Towards a geometry for syntax

Jonathan Sterling

March 28, 2023

Abstract

It often happens that *free* algebras for a given theory satisfy useful reasoning principles that are not preserved under homomorphisms of algebras, and hence need not hold in an arbitrary algebra. For instance, if M is the free monoid on a set A, then the scalar multiplication function $A \times M \to M$ is injective. Therefore, when reasoning in the *formal theory of monoids* under A, it is possible to use this injectivity law to make sound deductions even about monoids under A for which scalar multiplication is not injective — a principle known in algebra as the *permanence of identity*. Properties of this kind are of fundamental practical importance to the logicians and computer scientists who design and implement computerized proof assistants like Lean and Coq, as they enable the formal reductions of equational problems that make type checking tractable.

As type theories have become increasingly more sophisticated, it has become more and more difficult to establish the useful properties of their free models that facilitate effective implementation. These obstructions have facilitated a fruitful return to foundational work in type theory, which has taken on a more geometrical flavor than ever before. Here we expose a modern way to prove a highly non-trivial injectivity law for free models of Martin-Löf type theory, paying special attention to the ways that contemporary methods in type theory have been influenced by three important ideas of the Grothendieck school: the *relative point of view*, the language of *universes*, and the *recollement* of generalized spaces.

Comment and acknowledgment This paper is an interpretation of the ideas of Awodey [Awo18], Bocquet, Kaposi, and Sattler [BKS21], Coquand [Coq19], Fiore [Fio02], Newstead [New18], Rijke, Shulman, and Spitters [RSS20], Shulman [Shu13; Shu15], Sterling [Ste21], and Uemura [Uem21; Uem22] as well as several other cited authors; the results described in this paper are not new, but their explanation might be. In addition to the cited authors, I am also greatly indebted to Mathieu Anel, Carlo Angiuli, Lars Birkedal, Daniel Gratzer, and Robert Harper for years of enlightening conversations on these topics. I thank also Chris Gossack for their helpful comments and suggestions.

1 Introduction

The purpose of this paper is to explain several ways in which the Grothendieck school has influenced theoretical computer science, focusing on the subdiscipline of *type theory* and the study of its free models.

1.1 Type theory and the *relative point of view*

Type theory is, of course, the study of Types; but much like other important scientific and philosophical categories such as Space and Number, there is not a single definition of what a Type is. Although the field of type theory is often said to have been born with Russell's investigations [Rus37; Rus08] into a syntactic way to avoid the eponymous "paradox", it must be said that type theory today has very little in common with this early line of research. Type theory in the sense studied by professionals is rather aimed to provide both informal and formal mathematical language to speak of objects and structures varying "continuously" over a base — in other words, to define the mathematical foundations to operationalize Grothendieck's ***relative point of view***.

> The ***relative point of view*** states that instead of studying (*e.g.*) schemes X in the absolute, we should always study ***relative*** schemes $X \in \mathbf{Sch}/B$ for an arbitrary base B.

Category theory implements the relative point of view by means of ***slice categories***; but this language is greatly obfuscated in comparison to the simplicity of working with non-relative objects. The goal of type theory is to reconcile the expressivity of the relative point of view with the simplicity of the global point of view, by providing a language that makes movement between different slices (base change) seamless. Because type theory is built up from very simple and abstract axiomatics, many categories of interest possess *type theoretic internal languages* which provide streamlined accounts of relative objects [Mai05].

Example 1.1 (Relative schemes, type theoretically)**.** We recall that a *relative scheme* over a scheme B is conventionally defined to be a morphism $E \longrightarrow B$ in the category of schemes. In contrast, the type theoretic viewpoint turns the display of E over B on its side: in the type theoretic internal language of the (gros) Zariski topos [Ble17], a "scheme" B is nothing more than a type satisfying certain conditions and a relative scheme over B is nothing more than a scheme $E(x)$ varying in a parameter $x : B$. The constraints of type theoretic language *automatically* ensure that all the $E(x)$ can be glued together to form a single scheme $\sum_{x:B} E(x)$ and moreover that the projection $\sum_{x:B} E(x) \longrightarrow B$ is in fact a genuine morphism of schemes. In this way, type theoretic language more directly captures the base intuitions of the relative point of view while minimizing bureaucratic overhead.

1.2 Universes in type theory and category theory

Type theory replaces the display of relative objects $E \longrightarrow B$ with *families* of objects $E(x)$ varying in a formal parameter $x : B$. This is achieved by postulating an imaginary "object of all objects" or a ***universal object*** such that B-indexed families of objects can be phrased in terms of functions *into* the universal object. To start with, this idea of postulating an imaginary universal object seems quite dangerous; for instance, if types are interpreted as sets then this postulate seems to imply a "set of all sets" in which indexed-families of sets can be valued. More concerningly, if types are interpreted as (*e.g.*) topological spaces, to postulate a universal object seems to imply a "topological space of all topological spaces", which makes even less sense than a set of all sets.

It is a fundamental result of the field of type theory, however, that the extension of a given theory by a universal object in this sense is ***conservative***.

Theorem 1.2 (Lumsdaine and Warren [LW15] and Awodey [Awo18])**.** *Let $\mathscr{C}$ be a category with a class of morphisms $\mathcal{D}$ stable under pullback. Then the Yoneda embedding $\boldsymbol{h} : \mathscr{C} \hookrightarrow \mathbf{Pr}\,\mathscr{C}$ of $\mathscr{C}$ into the (larger) category of presheaves on $\mathscr{C}$ has a **universal family** $\pi_{\mathcal{D}} : \mathsf{El}_{\mathcal{D}} \longrightarrow \mathsf{Tp}_{\mathcal{D}}$ such that every $p : E \longrightarrow B \in \mathcal{D}$ arises from it by pullback:*

$$\begin{array}{ccc} \boldsymbol{h}E & \overset{\bar{\chi}_p}{\dashrightarrow} & \mathsf{El}_{\mathcal{D}} \\ {\scriptstyle \boldsymbol{h}p}\downarrow & \lrcorner & \downarrow{\scriptstyle \pi_{\mathcal{D}}} \\ \boldsymbol{h}B & \underset{\chi_p}{\dashrightarrow} & \mathsf{Tp}_{\mathcal{D}} \end{array}$$

Moreover, every fiber of $\mathsf{El}_{\mathcal{D}}$ over a representable is represented by an element of $\mathcal{D}$.

The import of the fundamental result above is that relative objects qua morphisms $p : E \longrightarrow B$ in $\mathcal{D}$ can just as well be manipulated in terms of their characteristic morphisms $\chi_p : \boldsymbol{h}B \longrightarrow \mathsf{Tp}_{\mathcal{D}}$ into the universal object. It is in this sense that type theory turns the display of relative objects "on its side"; note that the assumptions of Theorem 1.2 are extremely light and are easily accommodated in many scenarios of interest, as we see below.

Example 1.3. The following are examples of categories $\mathscr{C}$ equipped with a class of maps $\mathcal{D}$ satisfying the assumptions of Theorem 1.2:

1. The category **Set**, **Esp**, or **Sch** equipped with the class of all maps.

2. The category **Esp** equipped with the class of fiber bundles, or of trivial bundles, *etc*.

3. The category of simplicial sets equipped with the class of Kan fibrations.

1.2.1 Strict base change via universal objects

The practical advantages of viewing relative objects in terms of morphisms into a universal object are noticed immediately: whereas base change of $p : E \longrightarrow B$ along $b : C \longrightarrow B$ must be implemented by pullback, the base change of the characteristic map is given more simply by precomposition $\chi_p \circ \boldsymbol{h}b : \boldsymbol{h}C \longrightarrow \mathsf{Tp}_{\mathcal{D}}$. The presentation in terms of precomposition is simpler to work with because it is *strictly associative and unital* in relation to base changes.

It is the strictness of base change qua precomposition that allows us to directly speak of the fibers of a parameterized object $(x : B) \mapsto E(x)$, since for any $f : C \to B$ and $g : D \to C$ the notation $E(f(gx))$ cannot distinguish between "first composing f with g and then doing base change" and "doing base change along f and then base change along g". When base change is implemented by pullback, these two ways to restrict $E \longrightarrow B$ to D are distinct but linked by a canonical isomorphism. The strength of type theory is to completely avoid the need to manipulate such canonical isomorphisms without sacrificing rigor.

1.2.2 Grothendieck's universes

As the terminology suggests, there is a great deal of similarity between the idea of universal objects and Grothendieck's universes, which he famously employed in SGA 4 to deal rigorously with the size issues that can quickly arise when using category theory to organize mathematics [AGV72]. Indeed, a given Grothendieck universe is a universal object for the class of maps in **Set** whose fibers have cardinality strictly lower than a given strongly inaccessible cardinal.

Grothendieck's universes were defined in terms of set theory and the $\in$-relation, but subsequent developments by several authors (including Bénabou, Martin-Löf, Hofmann, Streicher, and others) has led to a more structural perspective on universe objects that is amenable to formalization in an arbitrary category. The most influential input has been that of Jean Bénabou who had introduced already in his 1971 lectures the notion of a *universe in a topos* [Bén73, §6], which is essentially a *full internal subcategory* of the topos satisfying certain closure conditions, later interpreted and developed substantially further by Streicher [Str05] and several other authors.

1.2.3 Universes in a category

Definition 1.4. A ***universe*** $\mathcal{S}$ in a category $\mathscr{C}$ is given by a single carrable morphism $\pi_{\mathcal{S}} : \mathsf{El}_{\mathcal{S}} \longrightarrow \mathsf{Tp}_{\mathcal{S}}$ called the ***generic family***.[1] For a morphism $f : E \longrightarrow B$ in $\mathscr{C}$, we will write $f \in \mathcal{S}$ or "f is ***classified by*** $\mathcal{S}$" to mean that f arises by pullback from $\pi_{\mathcal{S}} : \mathsf{El}_{\mathcal{S}} \longrightarrow \mathsf{Tp}_{\mathcal{S}}$, *i.e.* there exists a cartesian map $f \longrightarrow \pi_{\mathcal{S}}$ in the fundamental (codomain) fibration $\mathbf{P}_{\mathscr{C}}$.

Remark 1.5. A universe $\mathcal{S}$ in a cartesian closed category determines an internal category in $\mathscr{C}$, whose object of objects is $\mathsf{Tp}_{\mathcal{S}}$ itself and whose object of morphisms

[1]A carrable morphism is one along which all pullbacks exist.

is the exponential $\pi_1^*\mathsf{El}_{\mathcal{S}} \Rightarrow \pi_2^*\mathsf{El}_{\mathcal{S}}$ over $\mathsf{Tp}_{\mathcal{S}} \times \mathsf{Tp}_{\mathcal{S}}$. The externalization of the internal category $\mathcal{S}$ is then a full subfibration of the fundamental fibration $\mathbf{P}_{\mathscr{C}}$. For each $I \in \mathscr{C}$, the fiber of this full subfibration is given by morphisms $e : E \longrightarrow I$ that are ***classified*** by $\mathcal{S}$. In other words $e : E \longrightarrow I$ lies in the full subfibration when there exists a cartesian morphism $e \longrightarrow \boldsymbol{\pi}_{\mathcal{S}}$.

Definition 1.6 (Contextual class of objects)**.** Let $\mathcal{S}$ be a universe in a category $\mathscr{C}$ with a terminal object; a class of objects $\mathcal{X} \subseteq \mathscr{C}$ is called ***$\mathcal{S}$-contextual*** when it satisfies the following closure conditions:

1. the terminal object is contained in $\mathcal{X}$;
2. if $C \in \mathscr{C}$ is contained in $\mathcal{X}$ and $p : A \longrightarrow C$ lies in $\mathcal{S}$, then A is in $\mathcal{X}$.

Definition 1.7 (Contextual objects)**.** Let $\mathcal{S}$ be a universe in a category $\mathscr{C}$ with a terminal object; an object of $\mathscr{C}$ is called ***$\mathcal{S}$-contextual*** when it is contained in the smallest $\mathcal{S}$-contextual class in the sense of Definition 1.6.

1.2.4 Grothendieck–Bénabou universes inside a topos

If $\mathscr{E}$ is an elementary topos with a natural numbers object N, following Bénabou [Bén73] and Streicher [Str05] we can define a notion of ***Grothendieck–Bénabou universe*** in $\mathscr{E}$ that restricts to the familiar notion of Grothendieck universe when $\mathscr{E} = \mathbf{Set}$.

Definition 1.8. A universe $\mathcal{S}$ in $\mathscr{E}$ is called a ***Grothendieck–Bénabou universe*** when it satisfies the following conditions:

1. ***dependent sums*** and ***dependent products***: if both $e : E \longrightarrow B$ and $b : B \longrightarrow C$ are classified by $\mathcal{S}$, then both $b_!e$ and b_*e are classified by $\mathcal{S}$, where $b_! \dashv b^* \dashv b_*$ is the base change adjoint triple.
2. ***propositional resizing***: every monomorphism of $\mathscr{E}$ is classified by $\mathcal{S}$.
3. ***descent***: for any g and f such that g is classified by $\mathcal{S}$, if there is a cartesian epimorphism $g \longrightarrow f$ in $\mathbf{P}_{\mathscr{E}}$, then f is classified by $\mathcal{S}$.
4. ***subobject classifier***: $\Omega \longrightarrow \mathbf{1}_{\mathscr{E}}$ is classified by $\mathcal{S}$.
5. ***natural numbers object***: $N \longrightarrow \mathbf{1}_{\mathscr{E}}$ is classified by $\mathcal{S}$.

1.3 Abstract and concrete syntax of type theory

So far we have discussed type theory as a convenient notation for working with relative objects in various categories. Most users of type theory will, in fact, need *no more* than this informal perspective on type theory. In order to more thoroughly justify these applications, however, type theorists have rendered the interpretation of type theoretical notations in various categories as consequences of a more general discourse on the *syntax and semantics of type theory* [Hof97].

There are many ways to think about what a model of type theory ought to be, but most of them take the form of categories $\mathscr{C}$ equipped with additional structure in $\mathbf{Pr}\,\mathscr{C}$, axiomatizing the scenario of Theorem 1.2. The syntax of type theory can be studied both abstractly and concretely; the ***concrete syntax*** of type theory can be defined in terms of a (very complex) formal grammar, but it is just as well to define the ***abstract syntax*** of a type theory to be given by the initial object in the category of models of that type theory. That abstract syntax can in fact be constructed as a quotient of concrete syntax is a consequence of the results of Cartmell [Car78], later tackled in more specificity by Streicher [Str91] and Uemura [Uem21]. Renewed interest during the past decade [Voe16] has led to several creative re-treadings of the ground first paved by Cartmell.

1.3.1 Computerized proof assistants

One motivation to study the syntax of type theory is to facilitate its *implemention* in computerized proof assistants; these are tools into which human beings can enter formal type theoretical expressions representing mathematical objects and proofs and have their validity automatically checked. In addition to assuring the validity of constructions and proofs, proof assistants also assist with book-keeping tasks — such as displaying what it remains to show at any given point in an incomplete proof. Proof assistants such as Coq [Coq16], Lean [Mou+15], and Agda [Nor09] are now routinely used to develop and mechanically check the correctness of both old and new mathematics [Gon08; Gon+13; Hou+16; Sch22], and there are now very extensive and mature libraries of mathematical results available [mat20; MT20; VAG+].

1.3.2 External *vs.* internal equality

Type theory is a somewhat unique language, in that it contains two different kinds of equality: external and internal. Type theory's ***external equality*** is simply the congruence under which assertions of the form $u : A$ are stable; in particular, when a type A is *externally equal* to a type B, written $A \equiv B$, we may assert $u : A$ if and only if we may assert $u : B$. True to its name, external equality cannot be assumed or refuted inside type theory; in other words, it is part of the *grammar* rather than the *vocabulary* of type theory.[2] In a *model of type theory* (including the initial model), external equality is interpreted as ordinary mathematical equality between elements of the model.

The second kind of equality in type theory is ***internal equality***, which is part of the vocabulary of type theory. For every type A and elements $u, v : A$ there is a third type $u =_A v$ that classifies *identifications* of u and v as elements of A. Internal equality is meant to correspond to ordinary mathematical equality; so, for instance, if the notion of a *group* is formalized in type theory, the unit laws

[2] External equality in our sense is usually referred to as ***judgmental equality*** or ***definitional equality***; both the traditional terminologies carry some philosophical force and subtlety that we do not necessarily intend, so we prefer our more neutral terminology. We refer the reader to Martin-Löf [Mar75a; Mar96; Mar87] for further discussion of the philosophical aspects.

are stated in terms of internal equality. In this paper, we will not dwell further on internal equality, in spite of the fact that it has been the main topic of type theoretic research for more than two decades [HS98; Voe06; AW09; Uni13].

1.3.3 Decidability of external equality

Although there are a variety of possible designs for computerized proof assistants based on type theory, experience has verified that the most practical approach is to ensure that the relation of external equality can be *automatically* checked by the computer without any intervention by the user. This goal, however, places severe constraints on what kinds of equations can be part of external equality — as it is easy for equality equality to become *undecidable* if enough laws are added [CCD17]. For this reason, type theorists have accumulated a variety of design principles that tend to ensure effective decidability — though it remains very difficult to establish decidability in any specific case.

1.3.4 Running example: injectivity of type constructors

In addition to decidability, one of the key lemmas servicing the computerized implementation of type theory is the ***injectivity of type constructors***, which is what allows an algorithm to universally decompose the task of checking an equation like $A \Rightarrow B \equiv A' \Rightarrow B'$ to the task of checking both $A \equiv A'$ and $B \equiv B'$: the injectivity property states that the latter judgments are the *only* way that the two function spaces could be equal. Note that injectivity in this sense does *not* imply that the $(\Rightarrow)$ operator is a monomorphism in an arbitrary model of type theory (indeed, doing so would rule out most semantic models of interest!). Nonetheless, the injectivity property can be stated in terms of $(\Rightarrow)$ being a monomorphism in the *initial model* of type theory. In fact, we shall use this injectivity law as our running example throughout the rest of this paper.

