Syntactic categories for dependent type theory: sketching and adequacy

Daniel Gratzer Jonathan Sterling

December 22, 2020

Abstract

We argue that locally Cartesian closed categories form a suitable doctrine for defining dependent type theories, including non-extensional ones. Using the theory of sketches [KPT99], one may define syntactic categories for type theories in a style that resembles the use of Martin-Löf's Logical Framework [NPS90], following the "judgments as types" principle [HHP93; Mar87].

The concentration of type theories into their locally Cartesian closed categories of judgments is particularly convenient for proving syntactic metatheorems by semantic means (canonicity, normalization, *etc.*). Perhaps surprisingly, the notion of a *context* plays no role in the definitions of type theories in this sense, but the structure of a class of display maps can be imposed on a theory *post facto* wherever needed, as advocated by the Edinburgh school and realized by the `%worlds` declarations of the Twelf proof assistant [HHP93; PS99; HL07].

Uemura [Uem19] has proposed representable map categories together with a stratified logical framework for similar purposes. The stratification in Uemura's framework restricts the use of dependent products to be strictly positive, in contrast to the tradition of Martin-Löf's logical framework [Mar87; NPS90] and Schroeder-Heister's analysis of higher-level deductions [Sch87].

We prove a semantic adequacy result for locally Cartesian closed categories relative to Uemura's representable map categories: if a theory is definable in the framework of Uemura, the locally Cartesian closed category that it generates is a conservative (fully faithful) extension of its syntactic representable map category. On this basis, we argue for the use of locally Cartesian closed categories as a simpler alternative to Uemura's representable map categories.

1 Introduction

(1·1) What kind of objects are dependent type theories and their models? Unfortunately there are many possible answers to this question:

1) *Comprehension categories* [Jac99] express the structure of a category of contexts equipped with separate notions of type and term, connected by a context extension operation. Sometimes a comprehension category is "split" (modeling strictly associative substitution).

2) *Categories with attributes* [Car78] are *full* split comprehension categories; fullness means that the notion of an element is *derived* from the context extension.

3) *Categories with families* [Dyb96] express the notion of type, term, and context extension as a special kind of *universe* in the category of presheaves over a category of contexts. These are easily seen to be equivalent to categories with attributes, but they are arguably more type theoretic in style.

   Awodey [Awo18b] and Fiore [Fio12] have independently reformulated categories with families in terms of *representable natural transformations*; Awodey has coined the name *natural model* for this formulation.

4) *Contextual categories* [Car78] or *C-systems* [Voe15] are categories with attributes together with a structure that equips each context with a "length", reflecting the inductive generation of contexts in some presentations of the raw syntax of type theory.

5) *Display map categories* [Tay86; Tay99] or *clans* [Joy17] express the data of a category of contexts equipped with a class of "display maps" that generalize context extensions.

(1·2) The notion of a *contextual category* or a *C-system* is not as useful as it might at first seem: contextual categories differ from categories with attributes only by elevating to the status of a definition the incidental aspect of certain raw syntax presentations of type theory that contexts have a length. This influence of (raw) syntax on semantics has so far imparted no practical leverage, even when proving metatheorems of a syntactical nature: indeed, no theorem of dependent type theory could ever have depended on the fact that not every context is of length *one*.

Categories with attributes and natural models are in essence the same notion. Comprehension categories on the other hand occupy an awkward position: full split comprehension categories are the same as categories with attributes and natural models, and full non-split comprehension categories are the same as clans. From our perspective, the *canonical* notions among the above are therefore clans and natural models, representing the weak and strict notions of dependent type theory respectively.

(1·3) We have enumerated a number of scientific hypotheses as to what a model of type theory ought to be; aside from clans, however, **(1·1)** does not pose a definition of the *syntactic/classifying category* of a given type theory in the sense of functorial semantics [Law63]. Recently Uemura [Uem19] has proposed *representable map categories* to serve as the syntactic categories that classify the natural models of a given type theory, which we discuss below in Section 1.1.

