serve as a piece of mathematical terminology while making no claim as to whether such a formalization fully agrees with the original concept, so long as the word vaguely reminds us of said concept and evokes some intuition: consider "measure", "continuous", "bundle", or "Hegelian taco".

the other hand, the construction of a model, or any statement that refers to such a construction, is *analytic*. When it comes to defining the real numbers, Hilbert explicitly advocated for the axiomatic approach [22], stating that "for the final presentation and the complete logical grounding of our knowledge the axiomatic method deserves the first rank". This perspective is now standard: textbooks in real analysis (e.g. Royden [64]) commonly begin by postulating the existence of a complete, ordered field. The postulate is justified by a categoricity result: up to isomorphism there is only one such field; that it can be constructed out of Dedekind cuts or Cauchy sequences is a mere existence result that fades into the background.

These historical developments illustrate two facts:

- Working synthetically can restrict our domain of discourse to the relevant objects—points, lines, and planes, or numbers—forbidding any reference to irrelevant entities such as various sets that might exist in a particular model construction, and contingent features thereof.
- Synthetic proofs can be more general, for instance by applying to many distinct geometries.

1.2. Modalities

In this section I briefly review historical developments in modal logic, up until its modern constructive interpretations in dependent type theories supporting multiple *modes* and modalities. Along the way, I point out features and misfeatures of the encountered formal systems with relevance to the task of embedding internal language reasoning. I draw partly on Ballarin's article [6] concerning the history of modal logic, as well as on a survey of modal λ-calculi due to Kavvos [38].

1.2.1. Modal logic. The study of modal logic grew out of attempts to formalize the meaning of certain kinds of linguistic expressions. Consider the sentence "it may snow tomorrow". It is not saying that "it will snow tomorrow", nor that "it won't snow tomorrow", and (depending on how one interprets this particular utterance) not even that "it will either snow or not snow tomorrow". In saying that something "may" happen, a speaker communicates that they consider specific futures *possible*. In contrast, the sentence "there is no way the Steelers will win this weekend" expresses the exclusion of some futures from the realm of possibility.

Pioneered by MacColl [48] and Lewis [45], logical analyses of utterances of this form (and of *material implication*) eventually settled on a syntax including propositional operators $\Diamond$ and $\Box$, with $\Diamond\varphi$ meaning "possibly φ" and $\Box\varphi$ (classically defined as $\Box\varphi \triangleq \neg\Diamond\neg\varphi$) meaning "necessarily φ".

While initially focused on necessity and possibility (the *alethic* modalities), modal logic quickly outgrew its origins. In the 1930s, Gödel [23] observed that a provability modality in intuitionistic logic behaves identically to one of Lewis's systems.

Since then, modal logic has been productively applied to a variety of epistemic, mathematical, computational, and semantic phenomena including in the formal verification of distributed [43] and cyber-physical systems [60]. In metamathematics, Hamkins and Löwe [31] have investigated logics with modal operators denoting possibility/necessity in some/all set-theoretic forcing extensions.

1.2.2. Frame semantics and truth-functionality. The proliferation of modalities across multiple disciplines brings forth a question: do they all have something in common, or are

we lumping too many objects under the modal umbrella? One answer is that semantically, they are all *non truth-functional*: evaluating the truth of a general modal sentence $\Box\varphi$ requires additional information beyond the truth value of φ.

Let's see this in some detail. The most well-adopted semantics for modal logic is based on relational structures called *frames* [12]. A *frame* (often called a *Kripke frame* after a particularly prominent proposal for a semantics [41]) is a suggestive name for a directed graph (W, R) consisting of a set W of *worlds* together with a relation $R \subseteq W \times W$. A *model* $\mathfrak{M}$ consists of a frame together with a valuation $V : \mathrm{Atom} \to \mathcal{P}(W)$ sending each propositional atom to the set of worlds where it is true.

A formula φ is *true at a world* w *in* $\mathfrak{M}$, written $\mathfrak{M}, w \vDash \varphi$, under the following conditions:

- if $\varphi = p$ is an atom, $\mathfrak{M}, w \vDash p$ when $p \in V(w)$
- $\mathfrak{M}, w \vDash \varphi \wedge \psi$ when $\mathfrak{M}, w \vDash \varphi$ and $\mathfrak{M}, w \vDash \psi$
- $\mathfrak{M}, w \vDash \varphi \vee \psi$ when $\mathfrak{M}, w \vDash \varphi$ or $\mathfrak{M}, w \vDash \psi$
- $\mathfrak{M}, w \vDash \neg\varphi$ when $\mathfrak{M}, w \nvDash \varphi$
- $\mathfrak{M}, w \vDash \Box\varphi$ when for every v with wRv, $\mathfrak{M}, v \vDash \varphi$

The intended meaning of $\Box$ is determined by how we interpret R:

- if wRv means that an agent living in world w considers the world v to be possible, then $\Box$ means "necessarily": in this picture, a proposition is necessary iff it's true in every conceivable world;
- if R is the reachability relation in an automaton, then $\Box$ means "in every next state"; and so on.

Now observe that the truth of standard connectives $\wedge$, $\vee$ and $\neg$ is simply a function of the truth(s) of nested formulas; but this is not so for $\Box$ which quantifies over possible worlds. This is what it means for $\Box$ to be non truth-functional.

The concept of truth-functionality is only heuristic (what about a semantic interpretation with no concept of boolean truth value?), and does not tell the full story. Above, we only considered propositional logic. Quantification over a first-order domain is also non truth-functional. Indeed, quantifiers can be viewed as modal operators [51]. Yet there is also good reason to exclude them: propositional modal logics are commonly decidable (there exists a procedure to determine whether a given sentence is provable), whereas first-order logic is not [68]. Many researchers view having good computational properties as the defining characteristic of modal logics [12], thus excluding first-order logic.

Despite these ambiguities, truth-functionality generalizes more straightforwardly to the setting of modal type theory. In category-theoretic models of dependent types an analogous property of being *non-fibered* can be said to hold of type formers that enact a base change as opposed to computing within a single slice category. In contrast, not much can be said about decidability of type inhabitation in modal type theories, most of them being higher-order and undecidable.

1.2.3. From logic to type theory. To enable viewing modal propositions as the types of their proofs, modal logic has to be endowed with an assignment of proof terms in some correspondence with derivations of the logic.

When formulated as a sequent calculus, a basic requirement for a logic is to admit cut elimination so that proofs can be normalized, and the normal forms inspected to determine further properties. Indeed, modal sequent calculi such as in the early work of Ohnishi and Matsumoto [55] admit cut elimination. Here and afterwards, $\Box\Gamma$ denotes a context where all hypotheses are of the form $\Box A$ for some A.

$$\frac{\Box\Gamma \longrightarrow A}{\Box\Gamma \longrightarrow \Box A}\Box R \quad \frac{\Gamma, A \longrightarrow \Theta}{\Gamma, \Box A \longrightarrow \Theta}\Box L$$

$$\frac{\Gamma \longrightarrow \Theta, A}{\Gamma \longrightarrow \Theta, \Diamond A}\Diamond R \quad \frac{A \longrightarrow \Diamond\Gamma}{\Diamond A \longrightarrow \Diamond\Gamma}\Diamond L$$

Figure 1: Inference rules for necessity and possibility in Ohnishi and Matsumoto's [55] sequent calculus for the classical modal logic S4.

However, though sequent calculi have their own computational interpretations, to obtain a λ-calculus we generally seek a system of *natural deduction*. Rather than having rules operate on the left and right of $\longrightarrow$ as in sequent calculi, in natural deduction we have elimination and introduction rules for logical connectives. For modal logic, identifying a system of natural deduction turns out to be more difficult than one might initially expect. Consider the following seemingly canonical attempt with a term assignment, function types, and two rules for a necessity modality:

$$\frac{\Gamma, x : A \vdash t(x) : B}{\Gamma \vdash \lambda(x : A).\ t(x) : A \to B}{\to}I \quad \frac{\Gamma \vdash t : A \to B \qquad \Gamma \vdash u : A}{\Gamma \vdash t\ u : B}{\to}E$$

$$\frac{\Box\Gamma \vdash t : A}{\Box\Gamma \vdash \text{box}\ t : \Box A}\Box I \quad \frac{\Gamma \vdash t : \Box A}{\Gamma \vdash \text{unbox}\ t : A}\Box E$$

As already observed by Prawitz in 1965 [61], such a system *fails* to admit substitution. The derivation of $f\ a$ below cannot be substituted for x in box x because the context $A \to \Box B, A$ is not in the image of $\Box$; the term box $(f\ a)$ is not well-typed.

$$\frac{\dfrac{\dfrac{}{f, a \vdash f : A \to \Box B} \qquad \dfrac{}{f, a \vdash a : A}}{f : A \to \Box B, a : A \vdash f\ a : \Box B} \qquad \dfrac{\dfrac{}{x \vdash x : \Box B}}{x : \Box B \vdash \text{box}\ x : \Box\Box B}}{?}$$

$$f : A \to \Box B, a : A \vdash (\text{box}\ x)[f\ a/x] : \Box\Box B$$

Extensionality (η-expansion) of derivations of $\Box B$ fails in a very similar way: an arbitrary $\Gamma \vdash t : \Box A$ cannot be identified with $\Gamma \vdash \text{box}\ (\text{unbox}\ t) : \Box A$. After pointing out the above issue, Prawitz [61], and more recently Benton, Bierman, de Paiva and Hyland [9], offer a candidate solution: generalize the introduction rule to one that essentially builds in simultaneous substitutions.

$$\frac{\Gamma \vdash t_1 : \Box A_1 \qquad \cdots \qquad \Gamma \vdash t_n : \Box A_n \qquad x_1 : \Box A_1, \ldots, x_n : \Box A_n \vdash u : C}{\Gamma \vdash \text{let}\ x_1, \ldots, x_n = t_1, \ldots, t_n\ \text{in box}\ u : \Box C}\Box I$$

Substitution is shown to hold for the system of delayed substitutions: if $\Gamma \vdash a : A$ and $\Gamma, x : A \vdash b : B$, then $\Gamma \vdash b[a/x] : B$, and type preservation in the normalization relation follows.

1.2.4. Split contexts. A slightly different solution is identified by Davies and Pfenning [21] following Girard [29] and Andreoli [1]. In a system they called $\lambda_e^{\to\Box}$, a well-formed context is split into two sections: $\boxed{\vdash \Gamma; \Delta \text{ ctx}}$. Hypothetical logical judgments $A_1, ..., A_n; B_1, ..., B_m \vdash C$ are read as "if $A_1, ..., A_n$ are necessary, and $B_1, ..., B_m$ are true, then C is true". To express the modal law of necessitation, one notes that necessary conclusions can follow from necessary assumptions, but not assumptions of contingent truth, so that "if $A_1, ..., A_n$ are necessary, then C is necessary" can be written as $A_1, ..., A_n; \cdot \vdash C$. This is codified in the form of an introduction rule $\Box I$ that (read upwards) clears the truth context.

$$\frac{}{\Gamma; \Delta, x : A \vdash x : A}\,\text{Var} \qquad \frac{}{\Gamma, x :: A; \Delta \vdash x : A}\,\Box\text{Var}$$

$$\frac{\Gamma; \cdot \vdash t : A}{\Gamma; \Delta \vdash \text{box } t : \Box A}\,\Box I \qquad \frac{\Gamma; \Delta \vdash t : \Box A \qquad \Gamma, x :: A; \Delta \vdash u : B}{\Gamma; \Delta \vdash \text{let box } x = t \text{ in } u : B}\,\Box E$$

Figure 2: Selected rules of $\lambda_e^{\to\Box}$

Davies and Pfenning show that $\lambda_e^{\to\Box}$ is also proof-theoretically well-behaved. Substitution of true and necessary facts works in the two sections of the context, respectively:

- if $\Gamma; \Delta \vdash a : A$ and $\Gamma; \Delta, x : A \vdash b : B$ then $\Gamma; \Delta \vdash b[a/x] : B$
- if $\Gamma; \cdot \vdash a : A$ and $\Gamma, u : A; \Delta \vdash b : B$ then $\Gamma; \Delta \vdash b[a/u] : B$

Terms are furthermore subject to an equational theory of β-reductions and η-expansions for which subject reduction and expansion hold. Davies and Pfenning productively use their system to analyze and model staged computation, with $\Box A$ denoting the type of source trees of type A. One subtler point is that quotation is an intensional modality, so one might want to deny the congruence that would conclude $\Gamma \vdash \text{box } t \equiv \text{box } u : \Box A$ from $\Gamma \vdash t \equiv u : A$.

1.2.5. Pattern-matching and invertibility. Unfortunately, there are two issues with this presentation. First, the authors note that the context can be separated into two zones because the order of antecedents in a sequent is irrelevant. This is not so in dependent type theory. Dependently-typed systems with split context presentations, such as the linear/non-linear DLNL of Krishnaswami, Pradic and Benton [42], only allow dependencies pointing one way between the two zones (in their case, linear assumptions can depend on structural ones). While quite appropriate in certain settings, this is a serious limitation in cases where it would be natural to interleave modal and non-modal assumptions.

Second, the pattern-matching/delayed substitution style of destructuring $\Box$ is an obstruction to goal-driven backwards proof refinement. Suppose we are given two derivations, $t : \Box A$ and $u : \Box B$, and wish to show that if A, B are necessary then so is C.

In a simultaneous substitution system, we can do this as

$$\frac{\Gamma \vdash t : \Box A \qquad \Gamma \vdash u : \Box B \qquad x : \Box A, y : \Box B \vdash v : C}{\Gamma \vdash \text{let } x, y = t, u \text{ in box } v : \Box C}$$

With split contexts, we can instead write

$$\dfrac{\Gamma;\Delta \vdash t : \Box A \qquad \dfrac{\dfrac{\Gamma;\Delta \vdash u : \Box B}{\Gamma, x : A;\Delta \vdash u : \Box B} \qquad \dfrac{\Gamma, x : A, y : B;\cdot \vdash v : C}{\Gamma, x : A, y : B;\Delta \vdash \text{box } v : \Box C}}{\Gamma, x : A;\Delta \vdash \text{let box } y = u \text{ in box } v : C}}{\Gamma;\Delta \vdash \text{let box } x = t \text{ in let box } y = u \text{ in box } v : \Box C}$$

While both derivations are correct, observe that a kind of forward deduction is the only way to proceed in either: the proofs begin by destructuring t, u before continuing with the construction of v under box v. In fact, it *has* to be this way. Let ∇ be a logical connective/type former with an introduction form $\vdash \text{nab } t : \nabla A$. Call nab *invertible* if any derivation of ∇A can be written as nab t for some t, or in other words can start with nab. Introduction of $\Box$ with split contexts is clearly not invertible because it erases the truth context: if there is anything useful there, we must first move it into the validity context using let box $x = t$ in u. Introduction in the simultaneous substitution style *is* invertible, but forces us to invent all boxed terms that we might later need a priori, before proceeding with let $x_1, ..., x_n = t_1, ..., t_n$ in box u (after that, true assumptions are similarly erased).