1.4 Normalization and injectivity, for free monoids

Type theorists have found that the most reliable way to establish both decidability of external equality (Section 1.3.3) and injectivity of type constructors (Section 1.3.4) is to devise a *concrete* characterization of equivalence classes of expressions in terms of ***normal forms***, equipping the quotient of concrete syntax by external equality with a more canonical section that is amenable to effective computation. This process is referred to as ***normalization***.

Normalization is better understood first in a simpler context; to that end, we consider the theory of monoids below and a similar injectivity law that we might wish to establish for *free* monoids.

1.4.1 The theory of monoids

The algebraic theory of monoids subjects a nullary operation ϵ and a binary operation μ to the following equational laws:

$$\begin{aligned} x &\vdash \mu(\epsilon, x) \equiv x \\ x &\vdash \mu(x, \epsilon) \equiv x \\ x, y, z &\vdash \mu(\mu(x, y), z) \equiv \mu(x, \mu(y, z)) \end{aligned}$$

1.4.2 Constructing the free monoid on a set

Given a set A, we may construct the *free monoid* on A by taking a quotient of the well-formed expressions in the theory of monoids with $|A|$-many additional constants. First we may inductively define the set $\mathsf{Expr}\,A$ of expressions by the generators:

$$\frac{a \in A}{\eta(a) \in \mathsf{Expr}\,A} \qquad \frac{}{\epsilon \in \mathsf{Expr}\,A} \qquad \frac{u \in \mathsf{Expr}\,A \quad v \in \mathsf{Expr}\,A}{\mu(u, v) \in \mathsf{Expr}\,A}$$

Next we inductively define $(\sim) \subseteq \mathsf{Expr}\,A \times \mathsf{Expr}\,A$ to be the smallest congruence for the operations above closed under the following rules:

$$\frac{u \in \mathsf{Expr}\,A}{\mu(\epsilon, u) \sim u} \qquad \frac{u \in \mathsf{Expr}\,A}{\mu(u, \epsilon) \sim u} \qquad \frac{u \in \mathsf{Expr}\,A \quad v \in \mathsf{Expr}\,A \quad w \in \mathsf{Expr}\,A}{\mu(\mu(u, v), w) \sim \mu(u, \mu(v, w))}$$

Then the carrier set of the free monoid on A can be expressed as the quotient $\mathsf{F}A = \mathsf{Expr}\,A/{\sim}$. Because $(\sim)$ is a congruence, there is an evident monoid structure on $\mathsf{F}A$ and it is simple to show that this monoid structure is universal in relation to monoids equipped whose carriers lie underneath A.

1.4.3 Injectivity of scalar multiplication in the free monoid

There is a "scalar multiplication" $A \times \mathsf{F}A \to \mathsf{F}A$ function on the free monoid sending (a, u) to $\mu(\eta(a), u)$. A monoid-theoretic analogue to our running example (Section 1.3.4) would be to prove that the scalar multiplication function on free monoids is injective. With our presentation of $\mathsf{F}A$ as a quotient of $\mathsf{Expr}\,A$, however, it is very hard to see that this is necessariy the case — as we do not have any kind of a handle on equivalence classes.

The solution is to find an alternative presentation of the free monoid that can be defined inductively without any quotienting; and such an alternative presentation is referred to a ***normal forms presentation***. In the case of free monoids, there is a trivial candidate for the normal forms presentation: the set of *lists* $A^\star$ of elements of A, which can be defined inductively as follows:

$$\frac{}{[] \in A^\star} \qquad \frac{a \in A \quad m \in A^\star}{a \triangleleft m \in A^\star}$$

In other words, $A^\star$ is the initial algebra for the polynomial endofunctor $F(X) = \mathbf{1} + A \times X$. There is no need to quotient $A^\star$; the monoid operations are defined by the following equations, using the induction principle of $A^\star$:

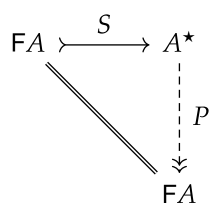
$$\begin{aligned} \eta_{A^\star}(a) &= a \triangleleft [] \\ \epsilon_{A^\star} &= [] \\ \mu_{A^\star}([], n) &= n \\ \mu_{A^\star}(a \triangleleft m, n) &= a \triangleleft \mu_{A^\star}(m, n) \end{aligned}$$

It is easy to show that $A^\star$ satisfies the equational laws of the monoid theory, again by induction on lists. But more importantly, it is possible to deduce immediately that the scalar multiplication on $A^\star$ is injective.

Theorem 1.9. *The scalar multiplication function $A \times A^\star \to A^\star$ sending each (a, m) to $\mu_{A^\star}(\eta_{A^\star}, m)$ is injective.*

Proof. Unfolding definitions, the scalar multiplication function is exactly the $(\triangleleft)$ operation on lists; writing $\alpha : \mathbf{1} + A \times A^\star \longrightarrow A^\star$ for structure map of $A^\star$ as an initial F-algebra, we recall that α is an isomorphism by Lambek's lemma, and so the constructor $(\triangleleft)$ is the right coproduct inclusion, which is injective as coproducts of sets are disjoint. $\square$

Therefore to deduce from Theorem 1.9 that the scalar multiplication function on $F_T A$ is injective, it suffices to construct an isomorphism of monoids under A between $F_T A$ and $A^\star$; in fact, it is even enough to exhibit $F_T A$ as a retract of $A^\star$, as depicted below where the horizontal arrow is the unique homomorphism of monoids under A determined by the universal property of $F_{\mathbb{T}}$:



Corollary 1.10. *The scalar multiplication function on the free monoid $\mathsf{F}A$ is injective.*

Proof. We define a retraction $P : A^\star \longrightarrow \mathsf{F}A$ of the universal map S, setting $P[] = \epsilon$ and $P(a \triangleleft m)$ to $\mu(\eta(a), P(m))$. Now fix $a, a' \in A$ and $u, u' \in \mathsf{F}A$ such that $\mu(\eta(a), u) = \mu(\eta(a'), u')$. Applying the section S and using the fact that it is a homomorphism, we have $a \triangleleft Su = a' \triangleleft Su'$; by Theorem 1.9 we have $a = a'$ and $Su = Su'$. From the latter we deduce $P(Su) = P(S'u)$; because P is a retraction of S, it follows that $u = u'$. $\square$

2 Free models of type theory and normalization

The normalization problem for free monoids that we explored in Section 1.4 is a particularly easy case. Unfortunately, things become significantly more difficult when we move from simple algebraic theories to full type theories, where we are trying to characterize the equivalence classes of *types* by normal forms; the difficulty is roughly that types and their normal forms do not (a priori) live in the same category, in contrast to the situation with monoids where both elements and normal forms are organized into sets.

2.1 Natural models of type theory

We have alluded in Section 1.3 to the many notions of "model of type theory"; here we will consider ***natural models*** [Awo18], a categorical reformulation of Dybjer's *categories with families* [Dyb96].

2.1.1 Representable maps and natural models

The definition of a natural model involves the concept of ***representable natural transformation***, which was incidentally introduced by Grothendieck and Dieudonné [GD60] in EGA 1.

Definition 2.1 (Relative representability)**.** Let $\mathscr{C}$ be a full subcategory of $\mathscr{E}$; a morphism $E \longrightarrow B$ of $\mathscr{E}$ is said to be ***relatively representable by an object of*** $\mathscr{C}$ when for any $\Gamma \longrightarrow B$ such that Γ lies in $\mathscr{C}$, the fiber product $E \times_B \Gamma$ lies in $\mathscr{C}$, identifying $\mathscr{C}$ with the image of the Yoneda embedding $\boldsymbol{h} : \mathscr{C} \hookrightarrow \mathbf{Pr}\,\mathscr{C}$.

Definition 2.2 (Awodey [Awo18])**.** A ***natural model*** $\mathbf{M}$ is defined to be an essentially small category $\mathscr{C}_{\mathbf{M}}$ with a terminal object and a natural transformation $\pi_{\mathbf{M}} : \mathsf{El}_{\mathbf{M}} \longrightarrow \mathsf{Tp}_{\mathbf{M}}$ in $\mathbf{Pr}\,\mathscr{C}_{\mathbf{M}}$ that is relatively representable by an object of $\mathscr{C}$.[3]

Observe that a natural model is really a special kind of *universe* (Definition 1.4) in a category of presheaves.

Exegesis 2.3. In a natural model $\mathbf{M}$, objects of $\mathscr{C}_{\mathbf{M}}$ are referred to as *contexts* and morphisms are called *substitutions*. When $\Gamma \in \mathscr{C}_{\mathbf{M}}$ is a context, an element of $\mathsf{Tp}_{\mathbf{M}}\Gamma$ is a *type* $A(\gamma)$ that depends on a parameter $\gamma : \Gamma$; the representability of $\pi_{\mathbf{M}}$ ensures for each type A over Γ, there is a context $\Gamma.A$ called the ***context comprehension*** classifying pairs (γ, a) where $\gamma : \Gamma$ and a is an element of $A\gamma$.

Definition 2.4 (Democratic natural models)**.** A natural model $\mathbf{M}$ is called ***democratic*** when every object $\Gamma \in \mathscr{C}_{\mathbf{M}}$ represents a $\mathbf{M}$-contextual object in the sense of Definition 1.7.

[3]Some previous expositions required $\mathscr{C}_{\mathbf{M}}$ to be small; nonetheless, the theory develops much more smoothly if we only require essential smallness.

2.1.2 Function spaces on a natural model

Further structures on a natural model can be imposed; for instance, function spaces correspond to cartesian squares of the following form in $\mathbf{Pr}\,\mathscr{C}_{\mathbf{M}}$:

$$\begin{array}{ccc}
\sum_{A,B:\mathsf{Tp}_{\mathbf{M}}} \pi_{\mathbf{M}}^{-1}A \Rightarrow \pi_{\mathbf{M}}^{-1}B & \xrightarrow{(\lambda_{\mathbf{M}})} & \mathsf{El}_{\mathbf{M}} \\
{\scriptstyle (A,B,f)\mapsto(A,B)}\downarrow & \lrcorner & \downarrow{\scriptstyle \pi_{\mathbf{M}}} \\
\mathsf{Tp}_{\mathbf{M}} \times \mathsf{Tp}_{\mathbf{M}} & \xrightarrow[(\Rightarrow_{\mathbf{M}})]{} & \mathsf{Tp}_{\mathbf{M}}
\end{array}$$

It can be shown that a natural model can be equipped with function spaces if and only if the corresponding universe is closed under pushforwards of product projection maps (this is a restriction of the condition that a Grothendieck–Bénabou universe be closed under dependent products).

2.1.3 The (2,1)-category of natural models

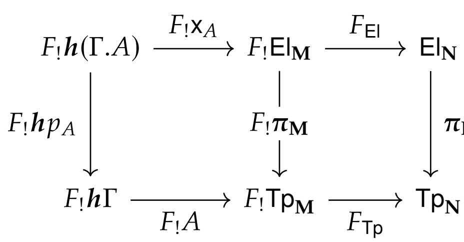
Natural models and their structured variants (*e.g.* natural models with function spaces, *etc.*) all arrange into (2,1)-categories. Here we will not dwell on the conditions for a morphism between natural models to extend to a morphism of natural models with function spaces; the interested reader should consult Uemura [Uem21] for more on this.

Definition 2.5 (Newstead [New18])**.** Let $\mathbf{M}$ and $\mathbf{N}$ be two natural models. A ***pre-morphism*** of natural models $F : \mathbf{M} \longrightarrow \mathbf{N}$ is given by a functor $F : \mathscr{C}_{\mathbf{M}} \longrightarrow \mathscr{C}_{\mathbf{N}}$ preserving the terminal object together with a square $\pi_F : F_!\pi_{\mathbf{M}} \longrightarrow \pi_{\mathbf{N}}$ in $\mathbf{Pr}\,\mathscr{C}_{\mathbf{N}}$:

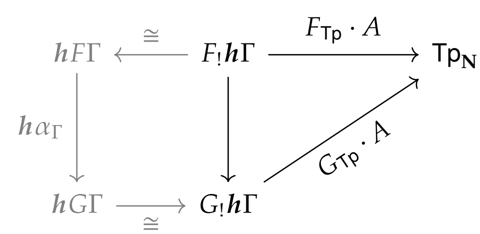
$$\begin{array}{ccc}
F_!\mathsf{El}_{\mathbf{M}} & \xrightarrow{F_{\mathsf{El}}} & \mathsf{El}_{\mathbf{N}} \\
{\scriptstyle F_!\pi_{\mathbf{M}}}\downarrow & \pi_F & \downarrow{\scriptstyle \pi_{\mathbf{N}}} \\
F_!\mathsf{Tp}_{\mathbf{M}} & \xrightarrow[F_{\mathsf{Tp}}]{} & \mathsf{Tp}_{\mathbf{N}}
\end{array}$$

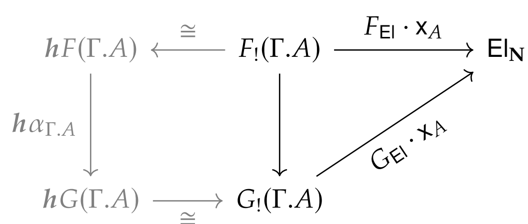
Notation 2.6. Given a pre-morphism $F : \mathbf{M} \longrightarrow \mathbf{N}$ and a type $\boldsymbol{h}\Gamma \longrightarrow \mathsf{Tp}_{\mathbf{M}}$, we shall write $F_{\mathsf{Tp}} \cdot A : F_!\boldsymbol{h}\Gamma \longrightarrow \mathsf{Tp}_{\mathbf{N}}$ for the composite $F_{\mathsf{Tp}} \circ F_!A$; we impose a similar notation on elements, setting $F_{\mathsf{El}} \cdot a$ to be $F_{\mathsf{El}} \circ F_!a$.

Definition 2.7 (Newstead [New18])**.** A pre-morphism $F : \mathbf{M} \longrightarrow \mathbf{N}$ is said to be a to be a ***morphism of natural models*** if it ***preserves context comprehensions*** in the sense that for every $\Gamma \in \mathscr{C}_{\mathbf{M}}$ and $A : \boldsymbol{h}\Gamma \longrightarrow \mathsf{Tp}_{\mathbf{M}}$, the composite square below is cartesian [New18]:



Definition 2.8 ([Uem21])**.** Let $F, G : \mathbf{M} \longrightarrow \mathbf{N}$ be two morphisms of natural models. An ***isomorphism*** from F to G is defined to be a natural isomorphism $\alpha : F \cong G$ between the underlying functors such that for each $A : \boldsymbol{h}\Gamma \longrightarrow \mathsf{Tp}_{\mathbf{M}}$, the black triangles below commute:





2.1.4 Free natural models: the abstract syntax of type theory

What is important for us is that the (2,1)-category of natural models (and its structured variants) is compactly generated or *presentable* in the sense of Lurie [Lur09] and therefore have *free objects*, *i.e.* the bi-initial natural model with function spaces on some constants, *etc*. Note that the bi-initial natural model of a given type theory is always democratic in the sense of Definition 2.4.

2.1.5 From universes to natural models

Let $\mathscr{E}$ be a locally small locally cartesian closed category, and let $\mathcal{S}$ be a universe in $\mathscr{E}$ such that $\mathbf{1}_{\mathscr{E}} \in \mathcal{S}$. Furthermore let $\mathscr{C} \subseteq \mathscr{E}$ be a full subcategory closed under all contextual objects with respect to $\mathcal{S}$ in the sense of Definition 1.7. In this section, we will define a natural model $\lceil \mathcal{S} \rceil_{\mathscr{C}}$ called the *externalization* of $\mathcal{S}$ over

$\mathscr{C}$. We define $\boldsymbol{\mathscr{C}}_{\lceil \mathcal{S} \rceil_{\mathscr{C}}}$ to be $\mathscr{C}$ itself. The inclusion functor $I : \mathscr{C} \hookrightarrow \mathscr{E}$ inducing a *nerve* $N_{\mathscr{C}} : \mathscr{E} \longrightarrow \mathbf{Pr}\,\mathscr{C}$ sending each object to its ***functor of*** $\mathscr{C}$***-valued points***:

$$N_{\mathscr{C}} : \mathscr{E} \longrightarrow \mathbf{Pr}\,\mathscr{C}$$
$$N_{\mathscr{C}} : E \longmapsto \mathbf{Hom}_{\mathscr{E}}(I-, E)$$

Lemma 2.9. *Let $p : E \longrightarrow B$ be a morphism in $\mathscr{E}$ that is relatively representable by an object of $\mathscr{C}$; then $N_{\mathscr{C}}\big(p : E \longrightarrow B\big)$ is a representable natural transformation in $\mathbf{Pr}\,\mathscr{C}$.*

Proof. We must check that for any $\Gamma \in \mathscr{C}$, the fiber product of any cospan $\boldsymbol{h}\Gamma \xrightarrow{A} N_{\mathscr{C}}B \xleftarrow{N_{\mathscr{C}}p} N_{\mathscr{C}}E$ is representable. Identifying $\boldsymbol{h}\Gamma$ with $N_{\mathscr{C}}I\Gamma$, we may assume without loss of generality that $A = N_{\mathscr{C}}A'$ for some $A' \longrightarrow I\Gamma B$. The fiber product of $I\Gamma \xrightarrow{A'} B \xleftarrow{p} E$ is *contextual* with respect to $\mathcal{S}$ by definition, and it lies in $\mathscr{C}$ by our assumption that $\mathscr{C}$ contains all contextual objects. □

We may therefore define the generic family $\boldsymbol{\pi}_{\lceil \mathcal{S} \rceil_{\mathscr{C}}}$ of $\lceil \mathcal{S} \rceil_{\mathscr{C}}$ to be $N_{\mathscr{C}}\boldsymbol{\pi}_{\mathcal{S}}$, which is representable by Lemma 2.9.