1.1 Uemura's representable map categories

(1.1·1) A representable map category is a finitely complete category $\mathscr{T}$ equipped with a pullback stable subcategory $\mathscr{T}^{rep} \subseteq \mathscr{T}$ of "representable maps" such that $\mathscr{T}$ contains dependent products along representable maps. Representable maps correspond roughly to display maps, and an object $\Gamma : \mathscr{T}$ such that $\Gamma \longrightarrow \mathbf{1}_{\mathscr{T}}$ is representable can be thought of as a *context*. An arbitrary (non-representable) object of $\mathscr{T}$ stands for a *judgment* — something that will be taken to a not necessarily representable presheaf in the natural model semantics.

(1.1·2) While a clan expresses the categorical structure of the category of contexts of a given type theory, a representable map category attempts to express the categorical structure of its *category of judgments*. In particular, while a clan need not be finitely complete (the diagonal is not a display map except in extensional type theories), unrestricted pullback in representable map categories corresponds to the fact that type theory has *judgmental equality.* Likewise, the presence of (non-representable) dependent products along representable maps corresponds exactly to the *hypothetico-general judgment* of dependent type theory in the sense of Martin-Löf [Mar96].

(1.1·3) The fact that dependent products need exist only along representable maps corresponds to the way that dependent type theory is conventionally presented using hypothetical judgments of one level only (corresponding by transpose to context extension). This realistic stratification, however, is not at all forced: Martin-Löf himself has promoted a presentation of the syntax of dependent type theory that supports hypothetical judgments of arbitrary level [NPS90; Mar87; Sch87].

One commonly cited example of the use of higher-level judgments is to present dependent products in terms of a "fun-split" operator [NPS90], but this example is not so convincing considering that the presentation is strictly isomorphic to one not involving a higher-level judgment.[1] A more convincing example is furnished by the W-type, whose elimination rule apparently cannot even be written down without higher-level hypothetical judgments in the absence of dependent product types.

1.2 Locally Cartesian closed categories

(1.2·1) If Uemura's notion of representable map category aims to capture the restriction of a category of judgments to contain just the hypothetical judgments tracked by context extensions, it is reasonable to consider the categorical structure of judgments *absent* such a stratification. Of course, as soon as we have both judgmental equality and unrestricted hypothetical judgment, the category of judgments is nothing less than a locally Cartesian closed category.

Such syntactic category may be equipped with a class of display maps, but this class is no longer intertwined with the definition of the syntactic category. In contrast, one cannot even write down the closure of the type theory *qua* Uemura [Uem19] under *(e.g.)* function types unless certain maps are representable.

(1.2·2) Because the category of judgments of a given type theory can be defined independently of the structure of a class of display maps / notion of context, it is in fact reasonable to avoid imposing such a structure until it is needed. The need for a notion of context arises in several situations:

1) Tautologically, one needs a notion of context when defining a formal "gammas and turnstiles" presentation of a given type theory.
2) To correctly state results like decidability of judgmental equality, one also needs a notion of context: judgmental equality can only decidable relative to a class of display maps that *does not include* all diagonals [CCD17].
3) A class of display maps can be used to present an ∞-categorical structure, as in Joyal's clans [Joy17] and Lurie's pre-geometries [Lur09a].

The idea of identifying the relevant display maps separately from a theory and locally to a given construction was promoted and used to great effect by the exponents

[1] *Pace* the observation of Garner [Gar09] that the fun-split formulation is strictly stronger than the conventional formulation *in the absence of the η-law* — perhaps a more realistic title for the cited work would have been "On the strength of dependent *non-products* in the type theory of Martin-Löf".

of the Edinburgh school of logical frameworks [HHP93; HL07]; the function of the `%worlds` declarations of the Twelf proof assistant [PS99] is nothing more than to specify the class of contexts relative to which a given metatheorem should hold, since almost no metatheorems hold unconditionally.

(1.2·3) While the presence of higher-level hypothetical judgments in the language of locally Cartesian closed categories is convenient, it is *a priori* correct to worry about whether it evinces an exotic (and therefore inadequate) notion of syntax for a conventional type theory. In Section 5, we prove that the presence of hypothetical judgments of arbitrary level is a *conservative* extension of the stratified language of Uemura's representable map categories; our result can be seen as a **semantic adequacy theorem** for a encodings of type theories as locally Cartesian closed categories.