Programming in such languages is reminiscent of programming in A-normal form [25], an intermediate representation for functional compilers in which the arguments to a procedure call have to be values, as opposed to general compound terms that might be subject to evaluation. In the example, t and u must be turned into variables before they can be used. From a theorem proving perspective, we cannot delay boxed constructions until they are really needed—as is standard practice in goal-driven proof refinement—having instead to provide them earlier on. Other complications include the fact that delayed substitutions may have to be subject to *commuting conversions* which commute term formers past each other, for instance

$$\text{let box } y = (\text{let box } x = t \text{ in } u) \text{ in } v \longrightarrow \text{let box } x = t \text{ in let box } y = u \text{ in } v$$

in order to recover a good equational theory.

1.2.6. Fitch-style systems. These awkward facts led the research community to search for different presentations of modalities in type theory. Eventually, Borghuis [14], Martini and Mansini [49], and Pfenning and Wong [58] all offered broadly similar solutions with the common characteristic of relying on modal introduction and elimination rules that modify the context in "symmetric" ways, as we shall see below. Such systems are now known as *Fitch-style* owing to their similarity to Fitch's diagrammatic depictions of derivations [24].

In this line of work, Davies and Pfenning define $\lambda^{\to\Box}$, an "implicit" formulation of $\lambda_e^{\to\Box}$. In $\lambda^{\to\Box}$, the principal hypothetical judgment is of the form $\boxed{\Gamma_1;\Gamma_2;...;\Gamma_n \vdash M : A}$ where each of the Γ_i is an intuitionistic typing context. The sequence $\Gamma_1;\Gamma_2;...;\Gamma_n$ is called a *context stack* or *world stack* and is abbreviated by Ψ.

Although frame semantics (Section 1.2.2) are not explicitly developed for the logic, it is strongly influenced by them. We read $\Gamma_1;\Gamma_2;...;\Gamma_n \vdash M : A$ as "if assumptions Γ_1 hold in world w_1, and assumptions Γ_2 hold in world w_2 accessible from w_1, ..., and assumptions Γ_n

hold in world w_n accessible from w_{n-1}, then M has type A in w_n". Note that assumptions within a context are unordered as usual, but context stacks are ordered

The following rules of inference are postulated:

$$\frac{}{\Psi; \Gamma, x : A \vdash x : A}\,\mathrm{Var}$$

$$\frac{\Psi; \Gamma, x : A \vdash M : B}{\Psi; \Gamma \vdash \lambda(x : A).\ m : A \to B} \to I \qquad \frac{\Psi; \Gamma \vdash M : A \to B \qquad \Psi; \Gamma \vdash N : A}{\Psi; \Gamma \vdash M\ N : B} \to E$$

$$\frac{\Psi; \Gamma; \cdot \vdash M : A}{\Psi; \Gamma \vdash \mathrm{box}\ M : \Box A} \Box I \qquad \frac{\Psi; \Gamma \vdash M : \Box A}{\Psi; \Gamma; \Gamma_1; ...; \Gamma_n \vdash \mathrm{unbox}_n M : A} \Box E$$

Figure 3: Inference rules of $\lambda^{\to\Box}$

The intuition behind $\Box I$ is that to prove that A is necessary in the current world w, we ought to conclude A from no assumptions in an arbitrary world w' accessible from w. Note that (reading upwards) the context stack is extended and no information is lost. Unlike in $\lambda_e^{\to\Box}$, this rule is invertible, meaning that we can always proceed to write box without worry that this might render the goal unprovable.

Symmetrically, facts that are seen to be necessary in a prior world may also be true in the current one, though not always: this depends on whether the accessibility relation is transitive or reflexive. This is codified in the rule $\Box E$ which allows us to access necessity assumptions from earlier worlds as well as the current one (using unbox_0 as below).

The departure from $\lambda_e^{\to\Box}$ causes no harm as the proof theory is still nice: substitution of assumptions in any world is admissible, term reduction and expansion preserves types, etc. In metaprogramming applications, worlds correspond to stages of computation.

A common feature of Fitch-style systems is that the choice of modal logic can be controlled by varying modal elimination. In the case of $\lambda^{\to\Box}$, a variant containing only unbox_1 proves axiom $K : \Box(A \to B) \to \Box A \to \Box B$, but not $T : \Box A \to A$ or $4 : \Box A \to \Box\Box A$. For these, one can check that unbox_0 and unbox_2 are needed, respectively.

1.2.7. Locked contexts. More recently, a Fitch-style λ-calculus was given by Clouston [18]. Its main advantage is a general and clean category-theoretic semantics. In this calculus, hypothetical judgments are the standard ones $\boxed{\Gamma \vdash t : A}$. However, the grammar of contexts is extended to

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

where $\Gamma, 🔒$ is a *locked* context. The rules for a modal λ-calculus K are then as follows

$$\frac{🔒 \notin \Gamma'}{\Gamma, x : A, \Gamma' \vdash x : A}\,\text{Var}$$

$$\frac{\Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda(x : A).\ t : A \to B} \to I \qquad \frac{\Gamma \vdash t : A \to B \qquad \Gamma \vdash u : A}{\Gamma \vdash t\ u : B} \to E$$

$$\frac{\Gamma, 🔒 \vdash t : A}{\Gamma \vdash \text{box}\ t : \Box A} \Box I \qquad \frac{\Gamma \vdash t : \Box A \qquad 🔒 \notin \Gamma'}{\Gamma, 🔒, \Gamma' \vdash \text{unbox}\ t : A} \Box E$$

Figure 4: Clouston's presentation of a Fitch-style λ-calculus for intuitionistic modal logic K.

This system is almost identical to $\lambda^{\to\Box}$. Comparing Figure 3 and Figure 4, by identifying the context stack $\Gamma_1; \Gamma_2; ...; \Gamma_n$ with $\vdash \Gamma_1, 🔒, \Gamma_2, ..., 🔒, \Gamma_n$ ctx, we can see that the two introduction rules are identical, and Clouston's unbox is unbox_1 in $\lambda^{\to\Box}$ (thus the logic is K rather than S4). Locking the context with 🔒 is intuitively analogous to pushing a new world onto the stack, or opening a new "proof box" whose conclusion is a necessity.

The system lends itself to a natural semantic interpretation. Let $\mathcal{C}$ be a given cartesian closed category, and $\Diamond \dashv \Box$ an adjunction of endofunctors on $\mathcal{C}$. Then one can interpret the system in $\mathcal{C}$ by extending the standard interpretation of simple type theory in CCCs in which contexts and types are interpreted as objects, and terms as morphisms. In particular, 🔒 corresponds to the action of $\Diamond$, whereas the type former $\Box$ to the action of the functor $\Box$:

$$\begin{aligned}
&[\![\vdash \Gamma\ \text{ctx}]\!] \in \mathcal{C} && [\![\vdash A\ \text{type}]\!] \in \mathcal{C} \\
&[\![\cdot]\!] = 1 && [\![A_0]\!] = A_0\ \text{(base type)} \\
&[\![\Gamma, x : A]\!] = [\![\Gamma]\!] \times [\![A]\!] && [\![1]\!] = 1 \\
&[\![\Gamma, 🔒]\!] = \Diamond [\![\Gamma]\!] && [\![A \times B]\!] = [\![A]\!] \times [\![B]\!] \\
& && [\![A \to B]\!] = [\![B]\!]^{[\![A]\!]} \\
& && [\![\Box A]\!] = \Box [\![A]\!]
\end{aligned}$$

A term $\Gamma \vdash t : A$ is interpreted as a morphism $[\![\Gamma]\!] \xrightarrow{[\![t]\!]} [\![A]\!]$. One then observes that box is precisely one direction of the hom-equivalence $\mathcal{C}(\Diamond A, B) \simeq \mathcal{C}(A, \Box B)$, whereas unbox is the other one preceded by a weakening. Do note that $[\![\Gamma, \Gamma']\!] = [\![\Gamma]\!] \times [\![\Gamma']\!]$ when $🔒 \notin \Gamma'$.

$$\frac{\Diamond [\![\Gamma]\!] \xrightarrow{[\![t]\!]} [\![A]\!]}{[\![\Gamma]\!] \xrightarrow{[\![\text{box}\ t]\!]} \Box [\![A]\!]} \qquad \frac{[\![\Gamma]\!] \xrightarrow{[\![t]\!]} \Box [\![A]\!]}{\Diamond [\![\Gamma]\!] \times [\![\Gamma']\!] \xrightarrow{\pi_1} \Diamond [\![\Gamma]\!] \xrightarrow{\widehat{[\![t]\!]}} [\![A]\!]}$$

This interpretation is sound with respect to the equational theory of proof terms—that is, if $\Gamma \vdash t \equiv_{\beta\eta} u : A$ then $[\![t]\!] = [\![u]\!]$—and can be made complete for the class of models "cartesian closed categories with adjunctions of endofunctors" via the addition of a type former $\Diamond$ to the syntax.

As in $\lambda^{\to\Box}$, one can control which modal logic is expressed by changing the elimination rule. While above we had intuitionistic K, Clouston shows that the following alternative rule yields intuitionistic S4. Semantically, this requires $\Box$ to be an idempotent comonad, i.e., a comonad whose comultiplication $\mu : \Box \Rightarrow \Box\Box$ is a natural isomorphism.

$$\frac{\Gamma \vdash t : \Box A}{\Gamma, \Gamma' \vdash \text{unbox } t : A}\,\Box E$$

Fitch-style systems are an ergonomic solution that extends well to dependent types. Indeed, Clouston's presentation has been adapted to Martin-Löf type theory (Section 1.2.11). Nevertheless, the frameworks presented thus far remain inadequate for the goal of embedding internal reasoning by virtue of being restricted to an *endomodality*, that is a modality which takes $\vdash A$ type in the language to another $\vdash \Box A$ type in the *same* language. There is no reason to expect that an internal language of a category $\mathcal{E}$ will be the same one as the ambient type theory which we use for "external" mathematics and in which $\mathcal{E}$ is defined. For the appropriate generalization, we must look to *multimode* or *adjoint* logics which arose out of investigations into *linear* logic.

1.2.8. Linear logic. Much progress on modalities in computer science has been driven by investigations into *linear logic* and its computational interpretations.[2]

Linear logic was invented in the 1980s by Girard [28,27], guided by a semantic analysis of intuitionistic logic in terms of *coherence spaces*: a version of Scott domains in which finite approximants must exist. From a proof-theoretic point of view, linear logic can be seen as discarding two out of the three *structural rules*. Presented as part of an intuitionistic sequent calculus, the structural rules are weakening, contraction, and exchange:

$$\frac{\Delta \longrightarrow C}{\Delta, A \longrightarrow C}\,\text{Weaken} \quad \frac{\Delta, A, A \longrightarrow C}{\Delta, A \longrightarrow C}\,\text{Contract} \quad \frac{\Delta, B, A, \Delta' \longrightarrow C}{\Delta, A, B, \Delta' \longrightarrow C}\,\text{Exchange}$$

In a *structural logic*, that is a logic which admits all three rules, hypothesis contexts can be identified with *sets* of assumptions in which the ordering and multiplicity of assumptions is irrelevant. *Substructural logics* drop some of the rules. Linear logic drops weakening and contraction, but keeps exchange. As a result, contexts become *multisets* in which assumptions appear with specific multiplicities. The context can be viewed as tracking the exact multiset of assumptions (or resources) required to prove a statement (or make a construction).

Linear logic is interesting for many reasons. One reason is that it allows us to view the structural logical connectives as composed of finer, linear connectives that collapse in the presence of structural rules. Conjunction, for instance, splits into a *multiplicative* tensor product $\otimes$ and an *additive* direct product $\&$:

$$\frac{\Delta \longrightarrow A \qquad \Delta' \longrightarrow B}{\Delta, \Delta' \longrightarrow A \otimes B}\otimes R \quad \frac{\Delta, A, B \longrightarrow C}{\Delta, A \otimes B \longrightarrow C}\otimes L$$

$$\frac{\Delta \longrightarrow A \qquad \Delta \longrightarrow B}{\Delta \longrightarrow A \& B}\,\&R \quad \frac{\Delta, A \longrightarrow C}{\Delta, A \& B \longrightarrow C}\,\&L_1 \quad \frac{\Delta, B \longrightarrow C}{\Delta, A \& B \longrightarrow C}\,\&L_2$$

[2] An understanding of linear logic is not essential to the project proposed here, but it provides a historical motivation for the development of adjoint logic. Parts of this section are copied from a miniproject report available at https://ice1000.org/lnl-modal/report.html°.

It is not difficult to check that with structural rules readded, $A \otimes B \longrightarrow A \& B$ and $A \& B \longrightarrow A \otimes B$ become derivable. On the other hand, an analysis of sequent derivations in cut-free normal form shows that without those rules, the connectives are not interderivable.

Linear logic can be extended by an operator !, with the intended meaning of $!A$ being "A is true structurally". The operator is also referred to as 'of course' or just 'bang'. Though one of Girard's original proposals viewed 'of course' as a derived operator defined in terms of linear connectives, it can be taken as a primitive unary connective. Here, $!\Delta$ stands for a context where all hypotheses are of the form $!A$ for some A.

$$\frac{!\Delta \longrightarrow A}{!\Delta \longrightarrow !A}\,!R \qquad \frac{\Delta, A \longrightarrow C}{\Delta, !A \longrightarrow C}\,!L$$

$$\frac{\Delta \longrightarrow C}{\Delta, !A \longrightarrow C}\,!\text{Weaken} \qquad \frac{\Delta, !A, !A \longrightarrow C}{\Delta, !A \longrightarrow C}\,!\text{Contract}$$

The purpose of ! is to restore a way of reasoning structurally. Indeed, structural intuitionistic logic can be soundly embedded into intuitionistic linear logic (ILL) with 'of course': a formula is provable in structural logic iff its translation in terms of ! is provable linearly [26]. Linear logics are thus strictly more expressive than their structural cousins.

Given just the rules above, it is easy to see that 'of course' obeys the laws of an S4 modality: in judgmental form, $!A \longrightarrow A$ and $!A \longrightarrow !!A$. We can consequently recognize linear logic as a kind of substructural S4 modal logic.

1.2.9. Mixed linear/non-linear logic. Shortly after Girard's invention, Seely [65,10] and de Paiva [57] independently outlined category-theoretic interpretations of intuitionistic and classical linear logic. Their approaches were somewhat complex, and Benton [7] eventually observed that the semantics can be simplified to a fully symmetric picture.

A *mixed linear/non-linear* (LNL) model is a triple of
- a cartesian closed category $\mathcal{C}$; and
- a symmetric monoidal closed category $\mathcal{L}$; and
- a symmetric monoidal adjunction $F \dashv G$ between them.