2.2 Injectivity of type constructors in free natural models

We now come to a precise version of our original discussion about injectivity of type constructors from Section 1.3.4.

Question 1. *Let $\mathbf{I}$ be the free natural model with function spaces generated by a base type* O *and two constants* $\mathsf{yes}, \mathsf{no} : \mathsf{O}$*. Is the function space constructor* $(\Rightarrow_{\mathbf{I}}) : \mathsf{Tp}_{\mathbf{I}} \times \mathsf{Tp}_{\mathbf{I}} \longrightarrow \mathsf{Tp}_{\mathbf{I}}$ *a monomorphism in* $\mathbf{Pr}\,\boldsymbol{\mathscr{C}}_{\mathbf{I}}$*?*

The answer to Question 1 is ultimately "Yes!" (Theorem 3.53), but this is as difficult to prove as Corollary 1.10 was easy. As we have alluded to at the beginning of Section 2, the problem is that although the collection of types is a presheaf on $\boldsymbol{\mathscr{C}}_{\mathbf{I}}$, we cannot very well define the collection of *normal forms of types* to be a presheaf on $\boldsymbol{\mathscr{C}}_{\mathbf{I}}$, as we will illustrate in Section 2.3.

2.3 Normal forms are not functorial in substitutions

The reason that a useful notion of normal form cannot be defined as a presheaf on $\boldsymbol{\mathscr{C}}_{\mathbf{I}}$ is that normal forms must distinguish between ***variables*** and the things that can be substituted for them. For instance, if $x : \mathsf{O} \Rightarrow \mathsf{O}$ represents a variable, then $x\ \mathsf{yes}$ should be represent normal form; but under the instantiation of x by $(\lambda z.z)$, the resulting expression $(\lambda z.z)\ \mathsf{yes}$ should *not* represent a normal form — the normal form representing for this expresion should simply be yes. It follows that the only way that normal forms could give rise to a presheaf on $\boldsymbol{\mathscr{C}}_{\mathbf{I}}$ is if *variables* gave rise to a presheaf on $\boldsymbol{\mathscr{C}}_{\mathbf{I}}$, but we will see that this will not obtain.

Definition 2.10. The ***variables*** in a natural model $\mathbf{M}$ are defined to be the smallest class of morphisms into $\mathsf{El}_{\mathbf{M}}$ such that for any $A : \boldsymbol{h}\Gamma \longrightarrow \mathsf{Tp}_{\mathbf{M}}$, the morphism $\mathsf{x}_A : \boldsymbol{h}(\Gamma.A) \longrightarrow \mathsf{El}_{\mathbf{M}}$ is a variable, and if $y : \boldsymbol{h}\Gamma \longrightarrow \mathsf{El}_{\mathbf{M}}$ is a variable than so is the composite $y \circ \boldsymbol{h}p_A : \boldsymbol{h}(\Gamma.A) \longrightarrow \mathsf{El}_{\mathbf{M}}$:

$$
\begin{array}{ccccc}
 & & \boldsymbol{h}(\Gamma.A) & \xrightarrow{\mathsf{x}_A} & \mathsf{El}_{\mathbf{M}} \\
 & \swarrow^{y \circ \boldsymbol{h}p_A} & \downarrow{\scriptstyle \boldsymbol{h}p_A} & \lrcorner & \downarrow{\scriptstyle \pi_{\mathbf{M}}} \\
\mathsf{El}_{\mathbf{M}} & \xleftarrow[y]{} & \boldsymbol{h}\Gamma & \xrightarrow[A]{} & \mathsf{Tp}_{\mathbf{M}}
\end{array}
$$

Problem 1 (Variables do not form a presheaf on $\mathscr{C}_{\mathbf{I}}$)**.** *If the collection of variables in the sense of Definition 2.10 formed a presheaf on $\mathscr{C}_{\mathbf{I}}$, then we could extend it to inductively define a presheaf normal forms satisfying our desired laws. We might try to define $\mathsf{Var}_{\mathbf{I}} \in \mathbf{Pr}\,\mathscr{C}_{\mathbf{I}}$ to assign to each context $\Gamma \in \mathscr{C}_{\mathbf{I}}$ the subset of $\mathbf{Hom}_{\mathbf{Pr}\,\mathscr{C}_{\mathbf{I}}}(\boldsymbol{h}\Gamma, \mathsf{El}_{\mathbf{I}})$ spanned by variables in the sense of Definition 2.10. This definition, however, is evidently not functorial in Γ: variables are closed under precomposition with projections $\Gamma.A \longrightarrow \Gamma$ and certain other maps derived from these, whereas functoriality in $\mathscr{C}_{\mathbf{I}}$ requires closure under precomposition with arbitrary maps.*

2.4 Models of variables and the method of computability

Although we have seen that the variables of a natural model $\mathbf{M}$ will not generally arrange themselves into a presheaf on $\mathscr{C}_{\mathbf{M}}$. Nonetheless, it is possible to imagine them forming a presheaf on a *different* category — perhaps, intuitively, a wide subcategory of $\mathscr{C}_{\mathbf{M}}$ that has fewer morphisms in it and thus induces a weaker functoriality condition.

If $\rho : \mathscr{R} \longrightarrow \mathscr{C}_{\mathbf{M}}$ represents the inclusion functor of such a wide subcategory on which the collection of variables forms a presheaf, then there is some hope for way to state define the collection of normal forms — not as a presheaf on $\mathscr{C}_{\mathbf{M}}$ but as a presheaf on $\mathscr{R}$. Of course, we must be able to link normal forms of types to the actual types they represent, so the collection of types $\mathsf{Tp}_{\mathbf{M}}$ must be imported into $\mathbf{Pr}\,\mathscr{R}$. This is easily done, however, by considering its restriction $\rho^*\mathsf{Tp}_{\mathbf{M}} \in \mathbf{Pr}\,\mathscr{R}$.

2.4.1 Models of variables over a natural model

The situation that we have intuitively described can be made more precise with the following more general notion of ***model of variables***.

Definition 2.11 (Bocquet, Kaposi, and Sattler [BKS21] and Uemura [Uem22])**.** A ***model of variables*** over a natural model $\mathbf{M}$ is defined to be a natural model $\mathbf{R}$ equipped with a homomorphism of natural models $\rho : \mathbf{R} \longrightarrow \mathbf{M}$ such that the induced map $\mathsf{Tp}_{\mathbf{R}} \longrightarrow \rho^*\mathsf{Tp}_{\mathbf{M}}$ is an isomorphism.

In the situation of Definition 2.11, then we may define $\mathscr{R}$ to be the underlying category $\mathscr{C}_{\mathbf{R}}$. Models of variables over $\mathbf{M}$ themselves arrange into a compactly generated (2,1)-category, and so we may consider the ***bi-initial model of variables*** over $\mathbf{I}$. In this case, $\mathsf{El}_{\mathbf{R}} \in \mathbf{Pr}\,\mathscr{C}_{\mathbf{R}}$ plays the role of the desired presheaf of variables; indeed, the bi-initiality property here corresponds to the *inductive* definition of variables (Definition 2.10).

Exegesis 2.12. The purpose of requiring $\mathsf{Tp}_{\mathbf{R}} \longrightarrow \rho^{*}\mathsf{Tp}_{\mathbf{M}}$ to be an isomorphism is to ensure that variables are classified by the same sorts of types as terms, and that the underlying functor $\rho : \mathscr{C}_{\mathbf{R}} \longrightarrow \mathscr{C}_{\mathbf{M}}$ is essentially surjective on objects. Note that even if $\mathbf{M}$ is structured (*e.g.* with function spaces, *etc.*), Definition 2.11 refers only to the bare structure of the natural model.

2.4.2 Why is it hard to build a model based on normal forms?

Recalling our construction of a normal forms presentation for free monoids in Section 1.4.3, we should be aiming to construct a natural model $\mathbf{N}$ containing normal forms equipped with a (pseudo-)retraction $P : \mathbf{N} \longrightarrow \mathbf{I}$ of the induced universal map $S : \mathbf{I} \longrightarrow \mathbf{N}$. Because we have a suitable notion of variable in $\mathbf{Pr}\,\mathscr{C}_{\mathbf{R}}$ it is tempting to attempt to define $\mathscr{C}_{\mathbf{N}} = \mathscr{C}_{\mathbf{R}}$ and then define $\mathsf{Tp}_{\mathbf{N}}$ to be a presheaf of normal forms of types and $\mathsf{El}_{\mathbf{N}}$ to be the presheaf of normal forms of elements. This proposal will fail almost immediately, however.

Problem 2. *When $\mathbf{R}$ is the bi-initial model of variables over $\mathbf{I}$, there are simply not enough morphisms in $\mathscr{C}_{\mathbf{R}}$ to build a model $\mathbf{N}$ of the full type theory* (e.g. *with function spaces) over it. For instance, the function space in $\mathbf{N}$ between two global types would necessarily induce an exponential between their context comprehensions in $\mathscr{C}_{\mathbf{R}}$, but this structure is not present in the bi-initial model of variables.*

A more promising idea to avoid Problem 2 is to let $\mathscr{C}_{\mathbf{N}}$ be a suitable full subcategory of $\mathbf{Pr}\,\mathscr{C}_{\mathbf{R}}$ closed under not only context comprehension from $\mathbf{R}$ but also the image of $\rho^{*} : \mathbf{Pr}\,\mathscr{C}_{\mathbf{I}} \longrightarrow \mathbf{Pr}\,\mathscr{C}_{\mathbf{R}}$. This doesn't work either, however.

Problem 3. *If we take $\mathscr{C}_{\mathbf{N}}$ to be a suitable full subcategory of $\mathbf{Pr}\,\mathscr{C}_{\mathbf{R}}$, then the resulting model cannot retain enough information about $\mathbf{I}$ to induce a pseudo-retraction $P : \mathbf{N} \twoheadrightarrow \mathbf{I}$ of the universal map $S : \mathbf{I} \longrightarrow \mathbf{N}$.*[4]

There is another problem besides the above with the idea of modeling types and terms by their normal forms, no matter what ambient category we may choose. Problem 4 below demonstrates that the problem of normalization for type theory with function spaces is vastly more difficult than that of (*e.g.*) the theory of monoids.

Problem 4. *In the presence of function spaces, the collections of normal forms cannot be used* directly *as a model. Roughly, the problem is that we would need to define* (e.g.) *an application function that takes a normal form of $u : A \to B$ and a normal form of $v : A$ to a normal form of $uv : B$, but this is exactly the problem we have been trying to solve in the first place — so we cannot define this function until we our proof is complete.*

[4]In fact, a normalization function can be defined in such a model [Fio22], but its correctness cannot be established without the pseudo-retraction $P : \mathbf{N} \twoheadrightarrow \mathbf{I}$.

2.4.3 Tait's method of computability

Problems 2 to 4 were first solved by Bill Tait, simultaneously, when he introduced the eponymous ***method of computability*** [Tai67], also variously known as *logical relations*, *logical predicates*, or the *reducibility method*.[5] Tait's brilliant solution to Problems 2 and 3, phrased in non-categorical language, was to devise a model in which a context is modeled as a predicate of some kind on a syntactic context; and a substitution is modeled by a syntactic substitution that preserves the corresponding predicates. Because every construct in the model is tracked by something syntactic, there is enough data to define a pseudo-retraction from the model onto the syntax. By imposing a further condition that the interpretation of every type be equipped with a *projection* onto normal forms, Tait solves Problem 4.

2.4.4 Freyd's categorical reconstruction of Tait computability

Tait's method was later rephrased by Peter Freyd in 1978 into categorical language as an instance of ***Artin gluing*** or ***recollement***, when he used it to give the first conceptual proof of the existence and disjunction properties in the free elmentary topos [Fre78]. Of course, Artin gluing was first introduced in SGA 4 as a way to reconstruct a topos from complementary open and closed subtopoi. Freyd considered only gluings along the global sections functor, whereas Tait's original situation (and ours) requires a more subtle gluing involving the functor that arises from a model of variables $\rho : \mathbf{R} \longrightarrow \mathbf{I}$. Scenarios of this kind were first investigated categorically by Jung and Tiuryn [JT93], Altenkirch, Hofmann, and Streicher [AHS95], Streicher [Str98], Fiore and Simpson [FS99], and Fiore [Fio02]. Our own "synthetic" approach to Tait's method, to be detailed in Section 3.1, is obtained by combining the observations of the cited authors with the viewpoint of the type theoretic internal language (Section 1.1) of glued topoi.

3 Normalization by gluing for free natural models

3.1 Synthetic Tait computability for models of type theory

We have seen in Section 2.4 that the normalization problem for type theory hinges on the concept of a *variable*, and introduced a technical notion of "model of variables" on a natural model (Definition 2.11) that can serve as a matrix in which to define normal forms. As we pointed out in Section 2.4.2, this is not enough to prove that normal forms adequately represent the constructs of the bi-initial natural model $\mathbf{I}$ of type theory.

The fundamental issue, exposed in Problems 2 and 3, is that any normalization model $\mathbf{N}$ must be structured with a homomorphism $\mathbf{N} \twoheadrightarrow \mathbf{I}$, which shall be seen to be a pseudo-retraction of the induced universal map $\mathbf{I} \rightarrowtail \mathbf{N}$ by an application

[5] In addition to Tait's original contribution, several other authors contributed greatly to the early development (and naming) of this concept, including for example Girard [Gir71], Martin-Löf [Mar75a; Mar75b; Mar71], Plotkin [Plo73; Plo80], Prawitz [Pra71], and Statman [Sta85].

of the latter's universal property; concretely, this means that both contexts and substitutions of the normalization model $\mathbf{N}$ must not forget the contexts and substitutions from the bi-initial model to which they pertain.

The reason the pseudo-retraction $\mathbf{N} \twoheadrightarrow \mathbf{I}$ is needed is the same as in our simpler example for free monoids (Section 1.4, Corollary 1.10): the universal map $\mathbf{I} \rightarrowtail \mathbf{N}$ sends each piece of term X to a construct of the normalization model from which we might expect to extract a normal form, and the purpose of the pseudo-retraction is to ensure that the resulting normal form is a normal form for X, rather than a normal form for some other term.

Bill Tait's solution to these problems was to define models that compositionally instrument syntactic constructs with additional data, namely the data of "normalizability" or "computability". In this section, we will see how the categorical reconstruction of Tait's method of computability arises naturally from the idea of formally gluing the constructs of bi-initial model $\mathbf{I}$ along the restriction functor $\rho^* : \mathbf{Pr}\,\mathscr{C}_{\mathbf{I}} \longrightarrow \mathbf{Pr}\,\mathscr{C}_{\mathbf{R}}$ onto data valued in the model of variables $\mathbf{R}$, leading to a notion of ***computability space*** from which a normalization model can be extracted by means of an application of the ***functor of points***.

3.1.1 The topos of computability spaces over a model of variables

Let $\rho : \mathbf{R} \longrightarrow \mathbf{M}$ be a model of variables over a natural model $\mathbf{M}$. We will first show each aspect of the model of variables translates into the geometric language of topoi. In particular, the two categories of presheaves $\mathbf{Pr}\,\mathscr{C}_{\mathbf{M}}$ and $\mathbf{Pr}\,\mathscr{C}_{\mathbf{R}}$ are topoi; in this paper, we are careful to distinguish the geometrical and algebraic aspects of topoi [AJ21; Vic07; BF06], so we shall write $\mathsf{E}_{\mathbf{M}}$ and $\mathsf{E}_{\mathbf{R}}$ for the topoi corresponding the two categories of presheaves respectively.

Notation 3.1. Given a topos X, we will write $\mathcal{S}_{\mathsf{X}}$ for the corresponding category; for example, we have $\mathcal{S}_{\mathsf{E}_{\mathbf{M}}} = \mathbf{Pr}\,\mathscr{C}_{\mathbf{M}}$. We refer to an object of $\mathcal{S}_{\mathsf{X}}$ as a *sheaf on* X.

Observation 3.2. *The underlying functor $\rho : \mathscr{C}_{\mathbf{R}} \longrightarrow \mathscr{C}_{\mathbf{M}}$ of our model of variables corresponds to an **essential morphism** of topoi $\boldsymbol{\rho} : \mathsf{E}_{\mathbf{R}} \longrightarrow \mathsf{E}_{\mathbf{M}}$ given by the adjoint triple induced by base change of presheaves:*

$$\mathbf{Pr}\,\mathscr{C}_{\mathbf{M}} \underset{\rho_*}{\overset{\rho_!}{\leftleftarrows}} \xrightarrow{\ \rho^*\ } \mathbf{Pr}\,\mathscr{C}_{\mathbf{R}} \qquad (\rho_! \dashv \rho^* \dashv \rho_*)$$

Lemma 3.3. *When $\mathbf{M}$ is* democratic *in the sense of Definition 2.4, the essential morphism $\boldsymbol{\rho} : \mathsf{E}_{\mathbf{R}} \longrightarrow \mathsf{E}_{\mathbf{M}}$ induced by the model of variables is a **surjection** of topoi.*

Proof. It can be shown that the underlying functor $\rho : \mathscr{C}_{\mathbf{R}} \longrightarrow \mathscr{C}_{\mathbf{M}}$ of a model of variables is essentially surjective on objects when $\mathscr{C}_{\mathbf{M}}$ is democratic. This is enough to see that the precomposition functor ρ^* is faithful, and so $\boldsymbol{\rho} = (\rho^* \dashv \rho_*)$ is surjective. □

Our goal is to classify a notion of *computability space* that instruments the constructs of $\mathbf{M}$ with data from $\mathbf{R}$; the fundamental example of a computability space would then be the space of normal forms of types: in this example, one instruments types that live in $\mathbf{M}$ with normal forms that live in $\mathbf{R}$. First we will define precisely what a computability space is, and then we will observe that that we may construct a topos $\mathbf{G}$ by Artin gluing whose sheaves are exactly the computability spaces.