1.3 Presenting syntactic categories by generators and relations

(1.3·1) While it is very simple to describe how an algebraic theory can be built from generators and relations, it is more technical to do so for the syntactic category of a dependent type theory. In essence this is because the collection of derived generators of a given dependent type theory appears *a priori* to depend on all the relations of that theory, a difficulty that can be tracked to the presence of the *conversion rule*:

$$\frac{\Gamma \vdash \mathrm{M} : \mathrm{A} \qquad \Gamma \vdash \mathrm{A} \equiv \mathrm{B}\ \textit{type}}{\Gamma \vdash \mathrm{M} : \mathrm{B}}$$

Moreover, the sort of a given generator may need to mention a pullback or dependent product that involves some other generators. Hence it is not immediately clear how to define all the generators before freely adding *(e.g.)* dependent products.

(1.3·2) *Logical frameworks.* One approach to "tie the knot" between generators, relations, and derived structure (pullback, dependent products, *etc.*) is to use a *logical framework*, in which these notions are all interleaved; this is the approach of Cartmell [Car78], Nordström, Peterson, and Smith [NPS90], Uemura [Uem19], and Harper [Har20].[2] Logical frameworks are very user-friendly, but it can be somewhat technical to make the connection between the syntactical artifacts of a logical framework and the categorical/mathematical objects they are meant to present.

(1.3·3) *Sketches.* A less syntactic and vastly more general way to present almost any kind of theory by generators and relations is furnished by the language of *sketches* for a given (finitary) 2-monad or "doctrine":

1) In the doctrine of categories with finite products, sketches present algebraic theories.
2) In the doctrine of categories with finite limits, sketches present essentially algebraic theories.
3) In the doctrine of locally Cartesian closed categories, sketches present (the categories of judgments of) dependent type theories.

[2]Harper, Honsell, and Plotkin [HHP93] also define a logical framework, but this logical framework only supports "pure" theories with only generators and no relations.

Sketches address the apparent interleaving between generators, relations, and derived structure in a different way from logical frameworks **(1.3·2)**. The generators are defined all at once, prior to imposing relations or adding derived structure; if a generator in a finite limit sketch needs to mention *(e.g.)* a pullback, this object is added *formally* as an additional generator and then "marked" as something that needs to become a real pullback when the sketch is knitted together into a finitely complete category. This picture works for any kind of theory that can be defined as a 2-monad!

(1.3·4) The benefit of 2-monads and their sketches is that they are considerably more general than any specific logical framework, and they likely encompass the expressivity of almost any future logical framework. Moreover, the connection between a sketch and the theory it presents can be rigorously developed once and for all without needing to deal with the purely bureaucratic aspects of syntax (*e.g.* variable binding, conversion, presupposition lemmas, *etc.*) that have engrossed and preoccupied some parts of the type theoretic research community for a number of years. Once a scientist understands the language of sketches, there is nothing to stop her from using a logical framework as a convenient *notation*, keeping in mind the "compilation" to the more basic notion.

(1.3·5) For the sake of exposition, we recall the basics of 2-monad theory and sketches from Kinoshita, Power, and Takeyama [KPT99] in Sections 2 and 3. In Section 4 we argue for the use of locally Cartesian closed categories as a convenient way to present the syntactic categories of (strict) dependent type theories. Only the conservativity result of Section 5 is novel.

2 Doctrines and the theory of 2-monads

(2·1) There are many different kinds of theory, each corresponding to different kinds of categorical structure: for instance, the structure of finite products, finite limits, finite colimits, infinite colimits, exponentials, etc. all give rise to different "notions of theory" or *doctrines*. The intuitive idea of a doctrine is realized at a technical level by the notion of a 2-monad, exposed in Section 2.1 below.

(2·2) In this section we recall the basic 2-monad theory necessary for this note. We emphasize that a deep understanding of this material not a prerequisite for sketching type theories; a casual reader may take **(2.2·8)** as given and proceed to Section 3.

2.1 The theory of 2-monads

(2.1·1) A strict 2-monad $\mathbf{T}$ on $\mathbf{Cat}$ (resp. $\mathbf{Cat}_g$) is an ordinary monad on $\mathbf{Cat}$ regarded as a 1-category, admitting an enrichment over $\mathbf{Cat}$ (resp. $\mathbf{Grpd}$).[3] The enrichment means that $\mathbf{T}$ must not only assign to each functor $f : \mathscr{C} \longrightarrow \mathscr{D}$ a functor $\mathbf{T}(f) : \mathbf{T}(\mathscr{C}) \longrightarrow \mathbf{T}(\mathscr{D})$: it also assigns to each 2-cell $\alpha : f \longrightarrow g$ a 2-cell $\mathbf{T}(\alpha) : \mathbf{T}(f) \longrightarrow \mathbf{T}(g)$, *strictly* respecting identity and composition.