The new viewpoint leads to a proof-theoretic innovation. Benton's LNL, the mixed linear/non-linear logic, is given a two-layer term assignment. The grammar of types is

$$A, B ::= A_0 \mid I \mid A \otimes B \mid A \multimap B \mid FX$$
$$X, Y ::= X_0 \mid 1 \mid X \times Y \mid X \to Y \mid GA$$

where the *linear types* A, B are to be interpreted in $\mathcal{L}$, the *structural types* X, Y in $\mathcal{C}$, and the two adjoint functors F, G are viewed as *shifts* mediating between the layers.

The split is completed by introducing two forms of judgment:

- $\boxed{X_1, \ldots, X_m; A_1, \ldots, A_n \vdash_{\mathcal{L}} a : B}$ for linear constructions; and
- $\boxed{X_1, \ldots, X_n \vdash_{\mathcal{C}} t : Y}$ for structural ones.

Note here that structural terms can only depend on structural assumptions, whereas linear terms use a split structural/linear context. Writing Γ for structural contexts and Δ for linear ones, the rules for layer-shifting are then

$$\frac{\Gamma \vdash_{\mathcal{C}} t : X}{\Gamma; \cdot \vdash_{\mathcal{L}} F(t) : FX}\, FI \qquad \frac{\Gamma; \Delta \vdash_{\mathcal{L}} b : FX \qquad \Gamma, x : X; \Delta' \vdash_{\mathcal{L}} a : A}{\Gamma; \Delta, \Delta' \vdash_{\mathcal{L}} \text{let } F(x) = b \text{ in } a : A}\, FE$$

$$\frac{\Gamma; \cdot \vdash_{\mathcal{L}} a : A}{\Gamma \vdash_{\mathcal{C}} G(a) : GA}\, GI \qquad \frac{\Gamma \vdash_{\mathcal{C}} t : GA}{\Gamma; \cdot \vdash_{\mathcal{L}} \text{derelict } t : A}\, GE$$

Figure 5: LNL λ-calculus rules for $F \dashv G$

The calculus is interpreted in LNL models. Terms are subject to an equational theory of β-reductions and, and as the term let $F(x) = b$ in a is a kind of delayed substitution, also commuting conversions (Section 1.2.5). These are all sound w.r.t. interpretation in LNL models.

The two layers can be put back together: there is a translation $(-)^\circ$ such that one has $\Gamma \vdash e : A$ in intuitionistic linear logic iff $(\Gamma)^\circ \vdash_{\mathcal{L}} e^\circ : A^\circ$. Importantly, $(!A)^\circ = FG(A^\circ)$. This is a reflection of the fact that semantically, the comonad FG of the adjunction $F \dashv G$ is symmetric monoidal; this is precisely what's needed to model 'of course'. For instance, the following is a derivation of $!A \multimap A$.

$$\frac{\dfrac{}{\Gamma; a : FGA \vdash_{\mathcal{L}} a : FGA} \qquad \dfrac{\dfrac{}{\Gamma, x : GA \vdash_{\mathcal{C}} x : GA}}{\Gamma, x : GA; \cdot \vdash_{\mathcal{L}} \text{derelict } x : A}}{\Gamma; a : FGA \vdash_{\mathcal{L}} \text{let } F(x) = a \text{ in derelict } x : A}$$

Developing this idea, one can establish that LNL models provide semantics for the multiplicative fragment (without & or $\oplus$) of intuitionistic linear logic.

1.2.10. Adjoint and multimode logic. It was understood for a long time that the categorical semantics of different formal systems with a modal component are closely related, all of them being variations of monads, comonads, and adjunctions that interact well with type structure (note that adjunctions appear in two distinct ways: in LNL, ! is the comonad of an adjunction, whereas in Clouston's calculus, $\Box$ is the right adjoint). Early on, Moggi [50] observed the connection between monads and modalities. Benton [7] pointed out that any LNL model is also a model of Moggi's computational metalanguage, though one of a special nature: in such a model, the monad is necessarily commutative. Benton and Wadler [8] further developed the connection, relating translations from pure to monadic λ-calculi with translations from structural to linear intuitionistic logic.

Proof-theoretically, the situation is described in generality by Reed's *adjoint logic* [62], a sequent calculus combining arbitrarily many layers, possibly substructural and each using different connectives, with one layer for each element of a preorder of *modes*, and with operations moving between layers p, q when $p \leq q$. Reed shows that systems such as $\lambda^{\to\Box}$ (Section 1.2.4), or Girard's linear logic with ! (Section 1.2.8), can all be formulated in this framework. This was generalized further by Licata and Shulman [46] from a preorder to a 2-category of modes, with applications in combined synthetic topology and homotopy theory.

1.2.11. Modal dependent type theory.[3] Having considered propositional logic and simple type theory, let us now see how modalities carry over to the dependently-typed setting.

The first dependently-typed modal calculus may have been the contextual modal S4 type theory of Nanevski, Pfenning and Pientka [54]. This system has been successfully used in the logical analysis of metavariables, and serves as foundation for the mechanized logical framework Beluga [59]. However, it is a system in the LF family, and is tailored for that specific use quite distinct from type theories aimed at directly encoding mathematics, with a stratification into types and kinds that excludes type-indexed type families (i.e., quantification over a type universe). It is consequently not applicable here.

I aim to develop Kripke-Joyal modalities as an application of the Fitch-style calculus of dependent right adjoints λ_{DRA} due to Birkedal and coauthors [11]. λ_{DRA} is syntactically and semantically similar to Clouston's simply typed modal λ-calculus (Section 1.2.7) for the logic K.

Some of the context, type, term formation, and term equality rules for λ_{DRA} with dependent products are as follows:

$\boxed{\vdash \Gamma \text{ ctx}}$

$$\frac{}{\vdash \cdot \text{ ctx}} \qquad \frac{\vdash \Gamma \text{ ctx} \qquad \Gamma \vdash A \text{ type}}{\vdash \Gamma, x : A \text{ ctx}} \, (x \notin \Gamma)$$

$$\frac{\vdash \Gamma \text{ ctx}}{\vdash \Gamma, \text{🔒} \text{ ctx}} \qquad \frac{\vdash \Gamma, x : A, y : B, \Gamma' \text{ ctx}}{\vdash \Gamma, y : B, x : A, \Gamma' \text{ ctx}} \, (x \text{ not free in } B)$$

$\boxed{\Gamma \vdash A \text{ type}}$

$$\frac{\Gamma \vdash A \text{ type} \qquad \Gamma, x : A \vdash B \text{ type}}{\Gamma \vdash \Pi(x : A).\ B \text{ type}} \qquad \frac{\Gamma, \text{🔒} \vdash A \text{ type}}{\Gamma \vdash \Box A \text{ type}}$$

$\boxed{\Gamma \vdash t : A}$

$$\frac{\Gamma \vdash t : A \qquad \Gamma \vdash A \equiv B \text{ type}}{\Gamma \vdash t : B} \qquad \frac{\vdash \Gamma, x : A, \Gamma' \text{ ctx}}{\Gamma, x : A, \Gamma' \vdash x : A} \, (\text{🔒} \notin \Gamma')$$

$$\frac{\Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda(x : A).\ t : \Pi(x : A).\ B} \qquad \frac{\Gamma \vdash t : \Pi(x : A).\ B \qquad \Gamma \vdash u : A}{\Gamma \vdash t\ u : B[u/x]}$$

$$\frac{\Gamma, \text{🔒} \vdash t : A}{\Gamma \vdash \text{box } t : \Box A} \qquad \frac{\Gamma \vdash t : \Box A \qquad \vdash \Gamma, \text{🔒}, \Gamma' \text{ ctx}}{\Gamma, \text{🔒}, \Gamma' \vdash \text{unbox } t : \Box A} \, (\text{🔒} \notin \Gamma')$$

$\boxed{\Gamma \vdash t \equiv u : A}$

$$\frac{\Gamma \vdash (\lambda(x : A).\ t)\ u : A}{\Gamma \vdash (\lambda(x : A).\ t)\ u \equiv t[u/x] : A} \qquad \frac{\Gamma \vdash t : \Pi(x : A).\ B}{\Gamma \vdash t \equiv (\lambda(x : A).\ t\ x) : A} \, (x \notin \Gamma)$$

$$\frac{\Gamma \vdash \text{unbox box } t : A}{\Gamma \vdash \text{unbox box } t \equiv t : A} \qquad \frac{\Gamma \vdash t : \Box A}{\Gamma \vdash t \equiv \text{box unbox } t : \Box A}$$

This system is then given a sound interpretation in any category with families (CwF) together with a *dependent right adjoint* (DRA), a generalization of a right adjoint to CwF semantics.

[3] This section may be difficult to read without consulting the citations.

λ_{DRA} does not support a multimode modality (Section 1.2.10), or multiple modalities. In a system they call FitchTT, Gratzer and collaborators [30] introduce such a generalization. The basic semantic object behind FitchTT is a *parametric right adjoint* (PRA): a functor $G : \mathcal{C} \to \mathcal{D}$ from a category with a terminal object 1 such that $G/1 : \mathcal{C}/1 \cong \mathcal{C} \to \mathcal{D}/G(1)$ is an (ordinary) right adjoint. The authors demonstrate that this structure is exactly what underlies admissibility of substitution in modal dependent type theories such as λ_{DRA}. The existence of a PRA guarantees admissibility of substitution even in the presence of "non-definable substitutions", that is morphisms between contexts present in a CwF or natural model [5] that are not lists of terms $\Gamma \vdash t_1 : A_1, \Gamma \vdash t_2 : A_2[t_1], ..., \Gamma \vdash t_n : A_n[t_1, ..., t_{n-1}]$ (i.e., morphisms in the image of the term model). In contrast, these are not admissible in λ_{DRA}. However, this property is only relevant in languages that allow reference to non-definable substitutions; since I do not intend to provide an internal language in the sense of Gratzer et al., I expect this not to be an issue and prefer to stick with the simpler syntax of λ_{DRA}.

Ultimately, I hope that some combination of the two approaches of λ_{DRA} and FitchTT will work as a starting point for the present project.

1.3. Toposes

In this section we recall a minimal set of fundamentals of topos theory and categorical logic, as seen in Mac Lane and Moerdijk [44], needed to specify the proposed project.

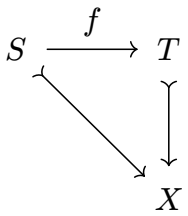
1.3.1. Subobjects.

Elementary toposes are categories with a well-behaved notion of *subobject*.

Def. 1.3.1.1: In a category $\mathcal{C}$, a *subobject* of $X \in \text{Ob}(\mathcal{C})$ is an equivalence class of monomorphisms over X (that is, monomorphisms $S \rightarrowtail X$) up to isomorphism. If $\mathcal{C}$ is locally small, we can speak of the set $\text{Sub}(X)$ of subobjects of X.

Example. In **Set**, we can take the "canonical" injection representing a subobject (a class of injections $i : S \hookrightarrow X$) to be the image $i[S] \subseteq X$.

Prop. 1.3.1.1: Each $\text{Sub}(X)$ is furthermore a poset with $S \leq T$ iff there is a map of subobjects $S \to T$, i.e., a map f that commutes as below. Note that any such map has to be unique by definition of monomorphism.



Prop. 1.3.1.2: In a locally small category $\mathcal{E}$ with pullbacks, the mapping $X \mapsto \text{Sub}(X)$ extends to a functor $\text{Sub} : \mathcal{E}^{\text{op}} \to \mathbf{Poset}$ where **Poset** is the category of partially ordered sets.

Def. 1.3.1.2: In a category $\mathcal{E}$ with finite limits, a *subobject classifier* is an object $\Omega \in \text{Ob}(\mathcal{E})$ and a monomorphism $1 \xrightarrowtail{\text{true}} \Omega$ such that:

- any mono $S \rightarrowtail X$ is the pullback of some map $X \xrightarrow{\varphi} \Omega$ as below; and

- isomorphic monos over X correspond to the same φ.

$$\begin{array}{ccc} S & \longrightarrow & 1 \\ \downarrow & & \downarrow{\scriptstyle \text{true}} \\ X & \xrightarrow[\varphi]{} & \Omega \end{array}$$

Example. In **Set**, the booleans $\mathbb{B} = \{0, 1\}$, with $\{\star\} \xrightarrow{\text{true}} \mathbb{B}$ given by $\text{true}(\star) = 1$, form a subobject classifier. The subobject/map correspondence is between subsets and their characteristic maps.

Prop. 1.3.1.3: In a locally small category with finite limits, a subobject classifier Ω is equivalently a representing object for Sub, i.e., an object Ω such that $\text{Sub} \cong よ\Omega$ where よ denotes the Yoneda embedding.

Notation. For an arrow $X \xrightarrow{\varphi} \Omega$, write $\{\varphi\}$ for the corresponding subobject. We say $\{\varphi\}$ is *represented* or *classified* by φ.

Note. The existence of a subobject classifier can be viewed as a simple case of a strictness result (in the sense of *strictification* [47]) because it provides an arrow-based representation of subobjects that is strictly unique (up to equality). However, it is not as helpful as in proof-relevant models since any two equivalent monos over an object are connected by a unique isomorphism.

Def. 1.3.1.3: An *elementary topos* is a cartesian closed category with finite limits and a subobject classifier.

Examples.
- The category **Set** of sets.
- For any $\mathcal{C}$, the presheaf category $\widehat{\mathcal{C}} = [\mathcal{C}^{\text{op}}, \mathbf{Set}]$.

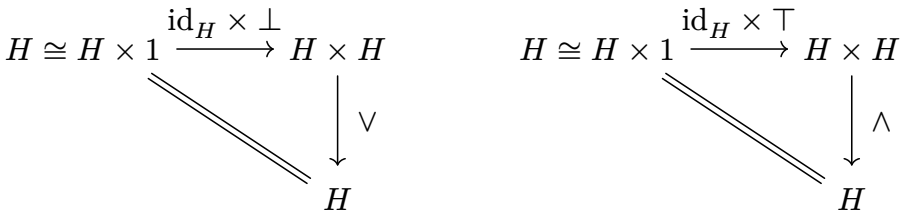
1.3.2. Internal representation of logical connectives.

Characteristic maps in a topos can be combined via propositional connectives, and quantification makes sense there.

Def. 1.3.2.1: An *internal Heyting algebra* (or a *Heyting algebra object*) in a category $\mathcal{C}$ is an object H together with maps

- $H \times H \xrightarrow{\wedge} H$
- $H \times H \xrightarrow{\vee} H$
- $H \times H \xrightarrow{\Rightarrow} H$
- $1 \xrightarrow{\top} H$
- $1 \xrightarrow{\bot} H$

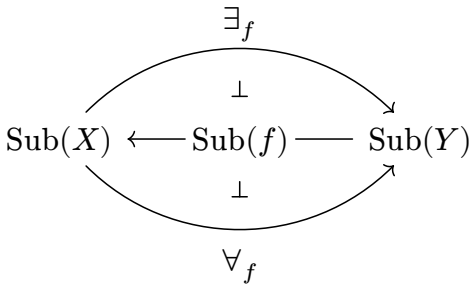
for which diagrams expressing the Heyting algebra laws commute. For example, $x \vee \bot = x$ and $x \wedge \top = x$ are expressed as



Example. In **Set**, an internal Heyting algebra is an ordinary/"external" Heyting algebra. In particular, the subobject classifier $\Omega = \mathbb{B}$ is a Boolean algebra.