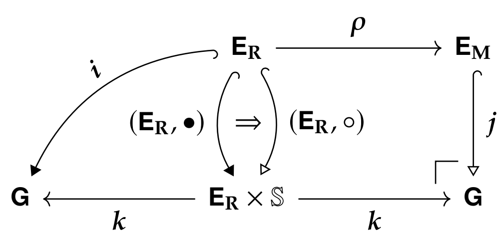
Definition 3.4. A ***computability space*** is given by a presheaf $E \in \mathbf{Pr}\,\mathscr{C}_{\mathbf{M}}$ together with a family of presheaves $\pi_E : \tilde{E} \longrightarrow \rho^*E \in \mathbf{Pr}\,\mathscr{C}_{\mathbf{R}}$. A morphism from a computability space (E, π_E) to a computability space (F, π_F) is given by a morphism $f : E \longrightarrow F \in \mathbf{Pr}\,\mathscr{C}_{\mathbf{M}}$ together with a morphism $\tilde{f} : \tilde{E} \longrightarrow \tilde{F} \in \mathbf{Pr}\,\mathscr{C}_{\mathbf{R}}$ such that the following square commutes:

$$\begin{array}{ccc} \tilde{E} & \xrightarrow{\tilde{f}} & \tilde{F} \\ {\scriptstyle \pi_E}\downarrow & & \downarrow{\scriptstyle \pi_F} \\ \rho^*E & \xrightarrow[\rho^*f]{} & \rho^*F \end{array}$$

Construction 3.5. We define the ***topos of computability spaces*** to be the following co-comma object in the bicategory of Grothendieck topoi below:

$$\begin{array}{ccc} \mathbf{E}_{\mathbf{R}} & \xrightarrow{\rho} & \mathbf{E}_{\mathbf{M}} \\ \Vert & \overset{\pi}{\Rightarrow} & \downarrow{\scriptstyle j} \\ \mathbf{E}_{\mathbf{R}} & \xrightarrow[i]{} & \mathbf{G} \end{array} \tag{1}$$

In Diagram 1, the morphism $j : \mathbf{E}_{\mathbf{M}} \hookrightarrow \mathbf{G}$ is a ***open immersion*** of topoi, and the $i : \mathbf{E}_{\mathbf{R}} \hookrightarrow \mathbf{G}$ is its complementary ***closed immersion***. This gluing can be computed more explicitly as a ***closed mapping cylinder*** using the Sierpiński interval $\mathbb{S} = \{\bullet \longrightarrow \circ\}$, as in Johnstone [Joh77]:



(2)

Observation 3.6. *Under the geometry–algebra duality, the co-comma topos* G *corresponds to the comma category* $\boldsymbol{\mathcal{S}}_{\mathsf{G}} \simeq (\mathbf{Pr}\,\mathscr{C}_{\mathbf{R}} \downarrow \boldsymbol{\rho}^*)$, i.e. *the* ***Artin gluing*** *of* $\boldsymbol{\rho}^*$. *Therefore, sheaves on* G *are the same thing as computability spaces* qua *Definition 3.4, and morphisms between sheaves are exactly morphisms of computability spaces.*

Lemma 3.7. *Both the open and closed immersions are essential,* i.e. *we have additional (necessarily fully faithful) left adjoints* $\boldsymbol{j}_! \dashv \boldsymbol{j}^*$ *and* $\boldsymbol{i}_! \dashv \boldsymbol{i}^*$.

Proof. That the open immersion is essential follows from the fact that $\mathbf{Pr}\,\mathscr{C}_{\mathbf{R}}$ has an initial object; that the closed immersion is essential follows from the fact that $\boldsymbol{\rho} : \mathsf{E}_{\mathbf{R}} \longrightarrow \mathsf{E}_{\mathbf{M}}$ is essential. Finally, in an adjoint triple $F \dashv G \dashv H$, the leftmost adjoint is fully faithful if and only if the rightmost one is. ◻

Exegesis 3.8. Under the identification of sheaves on G with computability spaces, we may examine the behavior of the adjoint triples corresponding to the essential open and closed immersions:

$$
\begin{array}{lll}
\boldsymbol{j}_* & : & E \mapsto (E, \mathbf{id}_{\boldsymbol{\rho}^* E}) \\
\boldsymbol{i}_* & : & R \mapsto \left(\mathbf{1}_{\mathbf{Pr}\,\mathscr{C}_{\mathbf{M}}}, !_R : R \longrightarrow \boldsymbol{\rho}^* \mathbf{1}_{\mathbf{Pr}\,\mathscr{C}_{\mathbf{M}}}\right) \\
\\
\boldsymbol{j}^* & : & (E, \pi_E) \mapsto E \\
\boldsymbol{i}^* & : & (E, \pi_E) \mapsto \tilde{E} = \operatorname{dom} \pi_E \\
\\
\boldsymbol{j}_! & : & E \mapsto \left(E, !_{\boldsymbol{\rho}^* E} : \mathbf{0}_{\mathbf{Pr}\,\mathscr{C}_{\mathbf{R}}} \longrightarrow \boldsymbol{\rho}^* E\right) \\
\boldsymbol{i}_! & : & R \mapsto \left(\boldsymbol{\rho}_! R, \eta_R : R \longrightarrow \boldsymbol{\rho}^* \boldsymbol{\rho}_! R\right)
\end{array}
$$

Exegesis 3.9. The additional left adjoint $\boldsymbol{i}_! \dashv \boldsymbol{i}^*$ will play an important role; it is uniquely determined by the property of sending representables Γ to a space of "variable renamings" for Γ in $\boldsymbol{\mathcal{S}}_{\mathsf{G}}$. If we think of Γ as a context, then an element of $\boldsymbol{i}_! \boldsymbol{h}\Gamma$ can be thought of as representing a sequence of variables that can be substituted for those classified by Γ.

Lemma 3.10. *The following square commutes up to isomorphism:*

$$
\begin{array}{ccccc}
\mathscr{C}_{\mathbf{R}} & \xrightarrow{\boldsymbol{h}_{\mathscr{C}_{\mathbf{R}}}} & \mathbf{Pr}\,\mathscr{C}_{\mathbf{R}} & \xrightarrow{\boldsymbol{i}_!} & \boldsymbol{\mathcal{S}}_{\mathsf{G}} \\
{\scriptstyle \rho}\downarrow & & & & \downarrow{\scriptstyle \boldsymbol{j}^*} \\
\mathscr{C}_{\mathbf{M}} & & \xrightarrow[\boldsymbol{h}_{\mathscr{C}_{\mathbf{M}}}]{} & & \mathbf{Pr}\,\mathscr{C}_{\mathbf{M}}
\end{array}
$$

3.1.2 Recollement of computability spaces

SGA 4 explains how the construction of G by gluing along $\boldsymbol{\rho} : \mathsf{E}_{\mathbf{R}} \longrightarrow \mathsf{E}_{\mathbf{M}}$ corresponds, in reverse, to the *partitioning* of the topos G into complementary open and closed subtopos [AGV72]. Under the latter viewpoint, the open and closed immersions become identified with the *inclusion* of the corresponding open and closed subtopoi. We will use this perspective to develop a more convenient language for constructing computability spaces intrinsically in the language of $\mathcal{S}_{\mathbf{G}}$ without bothering with the complex families of presheaves by which we originally defined computability spaces (Definition 3.4).

Definition 3.11 (Opens of a topos)**.** An *open* of a topos $\mathbf{X}$ is defined to be a subterminal sheaf on that topos, *i.e.* a subobject of $\mathbf{1}_{\mathcal{S}_{\mathbf{X}}}$. We will write $\mathcal{O}_{\mathbf{X}} \subseteq \mathcal{S}_{\mathbf{X}}$ for the poset (frame, in fact) of opens of $\mathbf{X}$.

Definition 3.12. Let $U \in \mathcal{O}_{\mathbf{X}}$ be an open of a topos $\mathbf{X}$; then a sheaf E is called U***-modal*** when the canonical map $E \longrightarrow E^U$ is an isomorphism. Conversely, a sheaf E is called U***-connected*** when the projection map $E \times U \longrightarrow U$ is an isomorphism.

Fact 3.13 (U-modal and U-connected reflection)**.** *For an open $U \in \mathcal{O}_{\mathbf{X}}$, the full subcategories of $\mathcal{S}_{\mathbf{X}}$ spanned by U-modal and U-connected sheaves are reflective.*

1. *The U-modal reflection of a sheaf $X \in \mathcal{S}_{\mathbf{X}}$ is given by the exponential X^U.*
2. *The U-connected reflection of a sheaf $X \in \mathcal{S}_{\mathbf{X}}$ is given by the pushout of the product span $X \xleftarrow{\pi_1} X \times U \xrightarrow{\pi_2} U$.*

The U-modal and U-connected reflections preserve finite limits. Therefore, the full subcategories of U-modal and U-connected sheaves present *subtopoi*; the subtopos of U-modal sheaves is referred to as the ***open subtopos*** determined by U and the subtopos of U-connected sheaves is referred to as the ***closed subtopos*** determined by U.

Fact 3.14 (Open subtopos as slice)**.** *Based on the definition of the U-modal reflector, it is not difficult to see that the slice category $\mathcal{S}_{\mathbf{X}}/U$ may be canonically identified with the category of sheaves on the open subtopos of $\mathbf{X}$ determined by U.*

Observation 3.15 (Lawvere–Tierney topologies)**.** *The open and closed subtopi can equivalently be described by Lawvere–Tierney topologies on $\mathbf{X}$, which simply internalize the reflectors as endomaps of the subobject classifier.*

1. *The topology of the open subtopos is given by the map $j_{/U}\phi = U \Rightarrow \phi$.*
2. *The topology of the closed subtopos is given by $j_{\backslash U}\phi = U \vee \phi$*

Construction 3.16 (Recollement of the topos of computability spaces)**.** What we take from SGA 4 [AGV72] is that up to categorical equivalence, we may reconstruct the gluing data for our own topos $\mathbf{G}$ from a certain open $\Phi \in \mathcal{O}_{\mathbf{G}}$, which can be equivalently described by either $\Phi = \boldsymbol{i}_*\bot$ or $\Phi = \boldsymbol{j}_!\top$. As a subterminal computability space, Φ is the family $\left(\mathbf{1}_{\mathbf{Pr}\,\mathscr{C}_{\mathbf{M}}}, \mathbf{0}_{\mathbf{Pr}\,\mathscr{C}_{\mathbf{R}}} \longrightarrow \boldsymbol{\rho}^*\mathbf{1}_{\mathbf{Pr}\,\mathscr{C}_{\mathbf{M}}}\right)$. It is then not difficult to see the following:

1. The essential image of $\boldsymbol{j}_* : \mathbf{Pr}\,\mathscr{C}_{\mathbf{M}} \hookrightarrow \mathcal{S}_{\mathbf{G}}$ is exactly the full subcategory spanned by Φ-modal computability spaces. Under this identification, the Φ-modal reflection takes a computability space X to $\boldsymbol{j}^*X$.

2. The essential image of $\boldsymbol{i}_* : \mathbf{Pr}\,\mathscr{C}_{\mathbf{R}} \hookrightarrow \boldsymbol{\mathcal{S}}_{\mathsf{G}}$ is exactly the full subcategory spanned by Φ-connected computability spaces. Under this identification, the Φ-connected reflection takes a computability space X to $\boldsymbol{i}^*X$.

Finally, we have a functor from Φ-modal sheaves to Φ-connected sheaves taking Φ-modal E to the Φ-connected reflection of E. Under the identifications above, this functor is isomorphic to $\boldsymbol{\rho}^* : \mathbf{Pr}\,\mathscr{C}_{\mathbf{M}} \longrightarrow \mathbf{Pr}\,\mathscr{C}_{\mathbf{R}}$. Thus the open $\Phi \in \boldsymbol{O}_{\mathsf{G}}$ controls *all* the gluing data of G except for the additional fact that $\boldsymbol{\rho}^*$ happens to be the inverse image component of an essential morphism of topoi.

Fact 3.17 (Recollement of computability spaces)**.** *Just as Construction 3.16 shows that the topos of computability spaces can be reconstructed from the induced open and closed subtopoi, something similar can be said of each individual computability space. In particular, for any $X \in \boldsymbol{\mathcal{S}}_{\mathsf{G}}$ the following square is always cartesian:*

$$
\begin{array}{ccc}
X & \xrightarrow{\ \eta_X^{\boldsymbol{i}_*\boldsymbol{i}^*}\ } & \boldsymbol{i}_*\boldsymbol{i}^*X \\
{\scriptstyle \eta_X^{\boldsymbol{j}_*\boldsymbol{j}^*}}\big\downarrow & \lrcorner & \big\downarrow{\scriptstyle \boldsymbol{i}_*\boldsymbol{i}^*\eta_X^{\boldsymbol{j}_*\boldsymbol{j}^*}} \\
\boldsymbol{j}_*\boldsymbol{j}^*X & \xrightarrow[\ \eta_{b\boldsymbol{j}_*\boldsymbol{j}^*X}^{\boldsymbol{i}_*\boldsymbol{i}^*}\ ]{} & \boldsymbol{i}_*\boldsymbol{i}^*\boldsymbol{j}_*\boldsymbol{j}^*X
\end{array}
$$

The import of Fact 3.17 is that it shows that any sheaf on G can be constructed entirely in terms of (left exact, idempotent) monads on $\boldsymbol{\mathcal{S}}_{\mathsf{G}}$ without bringing either $\mathbf{Pr}\,\mathscr{C}_{\mathbf{M}}$ nor $\mathbf{Pr}\,\mathscr{C}_{\mathbf{R}}$ into the picture.

3.1.3 The internal language of computibility spaces

Although we will not expose them all in this paper, there are a number of somewhat technical constructions of computability spaces that must ultimately be carried out. As these constructions are *relative* in nature and must constantly move between slices of $\boldsymbol{\mathcal{S}}_{\mathsf{G}}$, we may simplify things considerably by recalling from Section 1.1 that ***type theoretic internal languages*** are the appropriate linguistic foundation for the relative point of view.

It happens that *all* the constructions of Section 3.1.2 are stable under slicing, and can therefore be incorporated in a type theoretic internal language. As a result, we may rephrase the results of Section 3.1.2 as statements in the internal language of $\boldsymbol{\mathcal{S}}_{\mathsf{G}}$ by adopting the following single postulate:

Postulate 1. *There exists a proposition Φ,* i.e. *a type Φ satisfying the condition that every two of its elements are equal. We will write $\bigcirc$ for the reflection of Φ-modal types; we will write $\bullet$ for the reflection of Φ-connected types. We additionally assume that $\bigcirc A = A$ strictly when $\Phi = \top$.*[6]

[6]This final assumption can be removed, but it is convenient for our presentation.

Notation 3.18 (Subuniverses of modal types)**.** Given a universe $\mathcal{U}$, we will write $\mathcal{U}_\circ$ and $\mathcal{U}_\bullet$ for the subuniverses spanned by Φ-modal and Φ-connected types respectively. Note that unlike in univalent foundations [RSS20], it is not the case that $\mathcal{U}_\bullet$ is itself Φ-connected nor that $\mathcal{U}_\circ$ is Φ-modal.

The language of type theory extended by Postulate 1 is referred to by Sterling [Ste21] as ***synthetic Tait computability***, because it generates as if from the void an abstract form of Tait's computability out of the dynamics of Φ-modal and Φ-connected types in the internal language, as these correspond under the computability spaces interpretation to the syntactic components and their semantic instrumentations respectively. Indeed, the internal / type theoretic version of the recollement of computability spaces (Fact 3.17) is the following Observation 3.19, formally deducible in synthetic Tait computability.

Observation 3.19 (Recollement of computability spaces, synthetically)**.** *For any type X, the canonical "fracture function" defined below is an isomorphism:*

$$\begin{aligned} X &\to \textstyle\sum_{x_0:\circ X}\{x_1 : \bullet X \mid \eta^\bullet_{\circ X}x_0 =_{\bullet\circ X} \bullet\eta^\circ_X x_1\} \\ x &\mapsto (\eta^\circ_X x, \eta^\bullet_X x) \end{aligned}$$

Notation 3.20. For any Φ-modal type A, the map $k_A : A \to (\Phi \to A)$ is invertible by definition. We will permit the following abuse of notation: when constructing an element of a Φ-modal type A, we will write $\wr a \int$ to mean $k_A^{-1}(\lambda\_.a)$. Thus inside the delimiter, we implicity bind a variable $\_ : \Phi$.

Notation 3.21 (Extension types)**.** Let $A : \mathcal{U}$ be a type and let $\_ : \Phi \vdash a : A$ a be partial element of A. Then we shall write $A \mathbin{@} a$ for the subtype $\{x : A \mid \forall\_ : \Phi.x = a\}$, called the ***extension type*** after Riehl and Shulman [RS17].

Definition 3.22 (Vertical maps)**.** If A and B are types such that $\circ(A = B)$ holds, then we define a ***vertical map*** from A to B to be a function of the form $f : (A \to B) \mathbin{@} \lambda x.x$.

We refine the synthetic recollement of computability spaces (Observation 3.19) with a special type connective to build computability spaces from their Φ-modal and Φ-connected components.

Postulate 2 (Strict gluing [GSS22; SH22])**.** *On any of the ambient universes $\mathcal{U}$, we have a **strict gluing operation** that takes a Φ-modal type $A : \mathcal{U}_\circ$ and a family of Φ-connected types $B : A \to \mathcal{U}_\bullet$ to a type $\mathbb{G}_{x:A} Bx : \mathcal{U} \mathbin{@} A$ and an isomorphism $\mathsf{glue}_{A,B} : \sum_{x:A} Bx \cong \mathbb{G}_{x:A} Bx \mathbin{@} \pi_1$.*

Notation 3.23 (Gluing projections and constructor)**.** For $g : \mathbb{G}_{x:A} Bx$, the first projection $\pi_1\mathsf{glue}^{-1}_{A,B}g$ of $g : \mathbb{G}_{x:A} Bx$ can already be written $\wr g \int : A$. We shall write $\underline{g} : B\wr g\int$ for the second projection $\pi_2\mathsf{glue}^{-1}_{A,B}g$. Given $a : A$ and $b : Ba$ we shall write $a \lhd b$ for the element $\mathsf{glue}_{A,B}(a, b)$.