(2.1·2) While we will not be overly pedantic about the details, we must discuss the notion of a monad algebra for a 2-monad $\mathbf{T}$. A monad algebra is a category $\mathscr{A}$ together with a morphism $a : \mathbf{T}(\mathscr{A}) \longrightarrow \mathscr{A}$ which satisfies the following two equations, familiar

[3]Note that **Cat** or **Grpd** enrichment of a monad (indeed, an endofunctor) is merely a property, not a structure [Pow11].

from the 1-categorical case:

$$\begin{tikzcd}
\mathscr{A} \arrow[r, "\eta_{\mathscr{A}}"] \arrow[dr, equal] & \mathbf{T}(\mathscr{A}) \arrow[d, "a"] \\
& \mathscr{A}
\end{tikzcd}
\qquad
\begin{tikzcd}
\mathbf{T}(\mathbf{T}(\mathscr{A})) \arrow[r, "\mathbf{T}(a)"] \arrow[d, "\mu_{\mathscr{A}}"'] & \mathbf{T}(\mathscr{A}) \arrow[d, "a"] \\
\mathbf{T}(\mathscr{A}) \arrow[r, "a"'] & \mathscr{A}
\end{tikzcd}$$

The notion of morphism between **T**-algebras differs from the 1-categorical case as we are primarily concerned with a weaker morphism between monad algebras than the 1-dimensional case. In particular, a morphism of monad algebras $f : (\mathscr{A},a) \longrightarrow (\mathscr{B},b)$ is a functor $f : \mathscr{A} \longrightarrow \mathscr{B}$ equipped with an natural isomorphism $\alpha : b \circ \mathbf{T}(f) \longrightarrow f \circ a$. The natural isomorphism is required to satisfy certain coherence conditions, but we elide these here and refer the interested reader to Blackwell, Kelly, and Power [BKP89].

(2.1·3) The category of **T**-algebras, weak morphisms, and natural transformations organizes into a 2-category **T-Alg**. We say that a 2-category $\mathscr{E}$ is 2-monadic over **Cat** when it is 2-equivalent to the **T-Alg** for some **T**.

(2.1·4) Despite the strictness of 2-monads and their algebras, examples of 2-monads on **Cat** and $\mathbf{Cat}_g$ are plentiful. For instance, the functor assigning $\mathscr{C}$ to its finite limit completion $\mathscr{C} \longmapsto \mathscr{C}^{\mathsf{lex}}$ is a 2-monad and 2-category **Lex** is 2-monadic over **Cat**. Explicitly, **Lex** is equivalent to the category of monad algebras for $(-)^{\mathsf{lex}}$.

Monoidal, symmetric monoidal or finitely cocomplete categories are also all 2-monadic over **Cat**. The theory is developed systematically by Blackwell, Kelly, and Power [BKP89], but generally the locally full 2-subcategory of **Cat** consisting of structured categories and structure-preserving morphisms is 2-monadic over **Cat**.

The exception to the rule is categories with structure like exponentials or dependent products, as they behave contravariantly; here the enrichment can then be made only over the wide subcategory $\mathbf{Grpd} \subseteq \mathbf{Cat}$.

2.2 Constructing 2-monads from generators and relations

(2.2·1) The class of *finitary* 2-monads proves to be of particular importance. Recall that in 1-category theory, a monad is finitary when it preserves filtered colimits; the signifance of finitary monads in 1-category is that they correspond to (finitary) algebraic theories — in other words, finitary 1-monads can be presented by generators and relations.

The same notions can be adapted to any presentable category, and generalized to a presentable 2-category (for instance, **Cat** and $\mathbf{Cat}_g$) [KP93]. This generalization allows us to define present *2-monads* by generators and relations, just like finitary 1-monads. Many natural 2-monads are finitary, including all the examples presented above (finitely complete categories, monoidal categories, *etc.*).

(2.2·2) A more casual explanation of *generators and relations* perspective on finitary 2-monads is given by Kinoshita, Power, and Takeyama [KPT99], and we present some of the explanation given there but restricting to the case of **Cat** ($\mathbf{Cat}_g$ is analogous).