Prop. 1.3.2.1: In an elementary topos, the subobject classifier is an internal Heyting algebra with $\top = \mathrm{true}$.

Prop. 1.3.2.2: In an elementary topos, for any $f : X \to Y$ there is an adjoint triple $\exists_f \dashv \mathrm{Sub}(f) \dashv \forall_f$



Prop. 1.3.2.3: In an elementary topos $\mathcal{E}$, for any $X \in \mathrm{Ob}(\mathcal{E})$ there are *internal quantification maps* $\Omega^X \xrightarrow{\exists_X, \forall_X} \Omega$ such that for any $S \in \mathrm{Sub}(\Gamma \times X)$ represented by $\Gamma \times X \xrightarrow{\chi_S} \Omega$,
$\exists_{\pi_1}(S)$ is represented by $\Gamma \xrightarrow{\widetilde{\chi_S}} \Omega^X \xrightarrow{\exists_X} \Omega$,
and $\forall_{\pi_1}(S)$ by $\Gamma \xrightarrow{\widetilde{\chi_S}} \Omega^X \xrightarrow{\forall_X} \Omega$,
where $\widetilde{(-)} : \mathcal{E}(A \times B, C) \simeq \mathcal{E}(A, C^B)$.

1.3.3. Simple type theory and its interpretation.

Since elementary toposes are cartesian closed, they can soundly interpret simple type theory. To be semi-precise,[4] fix a grammar of types, contexts, and terms, and postulate the standard typing rules. Contexts are ordered. For now, the type and term constants A_0 and $(t_0 : A)$, respectively, are assumed to be elements of fixed sets of symbols; these will be disposed of later.

$$\text{types } A, B ::= A_0 \mid 1 \mid A \times B \mid A \to B$$
$$\text{contexts } \Gamma, \Delta ::= \cdot \mid \Gamma, x : A \quad (x \notin \Gamma)$$
$$\text{terms } t, u ::= (t_0 : A) \mid x \mid () \mid (t, u) \mid \pi_1\ t \mid \pi_2\ t \mid \lambda(x : A).\ t(x) \mid t\ u$$

[4] I am not aiming for precision w.r.t. issues of α-equivalence and variable-freshness, or presentations of λ-theories with equational axioms.

$$\frac{}{\Gamma \vdash (t_0 : A) : A} \qquad \frac{}{\Gamma, x : A, \Gamma' \vdash x : A} \qquad \frac{}{\Gamma \vdash () : 1}$$

$$\frac{\Gamma \vdash t : A \qquad \Gamma \vdash u : B}{\Gamma \vdash (t, u) : A \times B} \qquad \frac{\Gamma \vdash t : A \times B}{\Gamma \vdash \pi_1\ t : A} \qquad \frac{\Gamma \vdash t : A \times B}{\Gamma \vdash \pi_2\ t : B}$$

$$\frac{\Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda(x : A).\ t(x) : A \to B} \qquad \frac{\Gamma \vdash t : A \to B \qquad \Gamma \vdash u : A}{\Gamma \vdash t\ u : B}$$

Now fix an elementary topos $\mathcal{E}$, and an object A_0 of $\mathcal{E}$ for every type constant A_0. This induces interpretations of types as objects $[\![A]\!] \in \mathrm{Ob}(\mathcal{E})$, and of contexts as finite products $[\![\Gamma]\!] \in \mathrm{Ob}(\mathcal{E})$:

$$[\![A_0]\!] = A_0 \qquad [\![\cdot]\!] = 1$$
$$[\![1]\!] = 1 \qquad [\![\Gamma, x : A]\!] = [\![\Gamma]\!] \times [\![A]\!]$$
$$[\![A \times B]\!] = [\![A]\!] \times [\![B]\!]$$
$$[\![A \to B]\!] = [\![B]\!]^{[\![A]\!]}$$

Furthermore, given an interpretation $1 \xrightarrow{t_0} [\![A]\!]$ of every term constant $(t_0 : A)$, a term $\Gamma \vdash t : A$ can be interpreted as an arrow $[\![\Gamma]\!] \xrightarrow{[\![\Gamma \vdash t : A]\!]} [\![A]\!]$. Formally this proceeds by recursion on typing derivations; but well-typed pairs (Γ, t) are in bijective correspondence with their typing derivations. For brevity, one may write $[\![t]\!]$ for $[\![\Gamma \vdash t : A]\!]$.

$$[\![\Gamma \vdash (t_0 : A) : A]\!] = [\![\Gamma]\!] \xrightarrow{!_{[\![\Gamma]\!]}} 1 \xrightarrow{t_0} [\![A]\!]$$

$$[\![\Gamma, x : A, \Gamma' \vdash x : A]\!] = [\![\Gamma]\!] \times [\![A]\!] \times [\![\Gamma']\!] \xrightarrow{\pi_i} [\![A]\!]$$

$$[\![\Gamma \vdash () : 1]\!] = [\![\Gamma]\!] \xrightarrow{!_{[\![\Gamma]\!]}} 1$$

$$[\![\Gamma \vdash (t, u) : A \times B]\!] = [\![\Gamma]\!] \xrightarrow{\langle [\![t]\!], [\![u]\!] \rangle} [\![A]\!] \times [\![B]\!]$$

$$[\![\Gamma \vdash \pi_1\ t : A]\!] = [\![\Gamma]\!] \xrightarrow{[\![t]\!]} [\![A]\!] \times [\![B]\!] \xrightarrow{\pi_1} [\![A]\!]$$

$$[\![\Gamma \vdash \pi_2\ t : B]\!] = [\![\Gamma]\!] \xrightarrow{[\![t]\!]} [\![A]\!] \times [\![B]\!] \xrightarrow{\pi_2} [\![B]\!]$$

$$[\![\Gamma \vdash \lambda(x : A).\ t(x) : A \to B]\!] = [\![\Gamma]\!] \xrightarrow{[\![\Gamma, x : \widetilde{A \vdash t} : B]\!]} [\![B]\!]^{[\![A]\!]}$$

$$[\![\Gamma \vdash t\ u : B]\!] = [\![\Gamma]\!] \xrightarrow{\langle [\![t]\!], [\![u]\!] \rangle} [\![B]\!]^{[\![A]\!]} \times [\![A]\!] \xrightarrow{\mathrm{eval}} [\![B]\!]$$

No reasonable variation of a semantic interpreter should be able to distinguish equivalent terms. That is, we should have soundness: if $\Gamma \vdash t \equiv u : A$ then $[\![t]\!] = [\![u]\!]$ where $(\equiv)$ is the congruence closure of the generating computational/extensional (or β-reduction/η-expansion; or the two directions of a universal property) equations below.

$$\frac{\Gamma \vdash t : 1}{\Gamma \vdash t \equiv () : 1}$$

$$\frac{\Gamma \vdash t : A \qquad \Gamma \vdash u : B}{\Gamma \vdash \pi_1\ (t, u) \equiv t : A} \qquad \frac{\Gamma \vdash t : A \qquad \Gamma \vdash u : B}{\Gamma \vdash \pi_2\ (t, u) \equiv u : B} \qquad \frac{\Gamma \vdash t : A \times B}{\Gamma \vdash t \equiv (\pi_1\ t, \pi_2\ t) : A \times B}$$

$$\frac{\Gamma, x : A \vdash t : B \qquad \Gamma \vdash u : A}{\Gamma \vdash (\lambda(x : A).\ t)\ u \equiv t[u/x] : B} \qquad \frac{\Gamma \vdash t : A \to B}{\Gamma \vdash t \equiv (\lambda(x : A).\ t\ x) : A \to B}$$

1.3.4. Interpretation of propositions.

To serve as a term assignment for multisorted higher-order logic, simple type theory can be extended by a universe of propositions. We adjust the grammar and extend typing as follows.

types $A, B ::= \ldots \mid \mathrm{Prop}$

terms $t, u ::= \ldots \mid \varphi$

propositions $\varphi, \psi ::= t =_A u \mid \top \mid \bot \mid \varphi \wedge \psi \mid \varphi \vee \psi \mid \varphi \Rightarrow \psi \mid \forall(x : A).\ \varphi \mid \exists(x : A).\ \varphi$

$$\frac{\Gamma \vdash t : A \qquad \Gamma \vdash u : A}{\Gamma \vdash t =_A u : \mathrm{Prop}}$$

$$\frac{}{\Gamma \vdash \top : \mathrm{Prop}} \qquad \frac{}{\Gamma \vdash \bot : \mathrm{Prop}} \qquad \frac{\Gamma \vdash \varphi : \mathrm{Prop} \qquad \Gamma \vdash \psi : \mathrm{Prop}}{\Gamma \vdash \varphi \wedge \psi : \mathrm{Prop}}$$

$$\frac{\Gamma \vdash \varphi : \mathrm{Prop} \qquad \Gamma \vdash \psi : \mathrm{Prop}}{\Gamma \vdash \varphi \vee \psi : \mathrm{Prop}} \qquad \frac{\Gamma \vdash \varphi : \mathrm{Prop} \qquad \Gamma \vdash \psi : \mathrm{Prop}}{\Gamma \vdash \varphi \Rightarrow \psi : \mathrm{Prop}}$$

$$\frac{\Gamma, x : A \vdash \varphi : \mathrm{Prop}}{\Gamma \vdash \forall(x : A).\ \varphi : \mathrm{Prop}} \qquad \frac{\Gamma, x : A \vdash \varphi : \mathrm{Prop}}{\Gamma \vdash \exists(x : A).\ \varphi : \mathrm{Prop}}$$

Note that the propositional universe is impredicative, i.e., $\forall(x : A).\ \varphi$ is a proposition when φ is even though A is a type. Semantically, the additions make use of elementary topos structure: the subobject classifier, the internal Heyting algebra on Ω, and internal quantification operators. Type interpretation is extended by $[\![\mathrm{Prop}]\!] = \Omega$, whereas term semantics by

$$[\![\Gamma \vdash t =_A u : \mathrm{Prop}]\!] = [\![\Gamma]\!] \xrightarrow{\langle [\![t]\!], [\![u]\!] \rangle} [\![A]\!] \times [\![A]\!] \xrightarrow{\delta_{[\![A]\!]}} \Omega$$

$$[\![\Gamma \vdash \top : \mathrm{Prop}]\!] = [\![\Gamma]\!] \xrightarrow{!_{[\![\Gamma]\!]}} 1 \xrightarrow{\top} \Omega$$

$$[\![\Gamma \vdash \bot : \mathrm{Prop}]\!] = [\![\Gamma]\!] \xrightarrow{!_{[\![\Gamma]\!]}} 1 \xrightarrow{\bot} \Omega$$

$$[\![\Gamma \vdash \varphi \wedge \psi : \mathrm{Prop}]\!] = [\![\Gamma]\!] \xrightarrow{\langle [\![\varphi]\!], [\![\psi]\!] \rangle} \Omega \times \Omega \xrightarrow{\wedge} \Omega$$

$$[\![\Gamma \vdash \varphi \vee \psi : \mathrm{Prop}]\!] = [\![\Gamma]\!] \xrightarrow{\langle [\![\varphi]\!], [\![\psi]\!] \rangle} \Omega \times \Omega \xrightarrow{\vee} \Omega$$

$$[\![\Gamma \vdash \varphi \Rightarrow \psi : \mathrm{Prop}]\!] = [\![\Gamma]\!] \xrightarrow{\langle [\![\varphi]\!], [\![\psi]\!] \rangle} \Omega \times \Omega \xrightarrow{\Rightarrow} \Omega$$

$$[\![\Gamma \vdash \forall(x : A).\ \varphi : \mathrm{Prop}]\!] = [\![\Gamma]\!] \xrightarrow{[\![\Gamma, x : \widetilde{A \vdash \varphi} : \mathrm{Prop}]\!]} \Omega^{[\![A]\!]} \xrightarrow{\forall_{[\![A]\!]}} \Omega$$

$$[\![\Gamma \vdash \exists(x : A).\ \varphi : \mathrm{Prop}]\!] = [\![\Gamma]\!] \xrightarrow{[\![\Gamma, x : \widetilde{A \vdash \varphi} : \mathrm{Prop}]\!]} \Omega^{[\![A]\!]} \xrightarrow{\exists_{[\![A]\!]}} \Omega$$

where $\delta_X : X \times X \to \Omega$ is the map classifying $X \xrightarrow{\langle \mathrm{id}_X, \mathrm{id}_X \rangle} X \times X$.

1.3.5. A notion of proof.

Finally, a layer of proof terms to inhabit the propositions is convenient. As is usual in presentations of constructive type theory in the Brouwer-Heyting-Kolmogorov tradition, propositions can serve double duty as the types of their proofs, and proof terms can be ordinary terms.

We extend the syntax again:

$$\begin{aligned} &\text{types } A, B ::= \ldots \mid \varphi \\ &\text{terms } t, u ::= \ldots \mid \mathrm{refl} \mid \mathrm{propext}\ t \end{aligned}$$

The constructive logic internal to an elementary topos is proof-irrelevant, propositionally extensional, and features extensional identity types. Other kinds of proof terms are omitted to simplify the presentation.

$$\frac{\Gamma \vdash \varphi : \mathrm{Prop}}{\Gamma \vdash \varphi\ \mathrm{type}} \qquad \frac{\Gamma \vdash \varphi : \mathrm{Prop} \qquad \Gamma \vdash t : \varphi \qquad \Gamma \vdash u : \varphi}{\Gamma \vdash t \equiv u : \varphi}$$

$$\frac{\Gamma \vdash t : A}{\Gamma \vdash \mathrm{refl}\ t : t =_A t} \qquad \frac{\Gamma \vdash \_ : t =_A u}{\Gamma \vdash t \equiv u : A} \qquad \frac{\Gamma \vdash t : \varphi \Rightarrow \psi \qquad \Gamma \vdash u : \psi \Rightarrow \varphi}{\Gamma \vdash \mathrm{propext}\ t\ u : \varphi =_{\mathrm{Prop}} \psi}$$

The semantics is extended to propositions-as-types

$$[\![\Gamma \vdash \varphi\ \mathrm{type}]\!] = \{\varphi\}$$

where $\{\varphi\}$ is the subobject classified by $[\![\varphi]\!]$.