3.1.4 Internalizing the model of variables

The model of variables $\rho : \mathbf{R} \longrightarrow \mathbf{M}$ can be internalized into the synthetic Tait computability of $\mathcal{S}_{\mathsf{G}}$ by additional postulates.

Postulate 3 (The base model)**.** *There is a* Φ*-modal universe* $(\mathsf{Tp}, \mathsf{El})$ *such that for each code* $A : \mathsf{Tp}$ *the type* $\mathsf{El}\, A$ *is* Φ*-modal. Moreover,* Tp *is closed under function spaces as well as a base type* $\mathsf{O} : \mathsf{Tp}$ *and two constants* $\mathsf{yes}, \mathsf{no} : \mathsf{El}\,\mathsf{O}$*.*

Postulate 4 (The model of variables)**.** *There is an additional decoding family* Var *on* Tp *such that for each* $A : \mathsf{Tp}$*, we have* $\bigcirc(\mathsf{Var}\, A = \mathsf{El}\, A)$ *or equivalently* $\Phi \Rightarrow \mathsf{Var}\, A = \mathsf{El}\, A$*.*

3.1.5 The computability space of normal forms

With Postulate 3 and 4 in hand, it becomes possible to define a space of normal forms for types by means of an indexed inductive definition — or, for the more categorically inclined, as the initial algebra for a certain polynomial endofunctor on a slice of the ambient universe à la Fiore [Fio02]. In what follows, we will let $\mathcal{U}$ be a sufficiently large universe in the ambient type theory so as to classify each $\mathsf{El}\, A$ and $\mathsf{Var} A$.

Definition 3.24. A $\mathcal{U}$-small *normal form algebra* is defined to be a series of constants whose sorts we shall specify forthwith. First, a normal form algebra requires a sort NfTp of normal forms of types, and for each type $A : \mathsf{Tp}$ a pair of sorts $\mathsf{Nf}\, A, \mathsf{Ne}\, A$ classifying ***normal*** and ***neutral*** forms of elements of A.

$\mathsf{NfTp} : \mathcal{U} @ \mathsf{Tp}$
$\mathsf{Nf} : \prod_{A:\mathsf{Tp}} \mathcal{U} @ \mathsf{El}\, A$
$\mathsf{Ne} : \prod_{A:\mathsf{Tp}} \mathcal{U} @ \mathsf{El}\, A$

Next we require constructors for the normal forms of each type:

$\mathsf{nfO} : \mathsf{NfTp} @ \mathsf{O}$
$\mathsf{nfFun} : \prod_{A,B:\mathsf{NfTp}} \mathsf{NfTp} @ A \Rightarrow B$

Finally we require constructors for neutral and normal forms of terms.

$\mathsf{neVar} : \prod_{A:\mathsf{NfTp}} \prod_{x:\mathsf{Var}\wr A \wr} \mathsf{Ne}\, A @ x$
$\mathsf{neApp} : \prod_{A,B:\mathsf{NfTp}} \prod_{f:\mathsf{Ne}\wr A\Rightarrow B\wr} \prod_{x:\mathsf{Nf}\wr A\wr} \mathsf{Ne}\wr B \wr @ f x$

$\mathsf{nfNeO} : \prod_{x:\mathsf{Ne}\,\mathsf{O}} \mathsf{Nf}\,\mathsf{O} @ x$
$\mathsf{nfYes} : \mathsf{Nf}\,\mathsf{O} @ \mathsf{yes}$
$\mathsf{nfNo} : \mathsf{Nf}\,\mathsf{O} @ \mathsf{no}$
$\mathsf{nfLam} : \prod_{A,B:\mathsf{NfTp}} \prod_{f:\mathsf{Var}\, A\to\mathsf{Nf}\wr B\wr} \mathsf{Nf}\wr A \Rightarrow B \wr @ \lambda x. f x$

Definition 3.25. Let $\mathfrak{M}$ and $\mathfrak{N}$ be two $\mathcal{U}$-small normal form algebras. A ***morphism of normal form algebras*** from $H : \mathfrak{M} \longrightarrow \mathfrak{N}$ is given by functions between the three carriers

$H_{\mathsf{NfTp}} : \prod_{A:\mathsf{NfTp}_{\mathfrak{M}}} \mathsf{NfTp}_{\mathfrak{N}} @ A$

$H_{\mathsf{Nf}} : \prod_{A:\mathsf{Tp}} \prod_{x:\mathsf{Nf}_{\mathfrak{M}}\,A} \mathsf{Nf}_{\mathfrak{N}}\,A \mathbin{@} x$
$H_{\mathsf{Ne}} : \prod_{A:\mathsf{Tp}} \prod_{x:\mathsf{Ne}_{\mathfrak{M}}\,A} \mathsf{Ne}_{\mathfrak{N}}\,A \mathbin{@} x$

that preserve all the operations of the normal form algebra in the sense of the following representative equations:

$H_{\mathsf{NfTp}}\,\mathsf{nfO}_{\mathfrak{M}} = \mathsf{nfO}_{\mathfrak{N}}$
$H_{\mathsf{NfTp}}\,(\mathsf{nfFun}_{\mathfrak{M}}\,A\,B) = \mathsf{nfFun}_{\mathfrak{N}}\,(H_{\mathsf{NfTp}}A, H_{\mathsf{NfTp}}B)$
$H_{\mathsf{Ne}}\,\wr B \wr\,(\mathsf{neApp}_{\mathfrak{m}}\,A\,B\,f\,u) =$
$\quad \mathsf{neApp}_{\mathfrak{N}}\,(H_{\mathsf{NfTp}}A)\,(H_{\mathsf{NfTp}}B)\,(H_{\mathsf{Ne}}\,\wr A \Rightarrow B \wr\,f)\,(H_{\mathsf{Nf}}\,\wr A \wr\,u)$
$\ldots$

Lemma 3.26. *For* $X : \mathcal{U}_{\circ}$, *denote by* $\mathcal{U} \mathbin{@} X$ *the category whose morphisms are given by* vertical *maps. The functor* $\mathcal{U} \mathbin{@} X \longrightarrow \mathcal{U}_{\bullet}/X$ *sending each* $A : \mathcal{U} \mathbin{@} X$ *to the family* $(x : X) \mapsto A \mathbin{@} x$ *is an equivalence.*

Proof. This follows from Postulate 2. □

Lemma 3.27. *The exists an* ***initial normal form algebra****.*

Proof. Evidently, the initial normal form algebra would be the initial algebra for a certain endofunctor $\mathfrak{F}$ on the product category

$$\mathcal{U} \mathbin{@} \mathsf{Tp} \times \left(\textstyle\prod_{A:\mathsf{Tp}} \mathcal{U} \mathbin{@} \mathsf{El}\,A\right)^2$$

if such an initial algebra exists. By Lemma 3.26 and the disjointness property of sums we may equivalently present the category above as a slice of $\mathcal{U}_{\bullet}$:

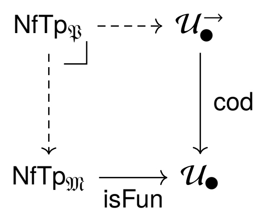
$$\begin{aligned}
&\mathcal{U} \mathbin{@} \mathsf{Tp}_{\mathbf{M}} \times \left(\textstyle\prod_{A:\mathsf{Tp}_{\mathbf{M}}} \mathcal{U} \mathbin{@} \mathsf{El}\,A\right)^2 \\
&\quad\simeq \mathcal{U}_{\bullet}/\mathsf{Tp} \times \left(\textstyle\prod_{A:\mathsf{Tp}} \mathcal{U}_{\bullet}/\mathsf{El}\,A\right)^2 && \text{(Lemma 3.26)} \\
&\quad\simeq \mathcal{U}_{\bullet}/\mathsf{Tp} \times \left(\mathcal{U}_{\bullet}/\left(\textstyle\sum_{A:\mathsf{Tp}} \mathsf{El}\,A\right)\right)^2 && \text{(disjointness)} \\
&\quad\simeq \mathcal{U}_{\bullet}/\left(\mathsf{Tp} + 2 \times \textstyle\sum_{A:\mathsf{Tp}} \mathsf{El}\,A\right) && \text{(disjointness)}
\end{aligned}$$

Under this identification, the endofunctor $\mathfrak{F}$ can be seen to be polynomial. Because $\mathcal{U}_{\bullet}$ has W-types and equality types, the initial algebra exists [GK13]. □

3.1.6 Injectivity of normal type constructors

Let $\mathfrak{M}$ be the initial normal form algebra.

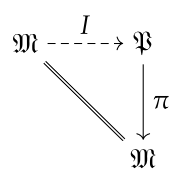
Construction 3.28. Let $\mathsf{isFun} : \mathsf{NfTp}_{\mathfrak{M}} \to \mathcal{U}_{\bullet}$ be the family sending each A to $\bullet\{(B, C) : \mathsf{NfTp}_{\mathfrak{M}}^2 \mid A = \mathsf{nfFun}_{\mathfrak{M}}\,B\,C\}$. We will define an auxiliary normal form algebra $\mathfrak{P}$ such that $\mathsf{NfTp}_{\mathfrak{P}}$ associates to each normal type $A : \mathsf{NfTp}_{\mathfrak{M}}$ a type A' equipped with a map into $\mathsf{isFun}\,A$. Of course, this description evokes the *Artin* *gluing* $\mathsf{NfTp}_{\mathfrak{M}} \downarrow \mathsf{isFun}$ when we view $\mathsf{NfTp}_{\mathfrak{M}}$ as a discrete category:



More explicitly, we define $\mathsf{NfTp}_{\mathfrak{P}}$ and the rest of the algebra as follows:

$\mathsf{NfTp}_{\mathfrak{P}} = \sum_{A:\mathsf{NfTp}_{\mathfrak{M}}} \sum_{A':\mathcal{U}_\bullet} (A' \to \mathsf{isFun}\, A)$
$\mathsf{Nf}_{\mathfrak{P}}\, (A, A', \_) = \mathsf{Nf}_{\mathfrak{M}}\, \wr A \int$
$\mathsf{nfO}_{\mathfrak{P}} = (\mathsf{nfO}_{\mathfrak{M}}, \Phi, \lambda\_.\star)$
$\mathsf{nfFun}_{\mathfrak{P}}\, A\, B = (\mathsf{nfFun}_{\mathfrak{M}}\, (\pi_1 A)\, (\pi_1 B), \top, \lambda\_.\eta_\bullet(\pi_1 A, \pi_1 B))$
...

We evidently have a homomorphism of algebras $\pi : \mathfrak{P} \longrightarrow \mathfrak{M}$ forgetting the additional information. As $\mathfrak{M}$ is initial, this projection homomorphism in fact has a (unique) section $I : \mathfrak{M} \longrightarrow \mathfrak{P}$:

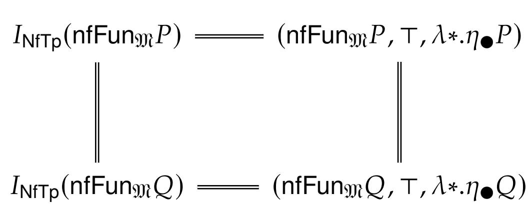


Lemma 3.29 (Modal injectivity of normal form constructors)**.** *The functorial map* $\bullet\mathsf{nfFun}_{\mathfrak{M}} : \bullet\mathsf{NfTp}^2_{\mathfrak{M}} \longrightarrow \bullet\mathsf{NfTp}_{\mathfrak{M}}$ *is a monomorphism.*

Proof. The claim is equivalent to the following formula:

$$\forall P, Q : \mathsf{NfTp}^2_{\mathfrak{M}}.\ \mathsf{nfFun}_{\mathfrak{M}} P = \mathsf{nfFun}_{\mathfrak{M}} Q \to \eta_\bullet P = \eta_\bullet Q$$

Fix P and Q such that $\mathsf{nfFun}_{\mathfrak{M}} F = \mathsf{nfFun}_{\mathfrak{M}} Q$. Considering the action of the universal map $I : \mathfrak{M} \longrightarrow \mathfrak{P}$ on this section, we have:



Thus by projection, we have $\eta_\bullet P = \eta_\bullet Q$. □

3.1.7 The universe of normalization spaces

Our goal has been to define a natural model of type theory lying over the bi-initial model **I** in which normal forms can be projected from the interpretations of types, following our discussion of Tait [Tai67] in Section 2.4.3. Tait's idea, which we will realize in a more technical form here, is to let the semantic universe of the normalization model assign a (vertical) projection map from each kind of semantic object into the corresponding space of normal forms. In order to close such a universe under function spaces, Tait noticed that it was necessary to have a vertical map *into* every semantic type from the space of neutral forms of elements of that type. In this section, we aim to define a universe of ***normalization spaces***, or computability spaces that are equipped with the structure described above.

Definition 3.30. A ***normalization space*** A is given by the following data:

1. a normal form $\Downarrow A : \mathsf{NfTp}$;
2. a type $\mathsf{El}^{\#}A : \mathcal{U} \mathbin{@} \mathsf{El} \Downarrow A$;
3. a "reflection" map $\Uparrow_A : \prod_{x:\mathsf{Ne}\,\wr\Downarrow A\wr} \mathsf{El}^{\#}A \mathbin{@} x$;
4. a "reification" map $\downarrow_A : \prod_{x:\mathsf{El}^{\#}A} \mathsf{Nf}\,\wr\Downarrow A\wr \mathbin{@} x$.

Of course, the reflection and reification maps can be stated as a sequence of *vertical* maps $\mathsf{Ne}\,\wr\Downarrow A\wr \to \mathsf{El}^{\#}A \to \mathsf{Nf}\,\wr\Downarrow A\wr$.

Construction 3.31 (The universe of normalization spaces)**.** By Postulate 2, we may define a type $\mathsf{Tp}^{\#}$ of normalization spaces such that $\bigcirc(\mathsf{Tp}^{\#} = \mathsf{Tp})$ strictly. Thus we have a universe $\mathcal{N} = (\mathsf{Tp}^{\#}, \mathsf{El}^{\#})$ that restricts under $\bigcirc$ to $(\mathsf{Tp}, \mathsf{El})$.

3.1.8 Closure of normalization spaces under connectives

We can close the universe of normalization spaces (Section 3.1.7) under the connectives that we have postulated on Tp in such a way that they restrict exactly to these under $\bigcirc$.

Construction 3.32 (The function space in normalization spaces)**.** For function spaces, we must define the following map (as well as corresponding maps for λ-abstraction and application):

$$(\Rightarrow^{\#}) : (\mathsf{Tp}^{\#} \times \mathsf{Tp}^{\#} \to \mathsf{Tp}^{\#}) \mathbin{@} (\Rightarrow)$$

Given two normalization spaces $A, B : \mathsf{Tp}^{\#}$ we must define a normalization space $(A \Rightarrow^{\#} B) : \mathsf{Tp}^{\#} \mathbin{@} A \Rightarrow B$. Below, we describe how to construct this space:

1. To define the normal form $\Downarrow(A \Rightarrow^{\#} B) : \mathsf{NfTp} \mathbin{@} A \Rightarrow B$, we use the normal forms of A and of B to construct $\mathsf{nfFun} \Downarrow A \Downarrow B$.
2. To define the type $\mathsf{El}^{\#}(A \Rightarrow^{\#} B)$ over $\mathsf{El}\,(A \Rightarrow B)$, we will use the function space $\mathsf{El}^{\#}A \to \mathsf{El}^{\#}B$. Note that this restricts only up to isomorphism to $\mathsf{El}\,(A \Rightarrow B)$, but that this can be corrected using Postulate 2. Therefore, we will not belabor the point further in our informal explanation.

3. To define the reflection map $\uparrow_{A\Rightarrow^{\#}B}$, we are given a neutral form $f : \mathsf{Ne}\,(A \Rightarrow B)$ and an element $x : \mathsf{El}^{\#}A$ and must produce an element $(\uparrow_{A\Rightarrow^{\#}B} f)x : \mathsf{El}^{\#}B \;@\; fx$. Applying the reflection map for B, it suffices to give a neutral form in $\mathsf{Ne}\,\wr B\wr \;@\; fx$; applying the neutral application constructor neApp, we need only a normal form in $\mathsf{Nf}\,\wr A\wr \;@\; x$, why we obtain by reification at A. All in all we have:

$$(\uparrow_{A\Rightarrow^{\#}B} f)x = \uparrow_B \mathsf{neApp}\, \Downarrow\! A \Downarrow\! B\, f\, (\downarrow_A x)$$

4. To define the reification map $\downarrow_{A\Rightarrow^{\#}B}$, we are given a function $f : \mathsf{El}^{\#}A \to \mathsf{El}^{\#}B$ and must exhibit a normal form $\downarrow_{A\Rightarrow^{\#}B} f : \mathsf{Nf}\,(A \Rightarrow B) \;@\; \lambda x.fx$. Applying the normal abstraction constructor nfLam, we are given a variable $x : \mathsf{Var}\,\wr A\wr$ and must construct a normal form in $\mathsf{Nf}\,\wr B\wr \;@\; fx$. Applying reification at B, it suffices to give an element of $\mathsf{El}^{\#}B \;@\; fx$; applying f itself, we need an element of $\mathsf{El}^{\#}A \;@\; x$ which we obtain from reflection at A and the neutral variable constructor neVar. To summarize:

$$\downarrow_{A\Rightarrow^{\#}B} f = \mathsf{neLam}\, \Downarrow\! A \Downarrow\! B\, (\lambda x. \downarrow_B f\, (\uparrow_A \mathsf{neVar} \Downarrow\! A\; x))$$

We leave the construction of λ-abstraction and application to the reader, as they are automatic by the fact that $\mathsf{El}^{\#}A \to \mathsf{El}^{\#}B$ is itself a function space.