(2.2·3) We define a functor $\mathsf{S} : \mathbb{S} \longrightarrow \mathbf{Cat}$ of *operations*, where $\mathbb{S}$ is the discrete category of isomorphism classes of finite categories. The finitary categories generalize the notion of *arity* from classical algebraic theories by allowing for a finite collection of objects and arrows to be specified as input to an operation. Given an isomorphism class **c** of finitary categories, the category $\mathsf{S}(\mathbb{c})$ is defined to be the product of the shape of results of all the operations with input arity $\mathbb{c}$.

An S-algebra is a category $\mathscr{A}$ equipped with functors $\nu_{\mathbb{c}} : [\mathbb{c},\mathscr{A}] \longrightarrow [\mathsf{S}(\mathbb{c}),\mathscr{A}]$ assigning a choice of input to the output associated with each operation. A morphism of algebras $f : (\mathscr{A},\nu_\bullet) \longrightarrow (\mathscr{B},\mu_\bullet)$ is a functor between carriers that satisfies the following equation at each $\mathbb{c} : \mathbb{S}$:

$$\begin{tikzcd}
{[\mathbb{c},\mathscr{A}]} \arrow[r, "\nu_{\mathbb{c}}"] \arrow[d, "\mu_{\mathbb{c}}"'] & {[\mathsf{S}(\mathbb{c}),\mathscr{A}]} \arrow[d, "{[\mathsf{S}(\mathbb{c}),f]}"] \\
{[\mathsf{S}(\mathbb{c}),\mathscr{A}]} \arrow[r, "{[\mathsf{S}(\mathbb{c}),f]}"'] & {[\mathsf{S}(\mathbb{c}),\mathscr{B}]}
\end{tikzcd}$$

(2.2·4) In order to specify the equational part of a presentation of a finitary 2-monad, we must generate the derived operations of a theory; these are the operations that can be built by combining the various primitive operations specified by S. We will define the functor of derived operations $\mathsf{S}_\omega : \mathbb{S} \longrightarrow \mathbf{Cat}$ by taking a direct limit over the derived operations of each possible finite depth.

We define the nth functor of derived operations $\mathsf{S}_n : \mathbb{S} \longrightarrow \mathbf{Cat}$ by induction on $n \in \mathbb{N}$, writing $\mathrm{J} : \mathbb{S} \hookrightarrow \mathbf{Cat}$ for the evident embedding:

$$\begin{aligned}
\mathsf{S}_0(\mathbb{c}) &= \mathrm{J}(\mathbb{c}) \\
\mathsf{S}_{n+1}(\mathbb{c}) &= \underbrace{\mathrm{J}(\mathbb{c})}_{\textit{vars.}} + \underbrace{\overbrace{\textstyle\sum_{\mathbb{d}\in\mathbb{S}}}^{\textit{arity}} \overbrace{[\mathbb{d},\mathsf{S}_n(\mathbb{c})]}^{\textit{arguments}} \times \overbrace{\mathsf{S}(\mathbb{d})}^{\textit{symbol}}}_{\textit{operations}}
\end{aligned}$$

There is a canonical map (often a monomorphism) $\mathsf{S}_n \longrightarrow \mathsf{S}_{n+1}$; hence we may define S_ω to be the colimit of the resulting sequence. A routine computation shows that an S-algebra structure $(\mathscr{A},\nu_\bullet)$ induces an S_ω-algebra structure $(\mathscr{A},\nu^\omega_\bullet)$ by induction.

(2.2·5) With S_ω in hand, we may specify the equations of a given theory through a functor $\mathsf{E} : \mathbb{S} \longrightarrow \mathbf{Cat}$ together with a pair of natural transformations $\tau_1,\tau_2 : \mathsf{E} \longrightarrow \mathsf{S}_\omega$. Roughly, $\mathsf{E}(\mathbb{c})$ specifies the shape of the equations imposed on the theory between terms with free variables of shape $\mathbb{c}$, and the natural transformations τ_1/τ_2 interpret this shape into actual configurations of terms.