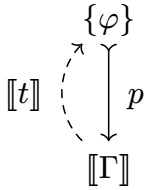
Accounting for proof terms needs a slight adjustment. The issue is that if we continued to interpret terms as morphisms, we'd have $[\![\Gamma \vdash t : \varphi]\!] \in \mathcal{E}([\![\Gamma]\!], \{\varphi\})$. But this is unsound for proof irrelevance because such hom-sets aren't necessarily singletons. (More precisely, one could postulate two inequal proof term constants that violate proof irrelevance; proof terms

that do not contain such constants *already are* sections and will not give rise to a contradiction.) To fix this (without going to full dependent types), we can change the type of the interpretation function to require that terms $\Gamma \vdash t : \varphi$ of a proposition-type be interpreted as morphisms that are furthermore sections of $\{\varphi\} \rightarrowtail [\![\Gamma]\!]$.

$$[\![\Gamma \vdash t : \varphi]\!] \in \left\{ s : \mathcal{E}([\![\Gamma]\!], \{\varphi\}) \mid s; p = \mathrm{id}_{[\![\Gamma]\!]} \right\}$$

$$[\![\Gamma \vdash \mathrm{refl}\ t : t =_A t]\!] = [\![\Gamma]\!] \xrightarrow{\mathrm{id}_{[\![\Gamma]\!]}} [\![\Gamma]\!] \cong \{t =_A t\}$$

$$\left[\!\left[\Gamma \vdash \mathrm{propext}\ t\ u : \varphi =_{\mathrm{Prop}} \psi\right]\!\right] = [\![\Gamma]\!] \xrightarrow{\mathrm{id}_{[\![\Gamma]\!]}} [\![\Gamma]\!] \cong \left\{\varphi =_{\mathrm{Prop}} \psi\right\}$$

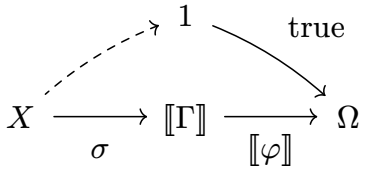


where we have used twice the fact that if $t, u : [\![\Gamma]\!] \to [\![A]\!]$ are equal arrows, then $\{t =_A u\} \cong [\![\Gamma]\!]$.

1.4. Kripke-Joyal semantics

Kripke-Joyal semantics provides a forcing-like notation $\Vdash$ for proving statements about an elementary topos by reasoning in the language outlined just above.

Def. 1.4.1:[5] Given an object $X \in \mathrm{Ob}(\mathcal{E})$, a proposition $\Gamma \vdash \varphi : \mathrm{Prop}$, and an arrow $X \xrightarrow{\sigma} [\![\Gamma]\!]$, we write $X \Vdash \varphi[\sigma]$ ("$\varphi[\sigma]$ is forced at stage X") to mean that $\sigma; [\![\varphi]\!]$ factors through true, as below.



Note. 'σ' is so named, and square-bracketed in $\varphi[\sigma]$, because it behaves like a top-level delayed substitution for $[\![\Gamma]\!]$. This is discussed again in Section 2.2.3.

Notation. The proposition $\mathcal{E} \Vdash \varphi[\sigma]$ stands for $1 \Vdash \varphi[\sigma]$.

Prop. 1.4.1 *(Soundness)*: If $\Gamma \vdash h : \varphi \Rightarrow \psi$ and $X \Vdash \varphi[\sigma]$, then $X \Vdash \psi[\sigma]$.

Prop. 1.4.2 *(Monotonicity)*: If $X \Vdash \varphi[\sigma]$, then $Y \Vdash \varphi[f; \sigma]$ for any $f : Y \to X$.

Prop. 1.4.3 *(Gluing)*: For any epimorphism $p : Y \twoheadrightarrow X$, if $Y \Vdash \varphi[p; \sigma]$ then $X \Vdash \varphi[\sigma]$.

Prop. 1.4.4: The following Tarski-style truth conditions characterize forcing:

[5] This presentation departs from the usual definition in terms of subobjects, but is equivalent to it. Thanks to Fernando Larrain for discussions about this.

$$
\begin{aligned}
X \Vdash t =_A u &\text{ iff } [\![t]\!] = [\![u]\!] \\
X \Vdash \top &\text{ always} \\
X \Vdash \bot &\text{ iff } X \cong 0 \\
X \Vdash \varphi[\sigma] \wedge \psi[\sigma] &\text{ iff } X \Vdash \varphi[\sigma] \text{ and } X \Vdash \psi[\sigma] \\
X \Vdash \varphi[\sigma] \vee \psi[\sigma] &\text{ iff there exists an epimorphism } [f, g] : Y + Y' \twoheadrightarrow X \\
&\quad \text{such that } Y \Vdash \varphi[f; \sigma] \text{ and } Y' \Vdash \psi[g; \sigma] \\
X \Vdash \varphi[\sigma] \Rightarrow \psi[\sigma] &\text{ iff for any } f : Y \to X, \\
&\quad \text{if } Y \Vdash \varphi[f; \sigma] \text{ then } Y \Vdash \psi[f; \sigma] \\
X \Vdash \forall(x : A).\ \varphi[\sigma, x] &\text{ iff for any } f : Y \to X \text{ and any } s : Y \to [\![A]\!], \\
&\quad Y \Vdash \varphi[(f; \sigma), s] \\
X \Vdash \exists(x : A).\ \varphi[\sigma, x] &\text{ iff there exists an epimorphism } p : Y \twoheadrightarrow X \text{ and } s : Y \to [\![A]\!] \\
&\quad \text{such that } Y \Vdash \varphi[(p; \sigma), s]
\end{aligned}
$$

1.4.1. Locality of forcing.

The following example illustrates one sense in which statements in the internal logic are only true "locally". We will see that "exists" and "or" statements (joins) only assert the existence of local witnesses on a cover. Consequently they may be true even when a corresponding global witness does not exist.

Def. 1.4.1.1: An epimorphism $f : A \to B$ is said to *split* when it has a *section* $s : B \to A$, i.e., an arrow s such that $s; f = \mathrm{id}_B$ exists.

Def. 1.4.1.2: For a group G, the category $\boldsymbol{G}$-**Set** has

- objects $(X, \cdot_X : G \times X \to X)$ where X is a set and $\cdot$ a left G-action on X; and
- morphisms $(X, \cdot_X) \xrightarrow{\text{f}} (Y, \cdot_Y)$ functions $f : X \to Y$ that are equivariant: $f(g \cdot_X x) = g \cdot_Y f(x)$ for every $g \in G$ and $x \in X$.

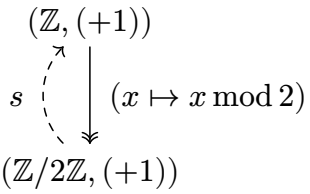
Prop. 1.4.1.1: Given any G, $\boldsymbol{G}$-**Set** is an elementary topos.

Def. 1.4.1.3: A category $\mathcal{C}$ is said to satisfy the *external axiom of choice* (external AC) when every epic arrow in $\mathcal{C}$ splits.

Def. 1.4.1.4: An elementary topos $\mathcal{E}$ is said to satisfy the *internal axiom of choice* (internal AC), when for all $A, B \in \mathcal{E}$,
$\mathcal{E} \Vdash \forall f : A \to B,\ (\forall b : B,\ \exists a : A,\ f\ a = b) \to \exists s : B \to A,\ \forall b : B,\ f\ (s\ b) = b.$

Prop. 1.4.1.2:[6] Let $\mathbb{Z}^+$ be the additive group $(\mathbb{Z}, +, 0)$. Then $\mathbb{Z}^+$-**Set** does not satisfy external AC.

[6] Thanks to Owen Milner for suggesting this example.



Proof. A $\mathbb{Z}^+$-set is equivalently a set equipped with a permutation. A map of $\mathbb{Z}^+$-sets is a permutation-preserving function $f : (X, \sigma_X \in S_X) \to (Y, \sigma_Y \in S_Y)$, in the sense that $f(\sigma_X(x)) = \sigma_Y(f(x))$ for every $x \in X$. Consider the $\mathbb{Z}^+$-sets $(\mathbb{Z}, x \mapsto x + 1)$ and $(\mathbb{Z}/2\mathbb{Z}, x \mapsto x + 1 \pmod 2)$. There is a permutation-preserving projection $x \mapsto x \bmod 2$ from $\mathbb{Z}$ onto $\mathbb{Z}/2\mathbb{Z}$, but any section s would have $s(0) = s(1) + 1$ and $s(1) = s(0) + 1$. Thus no section can exist. ■

Despite this, the internal axiom of choice does hold, being inherited from the ambient metatheory! This is actually easier to prove using the fact that internal AC holds iff every object is internally projective (every $(-)^P$ preserves epimorphisms) [44:p. 312], but we can use it as an example of Kripke-Joyal forcing in practice. We should first characterise exponential objects and G as a G-set.

Prop. 1.4.1.3: $(G, g \cdot_G h = gh)$ acting on itself by left-multiplication has $\boldsymbol{G}\textbf{-Set}(G, A) \simeq |A|$ where $|A|$ is the underlying set of A.

Proof. $\boldsymbol{G}\textbf{-Set}$ is equivalently the presheaf category $[G^{\mathrm{op}}, \mathbf{Set}]$ over G viewed as a groupoid with one object $\star$, and $G \cong \text{卐} \star$. The bijection follows by the Yoneda lemma. ■

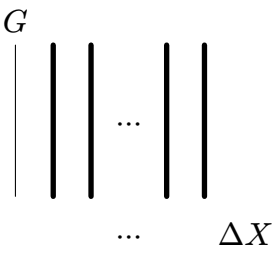
Prop. 1.4.1.4: For $A, B \in \mathrm{Ob}(\boldsymbol{G}\textbf{-Set})$, the exponential B^A is $\left(|B|^{|A|}, g \cdot_{B^A} f = (x \mapsto g \cdot_B f(g^{-1} \cdot_A x))\right)$.

Proof. By unfolding the characterisation of exponentials in a presheaf category. ■

Prop. 1.4.1.5:[7] (In a classical metatheory) $\boldsymbol{G}\textbf{-Set}$ satisfies internal AC.

Proof. By the truth conditions, we are given $f : X \to B^A$ such that $X \Vdash \forall b : B, \exists a : A, f\ a = b$ (i.e., f is internally surjective) and need to show that $X \Vdash \exists s : B \to A, \forall b : B, f\ (s\ b) = b$.

Given a G-set X, write ΔX for a discretized X, namely $(|X|, g \cdot_{\Delta X} x = x)$, the underlying set of X with a trivial action. Observe that X is covered by $G \times \Delta X$ via the epic map $(g, x) \overset{p_X}{\mapsto} g \cdot_X x$. This is equivariant because $h \cdot_{G \times \Delta X} (g, x) = (hg, x) \mapsto (hg) \cdot_X x = h \cdot_X (g \cdot_X x)$.



Orbits in $G \times \Delta X$

Now by monotonicity we may assume that $G \times \Delta X \Vdash \forall b : B, \exists a : A, (p_X; f)\ a = b$, and by gluing it suffices to show that $G \times \Delta X \Vdash \exists s : B \to A, \forall b : B, (p_X; f)\ (s\ b) = b$. It does not help to know that our local element factors through p_X, so generalize $p_X; f$ to suppose we are given some arbitrary $k : G \times \Delta X \to B^A$. By the characterization of exponentials and G, k is actually an $|X|$-indexed family K of ordinary (possibly non-equivariant) functions $K_i : |A| \to |B|$.

[7] Thanks to Chris Grossack for an exposition of similar results: https://grossack.site/2022/12/13/internal-logic-examples°.

Now, the assumption of internal surjectivity is equivalent [44:p. 310] to saying that $(G \times \Delta X) \times A \xrightarrow{\langle \pi_1, \hat{k} \rangle} (G \times \Delta X) \times B$ is epic, where $\hat{k}$ is the exponential transpose of k. In $\boldsymbol{G}$-$\mathbf{Set}$ this means the underlying function $|\langle \pi_1, \hat{k} \rangle|$ is surjective, and this in turn simply means that every K_i is surjective.

By the truth condition for existence and using the identity cover, it now suffices to prove that there exists $s : G \times \Delta X \to A^B$ such that $G \times \Delta X \Vdash \forall b : B, k\ (s\ b) = b$. This final forcing condition says that each element S_i of the $|X|$-indexed family S corresponding to s is right-inverse to K_i. So we can use the ambient axiom of choice to obtain all the S_i. ■

In summary, though equivariant maps may not have global sections, it suffices to find local ones one a discrete cover.

Note. Eagerly unfolding forcing statements according to the truth conditions quickly leads to a proliferation of quantifiers. We can collapse iterated forall/exists, but only if they don't alternate. As usual in mathematics, it doesn't pay to always unfold; it is better to prove lemmas such as the one relating $X \Vdash \forall b : B, \exists a : A, f\ a = b$ to some morphism being epic.

2. Kripke-Joyal notation as a modality

In this chapter, I describe the proposed thesis project.

2.1. Motivation

Proof assistants for a multitude of type theories exist. Many of these theories have known or conjectured classes of models for which they are sound: Lean with its set-valued models in ZFC with a large cardinals assumption [16,17], homotopy type theory with models in model categories [37,5], Cubical Agda [69] with cubical set models [19] and so on. Correspondingly, whenever a deduction or construction is stamped correct by the typechecker, we can be sure of the existence of a corresponding semantic entity. In this sense, computer-assisted verification of internal constructions has seen plenty of implementations. However, the connection between a theory and its models can be complex, and our understanding tenuous, making it difficult to see exactly what that semantic entity is. Software cannot assist us in this process if the model only exists on paper.

In this project, I seek to develop computer-aided reasoning "one level up": working within a proof assistant that implements some choice of a foundation for general mathematics, instantiate *both* the model, the theory, and the relation between them.

Previous projects had related goals involving the formalization of deductive systems as objects of study. These commonly write down a "deep embedding", meaning a sequence of datatypes presenting the syntax and derivation trees of the object logic, followed by results about theories such as soundness and completeness w.r.t. a class of models. For instance, the Flypitch project of Han and van Doorn [32,33] defines first-order logic and theories, the theory of ZFC, and proves that the continuum hypothesis is independent of ZFC using forcing.

In this case, the forced models exist primarily to tell us something about the theory rather than the other way around. Since the object logic is not a vehicle for substantial developments (e.g., there is no need to formalise real analysis in the just-defined ZFC), it is not of importance to provide a great user experience—not to mention anything remotely like a proof assistant—for building derivations in it. Nevertheless, the authors do find a need for sufficiently many internal constructions to warrant developing some custom automation. Their tactics automate tautologies such as $(a \Rightarrow b) \sqcap (b \Rightarrow c) \leq (a \Rightarrow c)$ in a complete Boolean algebra $\mathbb{B}$, and enable reasoning equationally in $\mathbb{B}$-valued structures. They note that an approach using *reflection*—that is one capable of relating deeply embedded first-order proof trees to proof terms of the metalogic (in this case Lean 3)—could be more helpful by enabling the reuse of preexisting meta-level automation.

The more object-level derivations we build, the more important such tooling becomes. In the limit, we desire a proof assistant that seamlessly mixes internal and external modes of reasoning. Developing such an assistant for the internal language of (1-)toposes expressed via Kripke-Joyal semantics is the goal of the proposed project, with the purpose of providing a convenient setting for internally proving statements about models. In this project I view models as primary, and internal language machinery as a convenient syntax for them. Besides contributing Kripke-Joyal semantics specifically, I aim to investigate how such connections should be built in general.