Construction 3.33 (The base type in normalization spaces)**.** For the base type, we must construct the following three constants:

$$\begin{aligned} \mathsf{O}^{\#} &: \mathsf{Tp}^{\#} \;@\; \mathsf{O} \\ \mathsf{yes}^{\#} &: \mathsf{El}^{\#}\mathsf{O}^{\#} \;@\; \mathsf{yes} \\ \mathsf{no}^{\#} &: \mathsf{El}^{\#}\mathsf{O}^{\#} \;@\; \mathsf{no} \end{aligned}$$

1. We choose $\Downarrow\!\mathsf{O}^{\#}$ to be nfO.

2. We will let $\mathsf{El}^{\#}\mathsf{O}^{\#}$ be the type $\mathsf{Nf}\,\mathsf{O}$ of normal forms in the base type itself.

3. The reflection map is given by $\mathsf{nfNeO} : \prod_{x:\mathsf{Ne}\,\mathsf{O}} \mathsf{Nf}\,\mathsf{O} \;@\; x$.

4. The reification map given by the identity function.

Because we have chosen $\mathsf{El}^{\#}\mathsf{O}^{\#} = \mathsf{Nf}\,\mathsf{O}$, we may interpret $\mathsf{yes}^{\#}, \mathsf{no}^{\#}$ as $\mathsf{nfYes}, \mathsf{nfNo}$ respectively.

3.2 From normalization spaces to a natural model of type theory

The results of Section 3.1 culminated with a topos $\mathbf{G}$ of computability spaces $\mathbf{G}$ equipped with a universe $\mathcal{N}$ of ***normalization spaces***, closed under the connectives of our type theory in a way that restricts under the open immersion $j : \mathbf{E_M} \hookrightarrow \mathbf{G}$ to the corresponding constructs of the natural model $\mathbf{M}$. In this section, we aim to use those constructions as the basis for an actual natural model **N** over **M**; later we will instantiate these results with **M** taken to be the bi-initial model **I**. In particular, we shall apply the results of Section 2.1.5 to transform the universe $\mathcal{N}$ of normalization spaces into a genuine natural model $\mathbf{N} = \lceil \mathcal{N} \rceil_{\mathscr{G}}$ where $\mathscr{G} \subseteq \boldsymbol{\mathcal{S}}_{\mathsf{G}}$ is some suitable full subcategory of "test objects" containing all $\mathcal{N}$-contextual objects. In order to choose a suitable subcategory $\mathscr{G}$, we make an auxiliary definition.

Definition 3.34 (Atomic computability spaces)**.** An object of $\boldsymbol{\mathcal{S}}_{\mathsf{G}}$ is called an ***atomic computability space*** when it lies in the image of the embedding $(\!|-|\!) : \mathscr{C}_{\mathbf{R}} \hookrightarrow \boldsymbol{\mathcal{S}}_{\mathsf{G}}$ defined as the composite $\mathscr{C}_{\mathbf{R}} \xrightarrow{h_{\mathscr{C}_{\mathbf{R}}}} \mathbf{Pr}\,\mathscr{C}_{\mathbf{R}} \xrightarrow{i_!} \boldsymbol{\mathcal{S}}_{\mathsf{G}}$.

We then follow Uemura [Uem22] in choosing $\mathscr{G}$ be the smallest $\mathcal{N}$-contextual full subcategory of $\boldsymbol{\mathcal{S}}_{\mathsf{G}}$ containing all atomic computability spaces (so we may write $(\!|-|\!) : \mathscr{C}_{\mathbf{R}} \hookrightarrow \mathscr{G}$). We will write $I_{\mathscr{G}} : \mathscr{G} \hookrightarrow \boldsymbol{\mathcal{S}}_{\mathsf{G}}$ for the full subcategory inclusion, and $N_{\mathscr{G}} : \boldsymbol{\mathcal{S}}_{\mathsf{G}} \longrightarrow \mathbf{Pr}\,\mathscr{G}$ for the corresponding nerve functor that sends each computability space $X \in \boldsymbol{\mathcal{S}}_{\mathsf{G}}$ to its ***functor of $\mathscr{G}$-valued points***.

Definition 3.35. We define the ***normalization model*** **N** to be the externalization $\lceil \mathcal{N} \rceil_{\mathscr{G}}$ of $\mathcal{N}$ at the smallest $\mathcal{N}$-contextual full subcategory $\mathscr{G} \subseteq \boldsymbol{\mathcal{S}}_{\mathsf{G}}$ containing all atomic computability spaces.

Lemma 3.36. *We have a morphism of natural models $P : \mathbf{N} \longrightarrow \mathbf{M}$ preserving all type structure (function spaces and the base type).*

Proof. The underlying functor $P : \mathscr{G} \longrightarrow \mathscr{C}_{\mathbf{M}}$ can be defined to factor like so:

$$
\begin{array}{ccc}
\mathscr{G} & \xhookrightarrow{\;I_{\mathscr{G}}\;} & \boldsymbol{\mathcal{S}}_{\mathsf{G}} \\
{\scriptstyle P}\,\vdots & & \downarrow{\scriptstyle j^*} \\
\mathscr{C}_{\mathbf{M}} & \xhookrightarrow[\;h_{\mathscr{C}_{\mathbf{M}}}\;]{} & \mathbf{Pr}\,\mathscr{C}_{\mathbf{M}}
\end{array}
$$

That $j^* I_{\mathscr{G}}$ factors through the Yoneda embedding follows from Lemma 3.10 and the fact that the property of restricting along j to a representable is $\mathcal{N}$-contextual in the sense of Definition 1.6. We omit the rest of the construction of the morphism because it is routine and uninteresting. □

3.3 The normalization result

Now instantiate the constructions before by setting **M** to be the bi-initial **I** natural model closed under the specified connectives, and suppose that $\rho : \mathbf{R} \longrightarrow \mathbf{I}$ is the bi-initial model of variables over **I**. By the universal property of **I**, we have a section $S : \mathbf{I} \rightarrowtail \mathbf{N}$ to the projection $P : \mathbf{N} \twoheadrightarrow \mathbf{I}$ that we constructed in Section 3.2. The underlying functor of this section $S : \mathscr{C}_{\mathbf{I}} \longrightarrow \mathscr{G}$ sends each context from **I** to its glued interpretation; with this in hand, we make the following definition by analogy with Definition 3.34.

Definition 3.37 (Canonical computability spaces)**.** An object of $\boldsymbol{\mathcal{S}}_{\mathsf{G}}$ is called a ***canonical computability space*** when it lies in the image of the functor $\llbracket - \rrbracket : \mathscr{C}_{\mathbf{R}} \longrightarrow \mathscr{G} \subseteq \boldsymbol{\mathcal{S}}_{\mathsf{G}}$ defined as the composite $\mathscr{C}_{\mathbf{R}} \xrightarrow{\rho} \mathscr{C}_{\mathbf{I}} \xrightarrow{S} \mathscr{G}$.

We conclude with some observations that relate $\llparenthesis - \rrparenthesis$ and $\llbracket - \rrbracket$ to the internal language of $\boldsymbol{\mathcal{S}}_{\mathsf{G}}$.

Construction 3.38 (Internalizing types from the model of variables)**.** Morphisms $\boldsymbol{h}\Gamma \longrightarrow \mathsf{Tp}_{\mathbf{R}}$ in $\mathbf{Pr}\,\mathscr{C}_{\mathbf{R}}$ can be canonically identified with morphisms $\llparenthesis \Gamma \rrparenthesis \longrightarrow \mathsf{Tp}$ in $\boldsymbol{\mathcal{S}}_{\mathsf{G}}$ by means of the following composite natural isomorphism:

$$\begin{aligned}
&\mathbf{Hom}_{\mathbf{Pr}\,\mathscr{C}_{\mathbf{R}}}(\boldsymbol{h}_{\mathscr{C}_{\mathbf{R}}}\Gamma, \mathsf{Tp}_{\mathbf{R}}) \\
&\quad \cong \mathbf{Hom}_{\mathbf{Pr}\,\mathscr{C}_{\mathbf{R}}}(\boldsymbol{h}_{\mathscr{C}_{\mathbf{R}}}\Gamma, \boldsymbol{\rho}^*\mathsf{Tp}_{\mathbf{I}}) \\
&\quad \cong \mathbf{Hom}_{\mathbf{Pr}\,\mathscr{C}_{\mathbf{I}}}(\boldsymbol{h}_{\mathscr{C}_{\mathbf{I}}}\rho\Gamma, \mathsf{Tp}_{\mathbf{I}}) \\
&\quad \cong \mathbf{Hom}_{\boldsymbol{\mathcal{S}}_{\mathsf{G}}}(\boldsymbol{j}_*\boldsymbol{h}_{\mathscr{C}_{\mathbf{I}}}\rho\Gamma, \mathsf{Tp}) \\
&\quad \cong \mathbf{Hom}_{\boldsymbol{\mathcal{S}}_{\mathsf{G}}}(\bigcirc\llparenthesis \Gamma \rrparenthesis, \mathsf{Tp}) \\
&\quad \cong \mathbf{Hom}_{\boldsymbol{\mathcal{S}}_{\mathsf{G}}}(\llparenthesis \Gamma \rrparenthesis, \mathsf{Tp})
\end{aligned}$$

We shall write $\llparenthesis - \rrparenthesis$, $\llparenthesis - \rrparenthesis^{-1}$ for the forward and inverse directions of the natural isomorphism above.

Observation 3.39. *Let $A : \boldsymbol{h}\Gamma \longrightarrow \mathsf{Tp}_{\mathbf{R}}$ be a type in $\mathbf{R}$; then the atomic computability space $\llparenthesis \Gamma.A \rrparenthesis$ is canonically isomorphic to the dependent sum $\sum_{\gamma:\llparenthesis \Gamma \rrparenthesis} \mathsf{Var}(\llparenthesis A \rrparenthesis^{-1}\gamma)$.*

Lemma 3.40. *The projection map $\pi_1 : \sum_{A:\mathsf{Tp}_{\mathbf{I}}} \mathsf{Var}\,A \longrightarrow \mathsf{Tp}_{\mathbf{I}}$ is relatively representable by an atomic computability space.*

Proof. Let $\llparenthesis \Gamma \rrparenthesis$ an atomic computability space and let $B : \llparenthesis \Gamma \rrparenthesis \longrightarrow \mathsf{Tp}_{\mathbf{I}}$; we compute the fiber of π_1 as follows:

$$\begin{array}{ccc}
\sum_{\gamma:\llparenthesis \Gamma \rrparenthesis} \mathsf{Var}\,(B\gamma) & \xrightarrow{(\gamma, x) \mapsto (B\gamma, x)} & \sum_{A:\mathsf{Tp}_{\mathbf{I}}} \mathsf{Var}\,A \\
\big\downarrow \lrcorner & & \big\downarrow \pi_1 \\
\llparenthesis \Gamma \rrparenthesis & \xrightarrow[B]{} & \mathsf{Tp}_{\mathbf{I}}
\end{array}$$

By Observation 3.39, the pullback above is isomorphic to the projection $\llparenthesis p_{\llparenthesis B \rrparenthesis^{-1}} \rrparenthesis : \llparenthesis \Gamma.\llparenthesis B \rrparenthesis^{-1} \rrparenthesis \longrightarrow \llparenthesis \Gamma \rrparenthesis$. ◻

Observation 3.41. *Let $A : \boldsymbol{h}\Gamma \longrightarrow \mathsf{Tp}_{\mathbf{R}}$ be a type in $\mathbf{R}$; recalling that $\llbracket - \rrbracket = S \circ \rho$ tracks a morphism of natural models, we have a type $(S \circ \rho)_{\mathsf{Tp}} \cdot A : \boldsymbol{h}\llbracket \Gamma \rrbracket \longrightarrow \mathsf{Tp}_{\ulcorner \mathcal{N} \urcorner_{\mathscr{G}}}$ in $\mathbf{N}$, which can equally well be described as a map $\llbracket A \rrbracket : \llbracket \Gamma \rrbracket \longrightarrow \mathsf{Tp}^{\#}$ in $\boldsymbol{\mathcal{S}}_{\mathsf{G}}$. From this perspective, the canonical computability space $\llbracket \Gamma.A \rrbracket$ is the dependent sum $\sum_{\gamma:\llbracket \Gamma \rrbracket} \mathsf{El}^{\#}\llbracket A \rrbracket\gamma$.*

3.3.1 The functors of atomic and canonical points

Given a space $X \in \mathcal{S}_{\mathsf{G}}$, a ***atomic point*** of X is defined to be a generalized element of X defined on an atomic computability space $(\!|\Gamma|\!)$; likewise, a ***canonical point*** of X is defined to be a generalized element of X defined on a canonical computability space $[\![\Gamma]\!]$. Thus the ***functors of (atomic, canonical) points*** of X are the presheaves $\mathbf{Hom}_{\mathcal{S}_{\mathsf{G}}}((\!|-|\!), X)$, $\mathbf{Hom}_{\mathcal{S}_{\mathsf{G}}}([\![-]\!], X)$ respectively in $\mathbf{Pr}\,\mathscr{C}_{\mathbf{R}}$.

Definition 3.42 (Restricting to a functor of points)**.** Let $F : \mathscr{C}_{\mathbf{R}} \longrightarrow \mathcal{S}_{\mathsf{G}}$ be a functor such that $\boldsymbol{j}^* \circ F \cong \boldsymbol{h}_{\mathscr{C}_{\mathbf{I}}} \circ \rho$; for any $X \in \mathcal{S}_{\mathsf{G}}$, the ***functor of*** F***-valued points*** of X is defined to be the presheaf $\mathbf{Hom}_{\mathcal{S}_{\mathsf{G}}}(F-, X)$ in $\mathbf{Pr}\,\mathscr{C}_{\mathbf{R}}$. We define the ***restriction of*** X ***to its functor of*** F***-valued points*** to be the space $X_F \in \mathcal{S}_{\mathsf{G}}$ determined by the following natural transformation $\mathbf{Hom}_{\mathcal{S}_{\mathsf{G}}}(F-, X) \longrightarrow \rho^* \boldsymbol{j}^* X$:

$$\mathbf{Hom}_{\mathcal{S}_{\mathsf{G}}}(F-, X) \to \mathbf{Hom}_{\mathbf{Pr}\,\mathscr{C}_{\mathbf{I}}}(\boldsymbol{j}^* F-, \boldsymbol{j}^* X) \xrightarrow{\cong} \mathbf{Hom}_{\mathbf{Pr}\,\mathscr{C}_{\mathbf{I}}}(\boldsymbol{h}\rho-, \boldsymbol{j}^* X) \to \rho^* \boldsymbol{j}^* X$$

Given a natural transformation $\alpha : F \longrightarrow G$ between two such functors, the precomposition map $\mathbf{Hom}_{\mathcal{S}_{\mathsf{G}}}(\alpha-, X)$ induces a vertical reindexing map $X_\alpha : X_G \longrightarrow X_F$.

Lemma 3.43. *For any space $X \in \mathcal{S}_{\mathsf{G}}$, the functor of canonical points $\mathbf{Hom}_{\mathcal{S}_{\mathsf{G}}}((\!|-|\!), X)$ is canonically isomorphic to the restriction $\boldsymbol{i}^* X$ of X along the closed immersion $\boldsymbol{i} : \mathsf{E}_{\mathbf{R}} \hookrightarrow \mathsf{G}$.*

Proof. This follows by adjointness and the definition $(\!|-|\!) = \boldsymbol{i}_! \circ \boldsymbol{h}$. ◻

Corollary 3.44. *The restriction $X_{(\!|-|\!)}$ of any space $X \in \mathcal{S}_{\mathsf{G}}$ to its functor of atomic points is canonically isomorphic to X itself.*

Construction 3.45 (Internalizing the action of S on types)**.** The map $S : \mathbf{I} \rightarrowtail \mathbf{N}$ determined by the universal property of the bi-initial model carries an action $S_{\mathsf{Tp}} \cdot -$ that transforms a type $A : \boldsymbol{h}\Gamma \longrightarrow \mathsf{Tp}_{\mathbf{I}}$ in the bi-initial model to a type $S_{\mathsf{Tp}} \cdot A : \boldsymbol{h}S\Gamma \longrightarrow \mathsf{Tp}_{\mathbf{N}}$ in the normalization model. This map internalizes directly into $\mathcal{S}_{\mathsf{G}}$ as a *vertical map* from $\mathsf{Tp} \longrightarrow \mathsf{Tp}^{\#}_{[\![-]\!]}$ from Tp to the restriction $\mathsf{Tp}^{\#}_{[\![-]\!]}$ of $\mathsf{Tp}^{\#}$ to its functor of canonical points. To define a vertical map $\mathsf{Tp} \longrightarrow \mathsf{Tp}^{\#}_{[\![-]\!]}$ is the same as to define a section of the projection map $\mathbf{Hom}_{\mathcal{S}_{\mathsf{G}}}([\![-]\!], \mathsf{Tp}^{\#}) \longrightarrow \rho^* \mathsf{Tp}_{\mathbf{I}}$:

$$\rho^* \mathsf{Tp}_{\mathbf{I}} \xrightarrow{\cong} \mathbf{Hom}_{\mathbf{Pr}\,\mathscr{C}_{\mathbf{I}}}(\boldsymbol{h}\rho-, \mathsf{Tp}_{\mathbf{I}}) \xrightarrow{S_{\mathsf{Tp}} \cdot -} \mathbf{Hom}_{\mathbf{Pr}\,\mathscr{G}}(\boldsymbol{h}[\![-]\!], N_{\mathscr{G}} \mathsf{Tp}^{\#}) \xrightarrow{\cong} \mathbf{Hom}_{\mathcal{S}_{\mathsf{G}}}([\![-]\!], \mathsf{Tp}^{\#})$$

Remark 3.46 (Toward an internal evaluation map)**.** The vertical map $\mathsf{Tp} \longrightarrow \mathsf{Tp}^{\#}_{[\![-]\!]}$ internalizing the action of S on types from Construction 3.45 is a good first step, what we *need* for our results is an unrestricted vertical map $\mathsf{Tp} \longrightarrow \mathsf{Tp}^{\#}$. We will do so by exhibiting for *any* X a canonical vertical map $X_{[\![-]\!]} \longrightarrow X$; recalling that $X \cong X_{(\!|-|\!)}$, it evidently suffices to define a (suitably vertical) natural transformation $(\!|-|\!) \longrightarrow [\![-]\!]$ from the functors of atomic points to the functors of canonical points, which we shall refer in Section 3.3.2 as ***hydration***.

3.3.2 Hydration of variables via Bocquet, Kaposi, and Sattler's *inserter*

The goal of this section is to define a suitably vertical natural transformation $(\!|-|\!) \longrightarrow [\![-]\!]$ that "hydrates" an element of an atomic computability space into an element of the corresponding canonical computability. Reindexing along this natural transformation, we would then obtain a map $X_{[\![-]\!]} \longrightarrow X_{(\!|-|\!)} \cong X$ that we could use to define an internal evaluation map $\mathsf{Tp} \longrightarrow \mathsf{Tp}^{\#}$ as in Remark 3.46.

We shall view both $\mathscr{C}_{\mathbf{R}}$ and $\mathscr{G}$ as categories *displayed* over $\mathbf{Pr}\,\mathscr{C}_{\mathbf{I}}$ via the functors $\boldsymbol{h}_{\mathscr{C}_{\mathbf{I}}} \circ \rho : \mathscr{C}_{\mathbf{R}} \longrightarrow \mathbf{Pr}\,\mathscr{C}_{\mathbf{I}}$ and $\boldsymbol{j}^{*} : \mathscr{G} \longrightarrow \mathbf{Pr}\,\mathscr{C}_{\mathbf{I}}$. We observe that both $(\!|-|\!)$ and $[\![-]\!]$ lift into the slice $\mathbf{Cat}_{/\mathbf{Pr}\,\mathscr{C}_{\mathbf{I}}}$, as witnessed by the following diagram:

$$
\begin{array}{ccccc}
\mathscr{G} & \xleftarrow{\;(\!|-|\!)\;} & \mathscr{C}_{\mathbf{R}} & \xrightarrow{\;[\![-]\!]\;} & \mathscr{G} \\
& \searrow^{\boldsymbol{j}^{*}} & \downarrow{\scriptstyle \boldsymbol{h}_{\mathscr{C}_{\mathbf{I}}} \circ \rho} & \swarrow_{\boldsymbol{j}^{*}} & \\
& & \mathbf{Pr}\,\mathscr{C}_{\mathbf{I}} & &
\end{array}
$$

Stated now with more precision, our goal is then define a 2-cell $\nearrow$: $(\!|-|\!) \longrightarrow [\![-]\!]$ in the slice $\mathbf{Cat}_{/\mathbf{Pr}\,\mathscr{C}_{\mathbf{I}}}$ that "hydrates" an element of an atomic computability space to an element of the corresponding canonical computability space. Our construction follows that of Uemura [Uem22, Appendix A], which is itself modeled on the original more cryptic formulation by Bocquet, Kaposi, and Sattler [BKS21]. In particular, we shall define a model of variables $\mathbf{H}$ over $\mathbf{R}$ from which we can extract the desired hydration map. This is essentially an inductive argument that will be carried out using the universal property of $\mathbf{R}$ as the bi-initial model of variables over $\mathbf{I}$.

Construction 3.47 (The hydration model)**.** We choose $\mathscr{C}_{\mathbf{H}}$ to be the ***inserter object*** determined by the morphisms $(\!|-|\!), [\![-]\!]$ in $\mathbf{Cat}_{/\mathbf{Pr}\,\mathscr{C}_{\mathbf{I}}}$. An object of the inserter $\mathscr{C}_{\mathbf{H}}$ is a pair of an object $\Gamma \in \mathscr{C}_{\mathbf{R}}$ and a *vertical* map $\eta_{\Gamma} : (\!|\Gamma|\!) \longrightarrow [\![\Gamma]\!]$; a morphism from (Δ, η_{Δ}) to (Γ, η_{Γ}) is given by a morphism $\gamma : \Delta \longrightarrow \Gamma$ such that the following square commutes:

$$
\begin{array}{ccc}
(\!|\Delta|\!) & \xrightarrow{\;(\!|\gamma|\!)\;} & (\!|\Gamma|\!) \\
{\scriptstyle \eta_{\Gamma}}\downarrow & & \downarrow{\scriptstyle \eta_{\Delta}} \\
[\![\Delta]\!] & \xrightarrow[\;[\![\gamma]\!]\;]{} & [\![\Gamma]\!]
\end{array}
$$

There is an evident projection functor $H : \mathscr{C}_{\mathbf{H}} \longrightarrow \mathscr{C}_{\mathbf{R}}$ sending each (Γ, η_{Γ}) to Γ. We define $\mathsf{Tp}_{\mathbf{H}} \in \mathbf{Pr}\,\mathscr{C}_{\mathbf{H}}$ to be the presheaf $H^{*}\mathsf{Tp}_{\mathbf{R}}$; in order to define $\mathsf{El}_{\mathbf{H}} \in \mathbf{Pr}\,\mathscr{C}_{\mathbf{H}}/\mathsf{Tp}_{\mathbf{H}}$, we first describe the comprehension of a given element

$A \in \mathsf{Tp}_{\mathbf{H}}(\Gamma, \eta_\Gamma)$ as an object $(\Gamma.A, \eta_{\Gamma.A}) \in \mathscr{C}_{\mathbf{H}}/(\Gamma, \eta_\Gamma)$. In particular, let $\Gamma.A$ be the corresponding comprehension in $\mathbf{R}$ as below:

$$\begin{array}{ccc} \boldsymbol{h}(\Gamma.A) & \longrightarrow & \mathsf{El}_{\mathbf{R}} \\ {\scriptstyle \boldsymbol{h}p_A}\big\downarrow \lrcorner & & \big\downarrow{\scriptstyle \pi_{\mathbf{R}}} \\ \boldsymbol{h}\Gamma & \underset{A}{\longrightarrow} & \mathsf{Tp}_{\mathbf{R}} \end{array}$$

We will define a vertical map $\eta_{\Gamma.A} : \llparenthesis \Gamma.A \rrparenthesis \longrightarrow \llbracket \Gamma.A \rrbracket$ fitting into the following commuting square:

$$\begin{array}{ccc} \llparenthesis \Gamma.A \rrparenthesis & \overset{\eta_{\Gamma.A}}{\dashrightarrow} & \llbracket \Gamma.A \rrbracket \\ {\scriptstyle \llparenthesis p_A \rrparenthesis}\big\downarrow & & \big\downarrow{\scriptstyle \llbracket p_A \rrbracket} \\ \llparenthesis \Gamma \rrparenthesis & \underset{\eta_\Gamma}{\longrightarrow} & \llbracket \Gamma \rrbracket \end{array}$$

Using Observations 3.39 and 3.41, we see that such a map can be defined using the following *internal* variable hydration map defined using the reflection map of any normalization space:

$$\begin{array}{l} \mathsf{hydrate} : \prod_{A:\mathsf{Tp}^\sharp} \mathsf{Var}\,\wr A \wr \Rrightarrow \mathsf{El}^\sharp A \\ \mathsf{hydrate}\, A\, x = \Uparrow_A \mathsf{neVar}\,(\Downarrow A)\, x \end{array}$$

The projection functor $H : \mathscr{C}_{\mathbf{H}} \longrightarrow \mathscr{C}_{\mathbf{R}}$ can now be seen to track a morphism of natural models $H : \mathbf{H} \longrightarrow \mathbf{R}$; moreover, this morphism exhibits $\mathbf{H}$ by definition as a *model of variables* over $\mathbf{R}$.

Construction 3.48 (The hydration map)**.** As $H : \mathbf{H} \longrightarrow \mathbf{R}$ is a model of variables over $\mathbf{R}$, the composite $\rho \circ H : \mathbf{H} \longrightarrow \mathbf{I}$ is also a model of variables over $\mathbf{I}$. As $\mathbf{R}$ is assumed to be the bi-initial model of variables, we have an essentially unique section $\mathbf{R} \longrightarrow \mathbf{H}$ over $\mathbf{I}$. The underlying functor of this section sends each context $\Gamma \in \mathscr{C}_{\mathbf{R}}$ to a morphism $\eta_\Gamma : \llparenthesis \Gamma \rrparenthesis \longrightarrow \llbracket \Gamma \rrbracket$, and functoriality guarantees that this assignment is natural. Therefore, we may define $\nearrow : \llparenthesis - \rrparenthesis \longrightarrow \llbracket - \rrbracket$ componentwise by $\nearrow_\Gamma = \eta_\Gamma$.

3.3.3 The normalization map and its injectivity

By reindexing along our vertical hydration map $\nearrow : \llparenthesis - \rrparenthesis \longrightarrow \llbracket - \rrbracket$, we therefore obtain a vertical map $X_\nearrow : X_{\llbracket - \rrbracket} \longrightarrow X_{\llparenthesis - \rrparenthesis} \cong X$. As we see below, this is enough to fulfill the problem posed by Remark 3.46.

Construction 3.49 (The internal evaluation map)**.** We shall now exhibit a vertical evaluation map $[\![-]\!]_{\mathsf{Tp}} : \mathsf{Tp} \longrightarrow \mathsf{Tp}^{\#}$ within $\mathcal{S}_{\mathbf{G}}$ sending each type **I**-type to the normalization space chosen by our model.

$$\mathsf{Tp} \xrightarrow{\text{Construction 3.45}} \mathsf{Tp}^{\#}_{[\![-]\!]} \xrightarrow{\mathsf{Tp}^{\#}_{\nearrow}} \mathsf{Tp}^{\#}$$

Construction 3.50 (The internal normalization map)**.** We may compose the internal evaluation map $[\![-]\!]_{\mathsf{Tp}} : \mathsf{Tp} \longrightarrow \mathsf{Tp}^{\#}$ with the vertical projection $\Downarrow - : \mathsf{Tp}^{\#} \longrightarrow \mathsf{NfTp}$ of normal forms from normalization spaces to obtain a vertical ***normalization map*** $\mathsf{norm}_{\mathsf{Tp}} : \mathsf{Tp} \longrightarrow \mathsf{NfTp}$ that takes any element of Tp to its normal form.

Observation 3.51. *The internal normalization map* $\mathsf{norm}_{\mathsf{Tp}} : \mathsf{Tp} \longrightarrow \mathsf{NfTp}$ *is a monomorphism, as it is a section of the unit map* $\mathsf{NfTp} \longrightarrow \bigcirc\mathsf{NfTp} \cong \mathsf{Tp}$*.*

3.4 Injectivity of type constructors

In Question 1, we have asked whether $\Rightarrow_{\mathbf{I}}: \mathsf{Tp}_{\mathbf{I}} \times \mathsf{Tp}_{\mathbf{I}} \longrightarrow \mathsf{Tp}_{\mathbf{I}}$ is a monomorphism in $\mathbf{Pr}\,\mathscr{C}_{\mathbf{I}}$. We can now answer in the affirmative, by virtue of the normalization result (Section 3.3).

Lemma 3.52. *The morphism of topoi* $\boldsymbol{\rho} : \mathbf{E}_{\mathbf{R}} \longrightarrow \mathbf{E}_{\mathbf{I}}$ *is a geometric surjection.*

Proof. This follows from Lemma 3.3, since the bi-initial model is always democratic [Uem21]. □

Theorem 3.53 (Injectivity of type constructors)**.** *The function space constructor* $(\Rightarrow_{\mathbf{I}}) : \mathsf{Tp}_{\mathbf{I}} \times \mathsf{Tp}_{\mathbf{I}} \longrightarrow \mathsf{Tp}_{\mathbf{I}}$ *is a monomorphism in* $\mathbf{Pr}\,\mathscr{C}_{\mathbf{I}}$*.*

Proof. As $\boldsymbol{\rho} : \mathbf{E}_{\mathbf{R}} \longrightarrow \mathbf{E}_{\mathbf{I}}$ is a surjection (Lemma 3.52), its inverse image functor is faithful; as $\boldsymbol{i} : \mathbf{E}_{\mathbf{R}} \hookrightarrow \mathbf{G}$ is an embedding, its direct image is (fully) faithful. As faithful functors reflect monomorphisms, it suffices for us to show that $\boldsymbol{i}_*\boldsymbol{\rho}^*(\Rightarrow_{\mathbf{I}})$ is a monomorphism in $\mathbf{Pr}\,\mathscr{C}_{\mathbf{R}}$. Since $\boldsymbol{\rho}^* \cong \boldsymbol{i}^*\boldsymbol{j}_*$ we have $\boldsymbol{i}_*\boldsymbol{\rho}^*(\Rightarrow_{\mathbf{I}}) \cong \boldsymbol{i}_*\boldsymbol{i}^*\boldsymbol{j}_*(\Rightarrow_{\mathbf{I}}) = \bullet\boldsymbol{j}_*(\Rightarrow_{\mathbf{I}})$. Hence it is enough to show that $\bullet\boldsymbol{j}_*(\Rightarrow_{\mathbf{I}})$ is a monomorphism.

Switching to the internal language, we suppress the embedding $\boldsymbol{j}_*$ and aim to check that the function $\bullet(\Rightarrow) : \bullet(\mathsf{Tp} \times \mathsf{Tp}) \to \bullet\mathsf{Tp}$ is injective. Fixing $A, A', B, B' : \mathsf{Tp}$ such that $(A \Rightarrow B) = (A' \Rightarrow B')$, our goal is to check that $\bullet((A, B) = (A', B'))$. We know that $\mathsf{norm}_{\mathsf{Tp}}(A \Rightarrow B) = \mathsf{norm}_{\mathsf{Tp}}(A' \Rightarrow B')$; unfolding the definition of $\mathsf{norm}_{\mathsf{Tp}}$ induced by the normalization model in Constructions 3.49 and 3.50 we conclude that $\mathsf{nfFun}(\mathsf{norm}_{\mathsf{Tp}}A, \mathsf{norm}_{\mathsf{Tp}}B) = \mathsf{nfFun}(\mathsf{norm}_{\mathsf{Tp}}A', \mathsf{norm}_{\mathsf{Tp}}B')$. Our goal then follows from the modal injectivity of normal form constructors (Lemma 3.29) together with our Observation 3.51 that the normalization function is injective. □

4 Concluding remarks

We have at long last shown in Theorem 3.53 how to prove that the type constructor for function spaces is a monomorphism in the bi-initial model of type theory with function spaces on a base type with two constants. A few things deserve additional comment.

Extension to more sophisticated results We have focused on the injectivity of ordinary function spaces for the sake of simplicity, but the methods exposed herein also apply to dependent product, dependent sums, *etc*. Likewise, our methods extend readily to prove more difficult results, including the fact that the normalization function is not only a section but in fact an isomorphism. From these results, one may deduce a solution to the word problem for Martin-Löf type theory. Finally, these methods can be adapted to apply to much more sophisticated type theories, including cubical type theory [Ste21; SA21], multi-modal type theory [Gra22], and even "∞-type theories" [Uem22].

Emphasis of universal properties over explicit constructions At every stage in our development, we have worked as much as possible with invariant universal properties rather than explicit constructions. For instance, we worked with the (2,1)-categorical universal property of bi-initial natural model not because we do not think that the concrete syntax of type theory is important, but because we want our proofs to be flexible enough to apply to *any* correct implementation of this concrete syntax, *i.e*. any presentation that can be shown to satisfy the universal property. The concrete presentation of type theoretic syntax is both highly non-trivial and deeply obscure: for this reason, it cannot be counted as a virtue for a proof to be applicable only to a specific obscure presentation that is likely to be superseded as the winds of fashion blow one way or another.

Likewise, it is possible to give an explicit construction of the "model of variables" in terms of syntactically defined telescopes (see Sterling [Ste21, §5.5] for such a construction), but we have followed the more modular proof technique of Bocquet, Kaposi, and Sattler [BKS21] not because we wish to worship abstraction for abstraction's sake, but because the proof applies to *any* presentation of the bi-initial model of variables. The flexibility to choose different presentations is very important for implementation because such choices can have a significant impact on efficiency; therefore, a modern proof that is invariant in this way is arguably much closer to practical applications than the more old-fashioned ones that emphasized explicit constructions.

References

[AHS95] Thorsten Altenkirch, Martin Hofmann, and Thomas Streicher. "Categorical reconstruction of a reduction free normalization proof". In: *Category Theory and Computer Science*. Ed. by David Pitt, David E. Rydeheard, and Peter Johnstone. Berlin, Heidelberg: Springer Berlin Heidelberg, 1995, pp. 182–199. ISBN: 978-3-540-44661-3. DOI: 10.1007/3-540-60164-3_27.

[AJ21] Mathieu Anel and André Joyal. "Topo-logie". In: *New Spaces in Mathematics: Formal and Conceptual Reflections*. Ed. by Mathieu Anel and Gabriel Catren. Vol. 1. Cambridge University Press, 2021. Chap. 4, pp. 155–257. DOI: 10.1017/9781108854429.007.

[Art11] Michael Artin. *Algebra*. 2nd. Pearson Education, 2011.

[AGV72] Michael Artin, Alexander Grothendieck, and Jean-Louis Verdier. *Théorie des topos et cohomologie étale des schémas*. Vol. 269, 270, 305. Lecture Notes in Mathematics. Séminaire de Géométrie Algébrique du Bois-Marie 1963–1964 (SGA 4), Dirigé par M. Artin, A. Grothendieck, et J.-L. Verdier. Avec la collaboration de N. Bourbaki, P. Deligne et B. Saint-Donat. Berlin: Springer-Verlag, 1972.