2.2. Design considerations

Any mechanization project begins with a choice of proof assistant. I choose to work in Lean 4 [52] for the following three reasons:

- At over 1.5MLOC as of early 2024,[8] the `mathlib` project [20] is a substantial library of classical mathematics on which one can rely for foundational results. In particular, the library contains definitions of Grothendieck topologies and sheaf toposes[9] which can be used as a target model.
- Lean 4 provides state-of-the-art facilities for syntax extensions and metaprogramming [67]. This technology makes it practical to embed expressive domain specific languages,[10] granting the ability to implement any type-theoretic extension whatsoever. For Kripke-Joyal semantics, we can express forcing `X ⊩ (-)` as a macro with custom elaboration (typechecking) behaviour.
- The type theory of Lean is reasonably close to the higher-order logic internal to (1-)toposes. See Section 2.3.2.

2.2.1. Prototype workflow. Around the summer of 2023, jointly with Gabriel Ebner I procured a prototype implementation of a macro for Kripke-Joyal forcing. The prototype is illustrated in the following sequence of refinements of a simple proof.

```
example {C : Type} [Category C] {J : GrothendieckTopology C}
    (U : C) (F : Sheaf J Type) : I(J | U ⊩ ∀ (x y : F), x = x) :=
  sorry
```

We begin by postulating a category `C` together with a Grothedieck topology `J` (i.e., a site), an object `U` of `C` to serve as the stage of forcing, and an object `F` in the topos of sheaves of types on `J`. Since `F` is an object of the topos, in the internal logic it is a type. The macro `I(J | U ⊩ -)` allows us to express that a given forcing statement is true. In this case, we claim that `U` forces the tautological statement `∀ (x y : F), x = x`. We begin the proof by admitting it with `sorry`. At this point, the (abbreviated) goal state at the `sorry` appears as

```
C : Type
inst✝¹ : Category.{0, 0} C
J : GrothendieckTopology C
U : C
F : Sheaf J Type
⊢ I(J | U ⊩ ∀ (x y : F), x = x)
```

The shown expected type is syntax sugar: the *actual* expected type is a statement of pure Lean (i.e., Lean without an `I(...)` macro) obtained by eagerly expanding the truth conditions:

```
⊢ ∀ (V : C), (V ⟶ U) →
    ∀ (s : F.obj (Opposite.op V)) (V_1 : C) (f : V_1 ⟶ V),
      F.obj (Opposite.op V_1) →
        (F.map f.op (s)) = (F.map f.op (s))
```

corresponding to the statement

[8] https://leanprover-community.github.io/mathlib_stats.html°

[9] https://leanprover-community.github.io/mathlib4_docs/Mathlib/CategoryTheory/Sites/Sheaf.html°

[10] For example a language of arithmetic expressions: https://lean-lang.org/lean4/doc/metaprogramming-arith.html°

$$\forall V : \mathcal{C}, \forall \_ : \mathcal{C}(V, U), \forall s : F(V), \forall V_1 : \mathcal{C}, \forall f : \mathcal{C}(V_1, V), \forall \_ : F(V_1), F|_f(s) = F|_f(s)$$

Now, we wish to prove this in the internal logic. To do so, we apply a modal introduction rule (Section 1.2.3) which says that to show `I(J | U ⊩ φ)`, it suffices to show `φ` internally. This appears as a term former `i(J | U ⊩ sorry)`

```
example {C : Type} [Category C] {J : GrothendieckTopology C}
    (F : Sheaf J Type) : I(J | U ⊩ ∀ (x y : F), x = x) :=
  i(J | U ⊩ sorry)
```

Upon applying `i`, the goal state at the new `sorry` becomes an internal one:

```
F : Type
⊢ ∀ (x y : F), x = x
```

And we can finish the proof using standard tactics by introducing binders and applying reflexivity.

```
example {C : Type} [Category C] {J : GrothendieckTopology C}
    (F : Sheaf gt Type) : I(J | U ⊩ ∀ (x y : F), x = x) :=
  i(J | U ⊩ by
    intros x y
    rfl)
```

This example, though simplistic, illustrates the workflow that I am aiming for: forcing statements mean something external, but can be proven internally.

Unfortunately, the macro turned out to be suffering from two deficiencies:

- A failure of substitution due to the use of an incorrect modal introduction rule, as explained in Section 1.2.3.
- An awkward interpretation procedure. See Section 2.2.3.

Nevertheless, the prototype also has good properties which I hope to preserve in future versions. Notably, proof terms in the internal logic—appearing inside `i(J | U ⊩ -)`—are standard Lean terms. As native elaboration facilities are used, one may even execute ordinary Lean tactics when reasoning internally. Such an alignment, when possible (requiring perhaps that the meta-logic and object-logic are sufficiently similar), enables the reuse of existing elaborators and tactics rather than reinventing them all. Not all metaprograms can be reused, as in Lean many of them are tailored for classical logic, but the ones that aren't are sufficiently useful. Furthermore, syntactic alignment makes the macro feel natural, and it should not take much for an existing user of the language to learn to use it.

2.2.2. Embedding variables. There are projects which embed an object logic in a proof assistant while also providing a usable, interactive proof environment for it. One widely used example is Iris [36], a program logic aimed at verifying programs in a choice of programming language against an operational semantics. Iris has been embedded in Coq. Though the object logic (Iris) is substructural whereas the metalogic (Coq) structural, the Iris Proof Mode (IPM) [39,40] provides a suite of tactics, notations, and other machinery that makes object-level proving as convenient as meta-level proving.

The authors of IPM point out that context-based management of variables as seen in every proof assistant—that is, the fact that one is given a bag of named hypotheses, each of which can be referred to at any point—is essential to supporting natural reasoning.

In Flypitch, Han and van Doorn observe the same issue: a manual proof of the tautology $(a \Rightarrow b) \sqcap (b \Rightarrow c) \leq (a \Rightarrow c)$ in a complete Boolean algebra $\mathbb{B}$ entails suffering in part because of the need to manually apply lemmas about commutativity of meets $\sqcap$ or joins $\sqcup$. In their case, object-level variables of the propositional logic implicit in $\mathbb{B}$ can be reified as meta-level variables via the Yoneda embedding for posets: to show $\varphi \leq \psi$, it suffices to show for any $\Gamma : \mathbb{B}$ that if $\Gamma \leq \varphi$ then $\Gamma \leq \psi$. The assumption $h : \Gamma \leq (a \Rightarrow b) \sqcap (b \Rightarrow c)$ can then be mechanically split into two assumptions $h_1 : \Gamma \leq a \Rightarrow b$ and $h_2 : \Gamma \leq b \Rightarrow c$ by the universal property of $\sqcap$. In this way one leverages the meta-level context to manage object-level assumptions.

Having learned that providing a notion of named variable is fundamental to working in an embedded logic, how do Flypitch, IPM, and this proposal compare in handling variables?

- In Flypitch, one proves statements about certain models—complete Boolean algebras and Boolean-valued structures—by "inventing" variables using the Yoneda trick above. Statements of the form $\Gamma \vdash_{\square\exists\forall\mathbb{N}} (T \vdash_{\mathrm{FOL}} \varphi)$ are also shown, where $\vdash_{\square\exists\forall\mathbb{N}}$ is standard Lean entailment, $\vdash_{\mathrm{FOL}}$ deeply embedded first-order entailment, T a first-order theory, and φ a first-order sentence. There appears to be no special support for these.
- In IPM, one constructs proofs of $\Gamma \vdash_{\text{🐓}} (\Delta \vdash_{\mathrm{Iris}} \varphi)$ where $\vdash_{\text{🐓}}$ is standard Coq entailment, $\vdash_{\mathrm{Iris}}$ shallowly embedded Iris entailment, Δ a *reified context*, and φ an Iris proposition. Reified contexts are maps from hypothesis names to their types. Keeping track of this data is exactly what allows IPM to make it seem as-if one is reasoning *in* Iris, even though one is really constructing a Coq representation of an Iris entailment.[11] For example, the goal

  $$x : A \vdash_{\text{🐓}} (\mathrm{HP} : P, \mathrm{H}\Psi : \Psi\ x, \mathrm{HR} : R \vdash_{\mathrm{Iris}} \Psi\ x * P)$$

  which is equivalent via a coercion to the goal

  $$x : A \vdash_{\text{🐓}} (P * \Psi\ x * R \vdash_{\mathrm{Iris}} \Psi\ x * P)$$

  will be displayed as

  ```
  x : A
  ──────────────────────────────────(1/1)
  "HP" : P
  "HΨ" : Ψ x
  "HR" : R
  ──────────────────────────────────*
  Ψ x * P
  ```

- Relying on the fact that the internal logic of a topos is structural, in this project I aim to represent object-level variables as meta-level variables. As outlined in Section 2.3.1, using the modal point of view we seek to construct derivations such as $\Gamma, (X \Vdash \varphi) \vdash_{\square\exists\forall\mathbb{N}} (X \Vdash \psi)$ where $X \Vdash (-)$ is the modal forcing operator. Note that internal-logic contexts never appear explicitly because they are embedded in meta-level contexts; an object-level assumption is a meta-level assumption in the image of $X \Vdash (-)$.

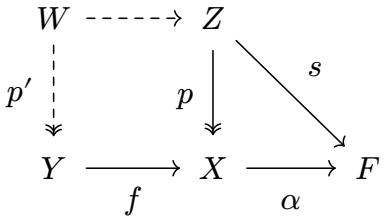
2.2.3. Embedding substitutions. The companion notion to "variable" is a notion of "substitution". I have already stressed the importance of the choice of modal inference rules to retain admissible substitutions. But there is another question specific to Kripke-Joyal semantics.

Recall that in the definition of forcing (Def. 1.4.1), the substitution σ is kept 'in front' of $X \Vdash \varphi[\sigma]$. In procedural terms, the meaning of this statement is obtained by first interpreting φ as an arrow $[\![\varphi]\!] : [\![\Gamma]\!] \to \Omega$ according to the internal logic of the topos. At this point φ is viewed as an *open formula* with variables in Γ. Only then do we precompose by σ to obtain the meaning of $X \Vdash \varphi[\sigma]$ as "σ; $[\![\varphi]\!]$ factors through true".

This framing makes it trivial to show monotonicity (see below) and not too difficult to show gluing [44:p. 304]. On the other hand, in real usage we want to push σ inside φ and directly reference local elements within our formulas. For instance, in the truth condition for equality, I wrote $X \Vdash t =_A u$ rather than $X \Vdash (x =_A y)[t, u]$. Similarly in the example proof of Prop. 1.4.1.5, we freely reference local elements from internal formulas, as does every use of Kripke-Joyal in the literature. But this leaves open the question of interpretation. Is a statement such as $X \Vdash t =_A u$ to be interpreted by "packing up" all the occurrences of local terms (here t, u) into σ and applying the standard definition of forcing, or is it to be interpreted by a direct recursion on the formula?

Disappointingly, this kind of pedantic distinction seems to matter when constructing proof automation software. In the prototype of Section 2.2.1, a direct recursion was used. Specifically, `I(J | U ⊩ φ)` is a macro that stands for (is interpreted as) the complete expansion of `φ` per its truth conditions.

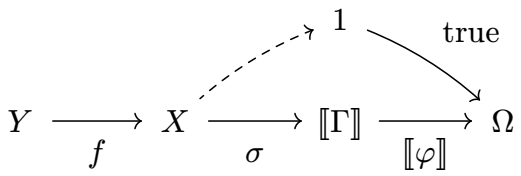
Now, given `f : V → U` and `h : I(J | U ⊩ φ)`, we should have by monotonicity `I(J | V ⊩ φ')`, where `φ'` is `φ` with all occurrences of local elements precomposed by `f`. But an interpretation based on truth conditions renders it algorithmically suboptimal to construct a proof of `I(J | V ⊩ φ')`.



Suppose for concreteness that we are given $X, F \in \mathrm{Ob}(\mathcal{E})$ and $\alpha : X \longrightarrow_{\mathcal{E}} F$ such that $X \Vdash \exists(y : F).\ y = \alpha$. Suppose we also have $f : Y \longrightarrow_{\mathcal{E}} X$ and want to show $Y \Vdash \exists(y : F).\ y = (f; \alpha)$. If $X \Vdash \varphi$ *means* whatever the truth conditions say, then equivalently we know there exists an epic map $p : Z \twoheadrightarrow_{\mathcal{E}} X$ and a local element $s : Z \longrightarrow_{\mathcal{E}} F$ such that $s = p; \alpha$. We need to show that there exists an object W, an epic map $p' : W \twoheadrightarrow_{\mathcal{E}} Y$, and a local element $s' : W \longrightarrow_{\mathcal{E}} F$ such that $s' = p'; f; \alpha$. One can take the pullback of p along f as pullback preserves epimorphisms in a topos. ■

While that was not difficult on its own, it seems that something has gone wrong: we are inspecting the structure of φ to prove something that can be proven parametrically over all φ: per Def. 1.4.1, one just observes that a factorization of σ; $[\![\varphi]\!]$ is still a factorization of $f; \sigma$; $[\![\varphi]\!]$.

[11] The shallow embedding of Iris means that one is proving validities in a model rather than constructing reified derivation trees; but that is beside the point, as either way some entailment-like relation of the object logic, or a semantics for it, is the subject of meta-level reasoning.



On top of the above, successive applications of monotonicity must be coherent (i.e., substitutions must compose)! For instance, $X \Vdash \varphi[(g; f); \sigma]$ iff $W \Vdash \varphi[g; (f; \sigma)]$. With delayed substitutions this is a single rewrite along a propositional equality $(g; f); \sigma = g; (f; \sigma)$, whereas in the expanded picture it entails a complex propagation of the rewrite to all term occurrences.

In summary, the delayed substitution formulation serves well as a vehicle for proving statements about forcing, but is not usable; whereas direct references are usable, but may be complicated to handle. One way or another, an implementation of forcing needs to account for this tension. The complications make it seem strictly better to implement Def. 1.4.1 directly. That, however, leaves open the question of how to provide a nice user experience for it.

2.2.4. Experimental approach. I advocate for an experimental approach to the design of formal systems and verification software: rather than attempting to develop something in a vacuum based on abstract principles, it is essential to target concrete theorems or constructions that the language should render expressible and try the system out on those. Only with this experience is it appropriate to go back to the drawing board and reason abstractly about the situation at hand.

The prototype of Section 2.2.1 exposed tangible issues; as it does not scale beyond the simplest of statements, these must now be repaired. After doing so, I intend to test the system on proof such as that of Prop. 1.4.1.5. It is a simple formalization target, but enough to exercise the forcing machinery.

More ambitious, long-term goals could include formalizations of synthetic algebraic geometry following Blechschmidt [13], or Mulvey's proof of a Serre-Swan-type theorem [53].