[Awo18] Steve Awodey. "Natural models of homotopy type theory". In: *Mathematical Structures in Computer Science* 28.2 (2018), pp. 241–286. DOI: 10.1017/S0960129516000268. arXiv: 1406.3219 [math.CT].

[Awo22] Steve Awodey. *On Hofmann-Streicher universes*. 2022. DOI: 10.48550/ARXIV.2205.10917.

[AW09] Steve Awodey and Michael A. Warren. "Homotopy theoretic models of identity types". In: *Mathematical Proceedings of the Cambridge Philosophical Society* 146.1 (Jan. 2009), pp. 45–55. ISSN: 0305-0041. DOI: 10.1017/S0305004108001783.

[Bén73] Jean Bénabou. *Problèmes dans les topos : d'après le cours de Questions spéciales de mathématique*. Séminaires de mathématique pure : Rapport, no 34. 34. Louvain-la-Neuve : Institut de mathématique pure et appliquée, Université catholique de Louvain, 1973.

[Ble17] Ingo Blechschmidt. "Using the internal language of toposes in algebraic geometry". PhD thesis. Universität Augsberg, 2017.

[BKS21] Rafaël Bocquet, Ambrus Kaposi, and Christian Sattler. *Relative induction principles for type theories*. 2021. DOI: 10.48550/ARXIV.2102.11649.

[BF06] Marta Bunge and Jonathon Funk. *Singular coverings of toposes*. Vol. 1890. Lecture Notes in Mathematics. Berlin: Springer, 2006. ISBN: 3-540-36359-9.

[Car78] John Cartmell. "Generalised Algebraic Theories and Contextual Categories". PhD thesis. Oxford University, Jan. 1978.

[CCD17] Simon Castellan, Pierre Clairambault, and Peter Dybjer. "Undecidability of Equality in the Free Locally Cartesian Closed Category (Extended version)". In: *Logical Methods in Computer Science* 13.4 (2017).

[Coq16] The Coq Development Team. *The Coq Proof Assistant Reference Manual*. 2016.

[Coq19] Thierry Coquand. "Canonicity and normalization for dependent type theory". In: *Theoretical Computer Science* 777 (2019). In memory of Maurice Nivat, a founding father of Theoretical Computer Science - Part I, pp. 184–191. ISSN: 0304-3975. DOI: 10.1016/j.tcs.2019.01.015. arXiv: 1810.09367 [cs.PL].

[Dyb96] Peter Dybjer. "Internal type theory". In: *Types for Proofs and Programs: International Workshop, TYPES '95 Torino, Italy, June 5–8, 1995 Selected Papers*. Ed. by Stefano Berardi and Mario Coppo. Berlin, Heidelberg: Springer Berlin Heidelberg, 1996, pp. 120–134. ISBN: 978-3-540-70722-6.

[Fio02] Marcelo Fiore. "Semantic Analysis of Normalisation by Evaluation for Typed Lambda Calculus". In: *Proceedings of the 4th ACM SIGPLAN International Conference on Principles and Practice of Declarative Programming*. PPDP '02. Pittsburgh, PA, USA: Association for Computing Machinery, 2002, pp. 26–37. ISBN: 1-58113-528-9. DOI: 10.1145/571157.571161.

[Fio22] Marcelo Fiore. *Semantic Analysis of Normalisation by Evaluation for Typed Lambda Calculus*. 2022. arXiv: 2207.08777 [cs.LO].

[FS99] Marcelo Fiore and Alex Simpson. "Lambda Definability with Sums via Grothendieck Logical Relations". In: *Typed Lambda Calculi and Applications*. Ed. by Jean-Yves Girard. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, pp. 147–161. ISBN: 978-3-540-48959-7.

[Fre78] Peter Freyd. "On proving that **1** is an indecomposable projective in various free categories". Unpublished manuscript. 1978.

[GK13] Nicola Gambino and Joachim Kock. "Polynomial functors and polynomial monads". In: *Mathematical Proceedings of the Cambridge Philosophical Society* 154.1 (2013), pp. 153–192. DOI: 10.1017/S0305004112000394.

[Gir71] Jean-Yves Girard. "Une extension de l'interprétation de Gödel à l'analyse, et son application à l'élimination de coupures dans l'analyse et la théorie des types". In: *Proceedings of the Second Scandinavian Logic Symposium*. 1971.

[Gon08] Georges Gonthier. "Formal Proof — The Four-Color Theorem". In: *Notices of the AMS* 55.11 (2008). URL: https://www.ams.org/notices/200811/tx081101382p.pdf.

[Gon+13] Georges Gonthier et al. "A Machine-Checked Proof of the Odd Order Theorem". In: *ITP 2013, 4th Conference on Interactive Theorem Proving*. Ed. by Sandrine Blazy, Christine Paulin, and David Pichardie. Vol. 7998. Lecture Notes in Computer Science. Rennes, France: Springer, July 2013, pp. 163–179. DOI: 10.1007/978-3-642-39634-2_14. URL: https://hal.inria.fr/hal-00816699.

[Gra22] Daniel Gratzer. "Normalization for Multimodal Type Theory". In: *Proceedings of the 37th Annual ACM/IEEE Symposium on Logic in Computer Science*. New York, NY, USA: Association for Computing Machinery, 2022. DOI: 10.1145/3531130.3532398.

[GSS22] Daniel Gratzer, Michael Shulman, and Jonathan Sterling. "Strict universes for Grothendieck topoi". Unpublished manuscript. Feb. 2022. DOI: 10.48550/arXiv.2202.12012. arXiv: 2202.12012 [math.CT].

[Gro86] Alexander Grothendieck. "Récoltes et Semailles, Réflexions et témoignages sur un passé de mathématicien". Autobiographical memoir circulated in the 1980s. 1986.

[GD60] Alexander Grothendieck and Jean Dieudonné. "Éléments de géométrie algébrique : I. Le langage des schémas". In: *Publications Mathématiques de l'IHÉS* 4 (1960), pp. 5–228. URL: http://www.numdam.org/item/PMIHES_1960__4__5_0.

[Hof97] Martin Hofmann. "Syntax and Semantics of Dependent Types". In: *Semantics and Logics of Computation*. Ed. by Andrew M. Pitts and Peter Dybjer. Cambridge University Press, 1997, pp. 79–130.

[HS98] Martin Hofmann and Thomas Streicher. "The groupoid interpretation of type theory". In: *Twenty-five years of constructive type theory (Venice, 1995)*. Vol. 36. Oxford Logic Guides. New York: Oxford Univ. Press, 1998, pp. 83–111. DOI: 10.1093/oso/9780198501275.001.0001.

[Hou+16] Kuen-Bang Hou (Favonia), Eric Finster, Daniel R. Licata, and Peter LeFanu Lumsdaine. "A Mechanization of the Blakers-Massey Connectivity Theorem in Homotopy Type Theory". In: *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science*. New York, NY, USA: Association for Computing Machinery, 2016, pp. 565–574. ISBN: 978-1-4503-4391-6. DOI: 10.1145/2933575.2934545.

[Joh77] Peter T. Johnstone. *Topos Theory*. Academic Press, 1977.

[Joh02] Peter T. Johnstone. *Sketches of an Elephant: A Topos Theory Compendium: Volumes 1 and 2*. Oxford Logical Guides 43. Oxford Science Publications, 2002.

[JT93] Achim Jung and Jerzy Tiuryn. "A new characterization of lambda definability". In: *Typed Lambda Calculi and Applications*. Ed. by Marc Bezem and Jan Friso Groote. Berlin, Heidelberg: Springer Berlin Heidelberg, 1993, pp. 245–257. ISBN: 978-3-540-47586-6.

[LW15] Peter LeFanu Lumsdaine and Michael A. Warren. "The Local Universes Model: An Overlooked Coherence Construction for Dependent Type Theories". In: *ACM Transactions on Computational Logic* 16.3 (July 2015), 23:1–23:31. ISSN: 1529-3785. DOI: 10.1145/2754931.

[Lur09] Jacob Lurie. *Higher Topos Theory*. Princeton University Press, 2009. ISBN: 978-0-691-14049-0. arXiv: math/0608040 [math.CT].

[MT20] Assia Mahboubi and Enrico Tassi. *Mathematical Components*. With contributions by Yves Bertot and Georges Gonthier. 2020. URL: https://math-comp.github.io/mcb/.

[Mai05] Maria Emilia Maietti. "Modular correspondence between dependent type theories and categories including pretopoi and topoi". In: *Mathematical Structures in Computer Science* 15.6 (2005), pp. 1089–1149. DOI: 10.1017/S0960129505004962.

[Mar71] Per Martin-Löf. "A Theory of Types". 1971.

[Mar75a] Per Martin-Löf. "About Models for Intuitionistic Type Theories and the Notion of Definitional Equality". In: *Proceedings of the Third Scandinavian Logic Symposium*. Ed. by Stig Kanger. Vol. 82. Studies in Logic and the Foundations of Mathematics. Elsevier, 1975, pp. 81–109.

[Mar75b] Per Martin-Löf. "An Intuitionistic Theory of Types: Predicative Part". In: *Logic Colloquium '73*. Ed. by H. E. Rose and J. C. Shepherdson. Vol. 80. Studies in Logic and the Foundations of Mathematics. Elsevier, 1975, pp. 73–118. DOI: 10.1016/S0049-237X(08)71945-1.

[Mar87] Per Martin-Löf. "Truth of a Proposition, Evidence of a Judgement, Validity of a Proof". In: *Synthese* 73.3 (1987), pp. 407–420.

[Mar96] Per Martin-Löf. "On the meanings of the logical constants and the justifications of the logical laws". In: *Nordic Journal of Philosophical Logic* 1.1 (1996), pp. 11–60.

[mat20] The mathlib Community. "The Lean Mathematical Library". In: *Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs*. CPP 2020. New Orleans, LA, USA: Association for Computing Machinery, 2020, pp. 367–381. ISBN: 978-1-4503-7097-4. DOI: 10.1145/3372885.3373824.

[Mou+15] Leonardo de Moura, Soonho Kong, Jeremy Avigad, Floris van Doorn, and Jakob von Raumer. "The Lean Theorem Prover (System Description)". In: *Automated Deduction - CADE-25: 25th International Conference on Automated Deduction, Berlin, Germany, August 1-7, 2015, Proceedings*. Ed. by Amy P. Felty and Aart Middeldorp. Cham: Springer International Publishing, 2015, pp. 378–388. ISBN: 978-3-319-21401-6.

[New18] Clive Newstead. "Algebraic Models of Dependent Type Theory". PhD thesis. Carnegie Mellon University, 2018. arXiv: 2103.06155 [math.CT].

[Nor09] Ulf Norell. "Dependently Typed Programming in Agda". In: *Proceedings of the 4th International Workshop on Types in Language Design and Implementation*. TLDI '09. Savannah, GA, USA: Association for Computing Machinery, 2009, pp. 1–2. ISBN: 978-1-60558-420-1.

[Plo73] Gordon D. Plotkin. *Lambda Definability and Logical Relations*. Memorandum SAI-RM-4. University of Edinburgh, 1973. URL: https://homepages.inf.ed.ac.uk/gdp/publications/logical_relations_1973.pdf.

[Plo80] Gordon D. Plotkin. "Lambda-Definability in the Full Type Hierarchy". In: *To H. B. Curry: Essays in Combinatory Logic, Lambda Calculus, and Formalism*. Ed. by J. P. Seldin and J. R. Hindley. New York: Academic Press, 1980, pp. 363–373. URL: https://homepages.inf.ed.ac.uk/gdp/publications/Lambda_Definability.pdf.

[Pra71] Dag Prawitz. "Ideas and Results in Proof Theory". In: *Proceedings of the Second Scandinavian Logic Symposium*. Ed. by J. E. Fenstad. Vol. 63. Studies in Logic and the Foundations of Mathematics. Elsevier, 1971, pp. 235–307. DOI: 10.1016/S0049-237X(08)70849-8.

[RS17] Emily Riehl and Michael Shulman. "A type theory for synthetic ∞-categories". In: *Higher Structures* 1 (1 2017), pp. 147–224. arXiv: 1705.07442 [math.CT]. URL: https://journals.mq.edu.au/index.php/higher_structures/article/view/36.

[RSS20] Egbert Rijke, Michael Shulman, and Bas Spitters. "Modalities in homotopy type theory". In: *Logical Methods in Computer Science* Volume 16, Issue 1 (Jan. 2020). DOI: 10.23638/LMCS-16(1:2)2020. arXiv: 1706.07526 [math.CT]. URL: https://lmcs.episciences.org/6015.

[Rus08] Bertrand Russell. "Mathematical Logic as Based on the Theory of Types". In: *American Journal of Mathematics* 30.3 (1908), pp. 222–262. ISSN: 00029327, 10806377. (Visited on 03/26/2023).

[Rus37] Bertrand Russell. *Principles of Mathematics*. Originally published in 1903. Routledge, 1937.

[Sch22] Peter Scholze. "Liquid Tensor Experiment". In: *Experimental Mathematics* 31.2 (2022), pp. 349–354. DOI: 10.1080/10586458.2021.1926016.

[Shu13] Michael Shulman. *Scones, Logical Relations, and Parametricity*. Blog post. 2013. URL: https://golem.ph.utexas.edu/category/2013/04/scones_logical_relations_and_p.html.

[Shu15] Michael Shulman. "Univalence for inverse diagrams and homotopy canonicity". In: *Mathematical Structures in Computer Science* 25.5 (2015), pp. 1203–1277. DOI: 10.1017/S0960129514000565.

[Sta85] R. Statman. "Logical relations and the typed λ-calculus". In: *Information and Control* 65.2 (1985), pp. 85–97. ISSN: 0019-9958. DOI: 10.1016/S0019-9958(85)80001-2.

[Ste21] Jonathan Sterling. "First Steps in Synthetic Tait Computability: The Objective Metatheory of Cubical Type Theory". Version 1.1, revised May 2022. PhD thesis. Carnegie Mellon University, 2021. DOI: 10.5281/zenodo.6990769.

[SA21] Jonathan Sterling and Carlo Angiuli. "Normalization for Cubical Type Theory". In: *2021 36th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*. Los Alamitos, CA, USA: IEEE Computer Society, July 2021, pp. 1–15. DOI: 10.1109/LICS52264.2021.9470719. arXiv: 2101.11479 [cs.LO].

[SH22] Jonathan Sterling and Robert Harper. "Sheaf semantics of termination-insensitive noninterference". In: *7th International Conference on Formal Structures for Computation and Deduction (FSCD 2022)*. Ed. by Amy P. Felty. Vol. 228. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Aug. 2022, 5:1–5:19. ISBN: 978-3-95977-233-4. DOI: 10.4230/LIPIcs.FSCD.2022.5. arXiv: 2204.09421 [cs.PL].

[Str91] Thomas Streicher. *Semantics of Type Theory: Correctness, Completeness, and Independence Results*. Cambridge, MA, USA: Birkhauser Boston Inc., 1991. ISBN: 0-8176-3594-7.

[Str98] Thomas Streicher. "Categorical intuitions underlying semantic normalisation proofs". In: *Preliminary Proceedings of the APPSEM Workshop on Normalisation by Evaluation*. Ed. by O. Danvy and P. Dybjer. Department of Computer Science, Aarhus University, 1998.

[Str05] Thomas Streicher. "Universes in toposes". In: *From Sets and Types to Topology and Analysis: Towards practical foundations for constructive mathematics*. Ed. by Laura Crosilla and Peter Schuster. Vol. 48. Oxford Logical Guides. Oxford: Oxford University Press, 2005, pp. 78–90. ISBN: 978-0-19-856651-9. DOI: 10.1093/acprof:oso/9780198566519.001.0001.

[Str21] Thomas Streicher. *Fibered Categories à la Jean Bénabou*. 2021. arXiv: 1801.02927 [math.CT].

[Tai67] W. W. Tait. "Intensional Interpretations of Functionals of Finite Type I". In: *The Journal of Symbolic Logic* 32.2 (1967), pp. 198–212. ISSN: 00224812. URL: http://www.jstor.org/stable/2271658.

[Uem21] Taichi Uemura. "Abstract and Concrete Type Theories". PhD thesis. Amsterdam: Universiteit van Amsterdam, 2021. URL: https://www.illc.uva.nl/cms/Research/Publications/Dissertations/DS-2021-09.text.pdf.

[Uem22] Taichi Uemura. *Normalization and coherence for ∞-type theories*. 2022. DOI: 10.48550/ARXIV.2212.11764.

[Uni13] The Univalent Foundations Program. *Homotopy Type Theory: Univalent Foundations of Mathematics*. Institute for Advanced Study: https://homotopytypetheory.org/book, 2013.

[Vic07] Steven Vickers. “Locales and Toposes as Spaces”. In: *Handbook of Spatial Logics*. Ed. by Marco Aiello, Ian Pratt-Hartmann, and Johan Van Benthem. Dordrecht: Springer Netherlands, 2007, pp. 429–496. ISBN: 978-1-4020-5587-4. DOI: 10.1007/978-1-4020-5587-4_8.

[Voe06] Vladimir Voevodsky. “A very short note on homotopy λ-calculus”. Unpublished note. Sept. 2006. URL: https://www.math.ias.edu/Voevodsky/files/files-annotated/Dropbox/Unfinished_papers/Dynamic_logic/Stage_9_2012_09_01/2006_09_Hlambda.pdf.

[Voe16] Vladimir Voevodsky. *Mathematical theory of type theories and the initiality conjecture*. Research proposal to the Templeton Foundation for 2016-2019, project description. Apr. 2016. URL: http://www.math.ias.edu/Voevodsky/other/Voevodsky%20Templeton%20proposal.pdf.

[VAG+] Vladimir Voevodsky, Benedikt Ahrens, Daniel Grayson, et al. *UniMath — a computer-checked library of univalent mathematics*. available at http://unimath.org. URL: https://github.com/UniMath/UniMath.