2.3. Proposal for a forcing modality

Having surveyed the area, outlined initial experiments, and pointed out some issues, I ought now to present a solution. Unfortunately, I do not yet know what that should be. In the remainder of this chapter, I sketch a possible way of going about it. I propose to embed intuitionistic HOL in Lean via a Fitch-style modal operator in the style of λ_{DRA} (Section 1.2.11) in order to express Kripke-Joyal forcing.

Warning: text below this point contains conjectures, speculation, and half-baked ideas. No correctness results have been established for the proposed language.

2.3.1. $\square\exists\forall\mathbb{N}^{\Vdash}$**.**

Consider the type theory underlying Lean as described by Carneiro [16,17]. We will extend this to a system $\square\exists\forall\mathbb{N}^{\Vdash}$ containing a forcing modality.

The raw syntax of Lean does not distinguish types and terms. Extend it as follows (additions in maroon):

expressions $e, f, g ::= x \mid U_\ell \mid c_{\overline{\ell}} \mid e\ f \mid \lambda(x : e).\ f \mid \forall(x : e).\ f \mid t \mid A \mid i_e(t) \mid I_e(A) \mid e \Vdash \varphi$

contexts $\Gamma ::= \cdot \mid \Gamma, x : e$

where t, A, φ belong to the syntactic categories of HOL terms, types, and propositions, respectively (Section 1.3.4), amended as follows (additions in maroon):

types $A, B ::= \text{unbox } e \mid 1 \mid A \times B \mid A \to B \mid \text{Prop}$

contexts $\Gamma_i ::= \cdot \mid \Gamma_i, x : A \mid \Gamma, \text{🔒}_e$

terms $t, u ::= \text{unbox } e \mid x \mid () \mid (t, u) \mid \pi_1\ t \mid \pi_2\ t \mid \lambda(x : A).\ t(x) \mid t\ u \mid \varphi$

propositions $\varphi, \psi ::= t =_A u \mid \top \mid \bot \mid \varphi \wedge \psi \mid \varphi \vee \psi \mid \varphi \Rightarrow \psi \mid \forall(x : A).\ \varphi \mid \exists(x : A).\ \varphi$

The changes to the HOL grammar are that:

- base constants are gone from the internal language; instead, we will get them from the enclosing environment by unboxing (or, if you like, *antiquoting*); and
- internal contexts Γ_i can be formed by locking an external (Lean) context Γ.

The combined language LEMN$^{\Vdash}$ has two layers of judgments: $\vdash_i$ in the internal logic, and $\vdash_{\Vdash}$ in the extended external layer.

While in Section 1.3.4 every syntactically well-formed HOL context was well-formed, here a context well-formedness judgment in the internal layer becomes necessary in order to account for the locking rule.

$\boxed{\vdash_i \Gamma_i \text{ ctx}}$

$$\frac{\Gamma \vdash_{\Vdash} (\mathcal{C}, J) : \text{Site} \qquad \Gamma \vdash_{\Vdash} U : \text{Ob}(\mathcal{C})}{\vdash_i \Gamma, \text{🔒}_{(U,J)} \text{ ctx}}$$

$$\frac{}{\vdash_i \cdot \text{ ctx}} \qquad \frac{\vdash_i \Gamma_i \text{ ctx} \qquad \Gamma_i \vdash_i A \text{ type}}{\vdash_i \Gamma_i, x : A \text{ ctx}}\ (x \notin \Gamma_i)$$

where Site (the type of sites with a Grothendieck topology) and $\text{Ob}(\mathcal{C})$ are as defined in `mathlib`.

Inference rules for $\vdash_i$ and $\vdash_{\Vdash}$ are now those of HOL and Lean, respectively, plus the following:

$\boxed{\Gamma_i \vdash_i A \text{ type}}$

$$\frac{\Gamma \vdash_{\Vdash} e : \text{Sh}(\mathcal{C}, J)}{\Gamma, \text{🔒}_{(U,J)}, \Gamma_i \vdash_i \text{unbox } e \text{ type}}$$

$\boxed{\Gamma_i \vdash_i t : A}$

$$\frac{\Gamma \vdash_{\Vdash} e : U \longrightarrow_{\text{Sh}(\mathcal{C},J)} I_{(U,J)}(A)}{\Gamma, \text{🔒}_{(U,J)}, \Gamma_i \vdash_i \text{unbox } e : A} \qquad \frac{\Gamma \vdash_{\Vdash} e : (U, J) \Vdash \varphi}{\Gamma, \text{🔒}_{(U,J)}, \Gamma_i \vdash_i \text{unbox } e : \varphi}$$

$\boxed{\Gamma \vdash_{\Vdash} e : f}$

$$\frac{\Gamma, \text{🔒}_{(U,J)} \vdash_i A \text{ type}}{\Gamma \vdash_{\Vdash} I_{(U,J)}(A) : \text{Sh}(\mathcal{C}, J)} \qquad \frac{\vdash_i \Gamma, \text{🔒}_{(U,J)}, \Gamma_i \text{ ctx}}{\Gamma \vdash_{\Vdash} I_{(U,J)}(\Gamma_i) : \text{Sh}(\mathcal{C}, J)}$$

$$\frac{\Gamma, \text{🔒}_{(U,J)}, \Gamma_i \vdash_i t : A}{\Gamma \vdash_{\Vdash} i_{(U,J)}(t) : U \times I_{(U,J)}(\Gamma_i) \longrightarrow_{\text{Sh}(\mathcal{C},J)} I_{(U,J)}(A)}$$

$$\frac{\Gamma, \text{🔒}_{(U,J)} \vdash_i \varphi : \text{Prop}}{\Gamma \vdash_{\Vdash} (U, J) \Vdash \varphi : \text{Prop}} \qquad \frac{\Gamma, \text{🔒}_{(U,J)} \vdash_i t : \varphi}{\Gamma \vdash_{\Vdash} i_{(U,J)}(t) : (U, J) \Vdash \varphi}$$

where $\mathrm{Sh}(\mathcal{C}, J)$ is the topos of sheaves on $(\mathcal{C}, J)$ (again as defined in `mathlib`), and $A \longrightarrow_{\mathcal{D}} B \triangleq \mathcal{D}(A, B)$.

In addition, to account for locality, we may want to include monotonicity and gluing operators that interact well with the modal structure. For monotonicity,

$$\frac{\Gamma \vdash_{\Vdash} f : V \longrightarrow_{\mathcal{C}} U \qquad \Gamma \vdash_{\Vdash} h : (U, J) \Vdash \varphi}{\Gamma \vdash_{\Vdash} h|_f : (V, J) \Vdash \varphi|_f} \qquad \frac{\Gamma \vdash_{\Vdash} f : V \longrightarrow_{\mathcal{C}} U \qquad \Gamma, \text{🔒}_{(U,J)} \vdash_i \varphi : \mathrm{Prop}}{\Gamma, \text{🔒}_{(V,J)} \vdash_i \varphi|_f : \mathrm{Prop}}$$

For the above, it may be possible to use mechanisms from adjoint and multimode type theories (Section 1.2.10) that support interacing modalities via the specification of a 2-category of modes.

One should furthermore expect to have an equational theory with laws such as $I_{(U,J)}(\mathrm{unbox}\ e) \equiv e$. However, I refrain from formulating one as I do not yet know what it should be (see below).

2.3.2. Soundness of elimination.

The point of $\sqsubset\exists\forall\mathbb{N}^{\Vdash}$ is to provide a convenient syntax for certain kinds of proofs. It should consequently be a conservative extension of Lean, in the sense that there exists a translation procedure $\underline{(-)}$ from $\sqsubset\exists\forall\mathbb{N}^{\Vdash}$ to Lean. In the implementation, a translation procedure consists of elaborators for the newly introduced syntax. For soundness,[12] we should have

$$\frac{\Gamma \vdash_{\Vdash} e : f}{\underline{\Gamma} \vdash_{\sqsubset\exists\forall\mathbb{N}} \underline{e} : \underline{f}} \qquad \frac{\Gamma \vdash_{\Vdash} e \equiv f : g}{\underline{\Gamma} \vdash_{\sqsubset\exists\forall\mathbb{N}} \underline{e} \equiv \underline{f} : \underline{g}}$$

However, it is not obvious how to ensure these properties. Terms that are internally definitionally equal, for instance connected by reduction, are generally only propositionally equal in the model, compromising soundness w.r.t. judgmental equality as formulated here. This, in turn, compromises soundness w.r.t. typing by the conversion rule. Identifying a "sufficiently intensional" formulation of $\sqsubset\exists\forall\mathbb{N}^{\Vdash}$ together with a translation for which the above (or a reformulation of the above) are true will be the first order of business in upcoming work. The resulting translation may be related to ones expressing the conservativity of extensional over intensional type theory [35,56,70].

Further complications include the fact that, if reusing the Lean typechecker for proof terms of the internal logic, we must ensure that HOL terms are judgmentally equal according to HOL if and only if they are equal according to Lean. This appears plausible as the theories are close: just like a subobject classifier, the universe `Prop` of propositions is proof-irrelevant, impredicative, and satisfies propositional extensionality. However, the type theory of Lean is intensional, so the internal logic has to forgo equality reflection despite it being validated in the model.

[12] An incorrect implementation of the translation cannot compromise soundness insofar as all its outputs are checked by the Lean kernel, but one may worry about the translation of *statements*: if every internal proposition is translated to `True`, and every proof to `trivial`, it may appear as if we have proven something when we have really not. That concern notwithstanding, correctness of translation is closer to a *liveness* property, ensuring that any expression that typechecks will be handled correctly.

2.4. Extensions and further applications

To conclude, I speculate on possible further work and related subprojects.

2.4.1. More expressive languages. The simple type theory internal to a topos may be incapable of carrying the weight of realistic mathematical developments. For those, it may be necessary to mechanize proof methods beyond standard Kripke-Joyal forcing, for instance Shulman's stack semantics [66], or forcing for extensional dependent type theory as developed by Awodey, Gambino, and Hazratpour [4]. Additionally, Kripke-Joyal forcing as presented in this document has a variation specialized to Grothendieck toposes known as *sheaf semantics*. Blechschmidt makes use of both stack and sheaf semantics. Nevertheless, simple types present themselves as a minimal usable target for a prototype. The present proposal aims to identify an approach to the mechanization of internal language interpretation and forcing in this spartan setting, with the hope that it later scales to more expressive languages.

2.4.2. Handling of algebraic structures. In a similar vein, a basic higher-order logic does not have any support for algebraic structures. That is, given generalized elements $m : X \to M^{M\times M}$ and $u : X \to M$, we are free to assume $X \Vdash \forall(x : M),\ m(x,u) = x$ and $X \Vdash \forall(x : M),\ m(u,x) = x$ together with the other monoid axioms, and prove other properties of X-local elements of M. However, this manual approach is not guaranteed to interact well with the meta-level `Monoid` typeclass; such a typeclass isn't expressible in an internal language without Σ-types.

Recent (as-yet unpublished) work by Topaz[13] is capable of automatically generating a presentation of the Lawvere theory corresponding to an algebraic structure from its definition as a Lean `structure`. This allows defining models of the theory in general finite product categories as functors out of the Lawvere theory. It may be possible and useful to incorporate tooling of this sort into the present project.

2.4.3. Staged metaprogramming. As of today, the most widely used library for type-safe staged metaprogramming in Lean 4 is Ebner's `quote4`.[14] Unfortunately, as of commit `64365c6`, `quote4` suffers from a failure of substitution analogous to that described in Section 1.2.3. Internally, the library elaborates eliminator-based vernacular syntax to `let`-patterns/delayed substitutions. For instance, the following example is accepted

```
import Qq
open Qq

example (a : α) (f : α → Q(Type)) (v : Q($(f a))) : Q($v = $v) :=
  q(sorry)
```

but `v : let a := f a; Q(unknown_1)` is displayed in the local context at the position of `sorry`. A residual deliverable of the proposed project could be to put the library on a firmer foundation, fixing admissible properties in the process.

2.4.4. Deinterpretation. The internal statements of certain properties are obvious: epimorphisms correspond to surjections. Others not so much: Blechschmidt [13] notes that a singular

[13] Described at https://leanprover.zulipchat.com/user_uploads/3121/1DDUZSIYeb2xCgAJ9koMJYgM/pres.pdf°.

[14] https://github.com/leanprover-community/quote4/tree/64365c656d5e1bffa127d2a1795f471529ee0178 °

difficulty in his development of synthetic algebraic geometry was relating external concepts to their internal formulations. Indeed, he singles out the building of such a dictionary as an important contribution.

The proposed system can only unbox ("deinterpret") proofs of statements syntactically (up to judgmental equality) in the image of $U \Vdash (-)$:

$$\frac{\Gamma \vdash_{\Vdash} e : (U, J) \Vdash \varphi}{\Gamma, \text{🔒}_{(U,J)}, \Gamma_i' \vdash_i \text{unbox } e : \varphi}$$

It might be interesting to semi-mechanically characterise this image. For instance, an interpreter could be used on numerous randomly-generated internal formulas to generate synthetic data consisting of pairs of the form $\left(\varphi, \widetilde{U \Vdash \varphi}\right)$ that would then be used to fit a statistical model with the purpose of computing the inverse function.

Bibliography

[1] Jean-Marc Andreoli. 1992. Logic Programming with Focusing Proofs in Linear Logic. *Journal of Logic and Computation* 2, (1992), 297–347. Retrieved from https://api.semanticscholar.org/CorpusID:205204264°

[2] Jeremy Avigad. 2022. *Mathematical Logic and Computation*. Cambridge University Press. https://doi.org/10.1017/9781108778756°

[3] S. Awodey. 2006. *Category Theory*. Ebsco Publishing.

[4] Steve Awodey, Nicola Gambino, and Sina Hazratpour. 2021. Kripke-Joyal forcing for type theory and uniform fibrations.

[5] Steve Awodey. 2017. Natural models of homotopy type theory.

[6] Roberta Ballarin. 2023. Modern Origins of Modal Logic. *The Stanford Encyclopedia of Philosophy*.

[7] P. N. Benton. 1994. A Mixed Linear and Non-Linear Logic: Proofs, Terms and Models (Extended Abstract). In *Computer Science Logic, 8th International Workshop, CSL '94, Kazimierz, Poland, September 25-30, 1994, Selected Papers (Lecture Notes in Computer Science)*, 1994. Springer, 121–135. https://doi.org/10.1007/BFB0022251°

[8] P. N. Benton and Philip Wadler. 1996. Linear Logic, Monads and the Lambda Calculus. In *Proceedings, 11th Annual IEEE Symposium on Logic in Computer Science, New Brunswick, New Jersey, USA, July 27-30, 1996*, 1996. IEEE Computer Society, 420–431. https://doi.org/10.1109/LICS.1996.561458°

[9] P. N. Benton, Gavin M. Bierman, Valeria de Paiva, and Martin Hyland. 1993. A Term Calculus for Intuitionistic Linear Logic. In *Typed Lambda Calculi and Applications, International Conference on Typed Lambda Calculi and Applications, TLCA '93, Utrecht, The Netherlands, March 16-18, 1993, Proceedings (Lecture Notes in Computer Science)*, 1993. Springer, 75–90. https://doi.org/10.1007/BFB0037099°

[10] G. M. Bierman. 1995. What is a categorical model of Intuitionistic Linear Logic?. In *Typed Lambda Calculi and Applications (Lecture Notes in Computer Science)*, 1995. Springer Berlin Heidelberg, Berlin, Heidelberg, 78–93. https://doi.org/10.1007/BFb0014046°

[11] Lars Birkedal, Ranald Clouston, Bassel Mannaa, Rasmus Ejlers Møgelberg, Andrew M. Pitts, and Bas Spitters. 2020. Modal dependent type theory and dependent right adjoints. *Math. Struct. Comput. Sci.* 30, 2 (2020), 118–138. https://doi.org/10.1017/S0960129519000197°

[12] Patrick Blackburn, Johan van Benthem, and Frank Wolter (Eds.). 2006. *Handbook of Modal Logic*. Elsevier.

[13] Ingo Blechschmidt. 2021. Using the internal language of toposes in algebraic geometry.

[14] V.A.J. Borghuis. 1994. Coming to Terms with Modal Logic: on the interpretation of modalities in typed λ-calculus. Technische Universiteit Eindhoven. https://doi.org/10.6100/IR427575°

[15] Oliver Byrne. 2022. *The First Six Books of the Elements of Euclid*. Taschen, Köln, Germany.

[16] Mario Carneiro. 2019. The Type Theory of Lean. Retrieved from https://github.com/digama0/lean-type-theory°

[17] Mario Carneiro. 2024. Lean4Lean: Towards a formalized metatheory for the Lean theorem prover.

[18] Ranald Clouston. 2018. Fitch-Style Modal Lambda Calculi. In *Foundations of Software Science and Computation Structures - 21st International Conference, FOSSACS 2018, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2018, Thessaloniki, Greece, April 14-20, 2018, Proceedings (Lecture Notes in Computer Science)*, 2018. Springer, 258–275. https://doi.org/10.1007/978-3-319-89366-2_14°

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

[20] The mathlib Community. 2020. The Lean Mathematical Library. In *Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs (CPP 2020)*, 2020. Association for Computing Machinery, New Orleans, LA, USA, 367–381. https://doi.org/10.1145/3372885.3373824°

[21] Rowan Davies and Frank Pfenning. 2001. A modal analysis of staged computation. *J. ACM* 48, 3 (2001), 555–604. https://doi.org/10.1145/382780.382785°

[22] William B. Ewald. 2000. *From Kant to Hilbert: A Source Book in the Foundations of Mathematics*. Clarendon Press.

[23] Solomon Feferman, Jr. John W. Dawson, Stephen C. Kleene, Gregory H. Moore, Robert M. Solovay, and Jean van Heijenoort. 1986. *Kurt Gödel: Collected Works: Volume I: Publications 1929-1936*. OUP USA.

[24] Frederic Brenton Fitch. 1952. *Symbolic Logic: An Introduction*. Ronald Press Company.

[25] Cormac Flanagan, Amr Sabry, Bruce F. Duba, and Matthias Felleisen. 1993. The Essence of Compiling with Continuations. In *Proceedings of the ACM SIGPLAN'93 Conference on Programming Language Design and Implementation (PLDI), Albuquerque, New Mexico, USA, June 23-25, 1993*, 1993. ACM, 237–247. https://doi.org/10.1145/155090.155113°

[26] Jean-Yves Girard and Yves Lafont. 1987. Linear Logic and Lazy Computation. In *TAPSOFT'87: Proceedings of the International Joint Conference on Theory and Practice of Software Development, Pisa, Italy, March 23-27, 1987, Volume 2: Advanced Seminar on Foundations of Innovative Software Development II and Colloquium on Functional and Logic Programming and Specifications (CFLP) (Lecture Notes in Computer Science)*, 1987. Springer, 52–66. https://doi.org/10.1007/BFB0014972°

[27] Jean-Yves Girard, Paul Taylor, and Yves Lafont. 1989. *Proofs and Types*. Cambridge University Press.

[28] Jean-Yves Girard. 1987. Linear logic. *Theoretical Computer Science* 50, 1 (1987), 1–101. https://doi.org/ https://doi.org/10.1016/0304-3975(87)90045-4°

[29] Jean-Yves Girard. 1993. On the Unity of Logic. *Annals of Pure and Applied Logic* 59, 3 (1993), 201–217. https://doi.org/10.1016/0168-0072(93)90093-s°

[30] Daniel Gratzer, Evan Cavallo, G. A. Kavvos, Adrien Guatto, and Lars Birkedal. 2022. Modalities and Parametric Adjoints. *ACM Trans. Comput. Log.* 23, 3 (2022), 1–29. https://doi.org/10.1145/3514241°

[31] Joel David Hamkins and Benedikt Loewe. 2005. The modal logic of forcing.

[32] Jesse Michael Han and Floris van Doorn. 2019. A Formalization of Forcing and the Unprovability of the Continuum Hypothesis. In *10th International Conference on Interactive Theorem Proving, ITP 2019, September 9-12, 2019, Portland, OR, USA (LIPIcs)*, 2019. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 1–19. https://doi.org/10.4230/LIPICS.ITP.2019.19°

[33] Jesse Michael Han and Floris van Doorn. 2020. A formal proof of the independence of the continuum hypothesis. In *Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs, CPP 2020, New Orleans, LA, USA, January 20-21, 2020*, 2020. ACM, 353–366. https://doi.org/10.1145/3372885.3373826°

[34] David Hilbert. 1977. *Foundations of Geometry*. Open Court Publishing, Chicago, IL, USA.

[35] Martin Hofmann. 1995. Conservativity of Equality Reflection over Intensional Type Theory. In *Types for Proofs and Programs, International Workshop TYPES'95, Torino, Italy, June 5-8, 1995, Selected Papers (Lecture Notes in Computer Science)*, 1995. Springer, 153–164. https://doi.org/10.1007/3-540-61780-9_68°

[36] Ralf Jung, Robbert Krebbers, Jacques-Henri Jourdan, Ales Bizjak, Lars Birkedal, and Derek Dreyer. 2018. Iris from the ground up: A modular foundation for higher-order concurrent separation logic. *J. Funct. Program.* 28, (2018), e20. https://doi.org/10.1017/S0956796818000151°

[37] Krzysztof Kapulkin and Peter LeFanu Lumsdaine. 2012. The Simplicial Model of Univalent Foundations (after Voevodsky). (2012).

[38] G. A. Kavvos. 2016. The Many Worlds of Modal λ-calculi: I. Curry-Howard for Necessity, Possibility and Time. *CoRR* (2016). Retrieved from http://arxiv.org/abs/1605.08106°

[39] Robbert Krebbers, Jacques-Henri Jourdan, Ralf Jung, Joseph Tassarotti, Jan-Oliver Kaiser, Amin Timany, Arthur Charguéraud, and Derek Dreyer. 2018. MoSeL: a general, extensible modal framework for interactive proofs in separation logic. *Proc. ACM Program. Lang.* 2, ICFP (2018), 1–30. https://doi.org/10.1145/3236772°

[40] Robbert Krebbers, Amin Timany, and Lars Birkedal. 2017. Interactive proofs in higher-order concurrent separation logic. In *Proceedings of the 44th ACM SIGPLAN Symposium on Principles of Programming Languages, POPL 2017, Paris, France, January 18-20, 2017*, 2017. ACM, 205–217. https://doi.org/10.1145/3009837.3009855°

[41] Saul A. Kripke. 1963. Semantical Analysis of Modal Logic I: Normal Modal Propositional Calculi. *Mathematical Logic Quarterly* 9, 5–6 (1963), 67–96.

[42] Neelakantan R. Krishnaswami, Cécilia Pradic, and Nick Benton. 2015. Integrating Linear and Dependent Types. In *Proceedings of the 42nd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2015, Mumbai, India, January 15-17, 2015*, 2015. ACM, 17–30. https://doi.org/10.1145/2676726.2676969°

[43] Leslie Lamport. 2002. *Specifying Systems: The TLA+ Language and Tools for Hardware and Software Engineers*. Addison-Wesley. Retrieved from https://www.microsoft.com/en-us/research/publication/specifying-systems-the-tla-language-and-tools-for-hardware-and-software-engineers/°

[44] Saunders Mac Lane and Ieke Moerdijk. 1994. Sheaves in Geometry and Logic. *Universitext* (1994). https://doi.org/10.1007/978-1-4612-0927-0°

[45] Clarence Irving Lewis. 1918. *A Survey of Symbolic Logic*. University of California Press.

[46] Daniel R. Licata and Michael Shulman. 2016. Adjoint Logic with a 2-Category of Modes. In *Logical Foundations of Computer Science - International Symposium, LFCS 2016, Deerfield Beach, FL, USA, January 4-7, 2016. Proceedings (Lecture Notes in Computer Science)*, 2016. Springer, 219–235. https://doi.org/10.1007/978-3-319-27683-0_16°

[47] Peter Lefanu Lumsdaine and Michael A. Warren. 2015. The Local Universes Model: An Overlooked Coherence Construction for Dependent Type Theories. *ACM Transactions on Computational Logic* 16, 3 (July 2015), 1–31. https://doi.org/10.1145/2754931°

[48] Hugh MacColl. 1906. *Symbolic Logic and Its Applications*. Longmans, Green,, Company.

[49] Simone Martini and Andrea Masini. 1996. A Computational Interpretation of Modal Proofs. In *Proof Theory of Modal Logic*, Heinrich Wansing (ed.). Springer Netherlands, Dordrecht, 213–241. https://doi.org/10.1007/978-94-017-2798-3_12°

[50] Eugenio Moggi. 1991. Notions of Computation and Monads. *Inf. Comput.* 93, 1 (1991), 55–92. https://doi.org/10.1016/0890-5401(91)90052-4°

[51] Richard Montague. 1960. Logical necessity, physical necessity, ethics, and quantifiers. *Inquiry* 3, 1–4 (1960), 259–269.

[52] Leonardo de Moura and Sebastian Ullrich. 2021. The Lean 4 Theorem Prover and Programming Language. In *Automated Deduction - CADE 28 - 28th International Conference on Automated Deduction, Virtual Event, July 12-15, 2021, Proceedings (Lecture Notes in Computer Science)*, 2021. Springer, 625–635. https://doi.org/10.1007/978-3-030-79876-5_37°

[53] Christopher Mulvey. 1974. Intuitionistic algebra and representations of rings. *Memoirs of the American Mathematical Society* 148 (1974), 3–57.

[54] Aleksandar Nanevski, Frank Pfenning, and Brigitte Pientka. 2008. Contextual modal type theory. *ACM Transactions on Computational Logic* 9, 3 (2008), 1–49. https://doi.org/10.1145/1352582.1352591°

[55] Masao Ohnishi and Kazuo Matsumoto. 1957. Gentzen Method in Modal Calculi. *Osaka Journal of Mathematics* 9, (1957), 113–130.

[56] Nicolas Oury. 2005. Extensionality in the Calculus of Constructions. In *Theorem Proving in Higher Order Logics, 18th International Conference, TPHOLs 2005, Oxford, UK, August 22-25, 2005, Proceedings (Lecture Notes in Computer Science)*, 2005. Springer, 278–293. https://doi.org/10.1007/11541868_18°

[57] Valeria de Paiva. 1989. The Dialectica Categories. *Categories in Computer Science and Logic* 92, (1989), 47–62.

[58] Frank Pfenning and Hao-Chi Wong. 1995. On a modal lambda calculus for S4. In *Eleventh Annual Conference on Mathematical Foundations of Programming Semantics, MFPS 1995, Tulane University, New Orleans, LA, USA, March 29 - April 1, 1995 (Electronic Notes in Theoretical Computer Science)*, 1995. Elsevier, 515–534. https://doi.org/10.1016/S1571-0661(04)00028-3°

[59] Brigitte Pientka and Jana Dunfield. 2010. Beluga: A Framework for Programming and Reasoning with Deductive Systems (System Description). In *Automated Reasoning, 5th International Joint Conference, IJCAR 2010, Edinburgh, UK, July 16-19, 2010. Proceedings (Lecture Notes in Computer Science)*, 2010. Springer, 15–21. https://doi.org/10.1007/978-3-642-14203-1_2°

[60] André Platzer. 2008. Differential Dynamic Logic for Hybrid Systems. *J. Autom. Reas.* 41, 2 (2008), 143–189. https://doi.org/10.1007/s10817-008-9103-8°

[61] Dag Prawitz. 1965. *Natural Deduction: A Proof-Theoretical Study*.

[62] Jason Reed. 2009. A Judgmental Deconstruction of Modal Logic. *Unpublished manuscript* (January 2009). Retrieved from https://www.cs.cmu.edu/~jcreed/papers/jdml.pdf°

[63] Emily Riehl. 2017. *Category Theory in Context*. Dover Publications.

[64] Halsey L. Royden and Patrick Fitzpatrick. 2018. *Real Analysis*. Pearson.

[65] R. A. G. Seely. 1989. Linear Logic, $*$-Autonomous Categories and Cofree Coalgebras. In *Categories in Computer Science and Logic (Contemporary Mathematics)*, 1989. American Mathematical Society, Boulder, Colorado, 371–382.

[66] Michael A. Shulman. 2010. Stack semantics and the comparison of material and structural set theories.

[67] Sebastian Ullrich and Leonardo de Moura. 2020. Beyond Notations: Hygienic Macro Expansion for Theorem Proving Languages. In *Automated Reasoning - 10th International Joint Conference, IJCAR 2020, Paris, France, July 1-4, 2020, Proceedings, Part II (Lecture Notes in Computer Science)*, 2020. Springer, 167–182. https://doi.org/10.1007/978-3-030-51054-1_10°

[68] Moshe Y. Vardi. 1996. Why is Modal Logic So Robustly Decidable?. In *Descriptive Complexity and Finite Models, Proceedings of a DIMACS Workshop 1996, Princeton, New Jersey, USA, January 14-17, 1996 (DIMACS Series in Discrete Mathematics and Theoretical Computer Science)*, 1996. DIMACS/AMS, 149–183. https://doi.org/10.1090/DIMACS/031/05°

[69] Andrea Vezzosi, Anders Mörtberg, and Andreas Abel. 2019. Cubical Agda: A Dependently Typed Programming Language with Univalence and Higher Inductive Types. *Proc. ACM Program. Lang.* 3, ICFP (2019), 1–29. https://doi.org/10.1145/3341691°

[70] Théo Winterhalter, Matthieu Sozeau, and Nicolas Tabareau. 2019. Eliminating reflection from type theory. In *Proceedings of the 8th ACM SIGPLAN International Conference on Certified Programs and Proofs, CPP 2019, Cascais, Portugal, January 14-15, 2019*, 2019. ACM, 91–103. https://doi.org/10.1145/3293880.3294095°