An S-algebra $(\mathscr{A}, \nu_\bullet)$ satisfies the equations specified by E when the following diagram commutes for each $\mathbb{c} : \mathbb{S}$:

$$\begin{tikzcd}
{[\mathbb{c},\mathscr{A}]} \arrow[r, "\nu^\omega_{\mathbb{c}}"] & {[\mathsf{S}_\omega(\mathbb{c}),\mathscr{A}]} \arrow[r, shift left, "{[\tau_1^{\mathbb{c}},\mathscr{A}]}"] \arrow[r, shift right, "{[\tau_2^{\mathbb{c}},\mathscr{A}]}"'] & {[\mathsf{E}(\mathbb{c}),\mathscr{A}]}
\end{tikzcd} \tag{2.2·5·1}$$

Specializing again to classical algebraic theories, this diagram commutes when $\mathscr{A}$ satisfies the equations specified by E after instantiating the free variables with arbitrary elements of $\mathscr{A}$.

(2.2·6) If $(\mathscr{A},\nu)$ is a S-algebra for which Diagram 2.2·5·1 commutes for each $\mathbb{c}$, we call it an $\langle\mathsf{S},\mathsf{E}\rangle$-algebra. We take the category of $\langle\mathsf{S},\mathsf{E}\rangle$-algebras to be the evident full subcategory of S-algebras.

(2.2·7) For any $\langle \mathsf{S},\mathsf{E}\rangle$, there exists a finitary 2-monad $\mathbf{T}$ and an equivalence between the category of $\langle \mathsf{S},\mathsf{E}\rangle$-algebras and $\mathbf{T}$-algebras.

(2.2·8) The machinery exposed in this section directly implies that locally Cartesian closed categories can be characterized by a finitary 2-monad on $\mathbf{Cat}_g$. By a general result, therefore, **LCCC** is bi-cocomplete and pseudo-complete [Kel89; BKP89].[4]

2.2.1 An extended example: Lex

(2.2.1·0) For expository purposes we demonstrate how to construct the 2-monad presenting **Lex**, the 2-category of finitely complete categories.

(2.2.1·1) It is well-known that a category is finitely complete when it contains a terminal object and all pullbacks. Both of these conditions can be encoded as a pair of operations, with one operation producing the terminal object (resp. pullback) and a second equipping it with the distinguished maps provided by its universal property.

(2.2.1·2) *The terminal object.* The operation that produces the terminal object requires no input and produces a single object as output. The universal property takes a single object X and extends it to a map $\mathrm{X} \longrightarrow \mathbf{1}$. Both of these are encoded as operations:

$$\mathsf{S}(\{\}) = \{\mathtt{T}\} \qquad \mathsf{S}(\{\bullet\}) = \{\mathtt{t}_\bullet : \partial_0 \mathtt{t}_\bullet \longrightarrow \partial_1 \mathtt{t}_\bullet\}$$

(2.2.1·3) *Pullbacks.* We require a similar set of operations for pullbacks, though they take a span of objects as input, and produce a commuting diagram as output:

$$\mathsf{S}(\{\bullet \longrightarrow \bullet \longleftarrow \bullet\}) = \mathscr{C} \qquad \mathsf{S}(\mathbb{d}) = \mathscr{D}$$

We define $\mathscr{C}$, $\mathbb{d}$ and $\mathscr{D}$ to be the following categories:

$$\mathscr{C} = \mathbb{d} = \begin{tikzcd} \bullet \arrow[r] \arrow[d] & \bullet \arrow[d] \\ \bullet \arrow[r] & \bullet \end{tikzcd} \qquad \mathscr{D} = \begin{tikzcd} \bullet \arrow[rd] \arrow[rrd, bend left] \arrow[rdd, bend right] & & \\ & \bullet \arrow[r] \arrow[d] & \bullet \arrow[d] \\ & \bullet \arrow[r] & \bullet \end{tikzcd}$$

(2.2.1·4) On their own, the operations specified in **(2.2.1·2)** and **(2.2.1·3)** are not sufficient to ensure that an algebra has terminal objects and pullbacks. To be concrete, let $(\mathscr{A},\nu)$ be an algebra for S. There is a distinguished object $\mathtt{T}$ induced by the first operation of **(2.2.1·2)** and a choice of map $\mathtt{t}_\mathrm{A} : \partial_0 \mathtt{t}_\mathrm{A} \longrightarrow \partial_1 \mathtt{t}_\mathrm{A}$ for each $\mathrm{A} : \mathscr{A}$ induced by the second operation; so far, nothing we have done has ensured that fixed the boundary of $\mathtt{t}_\mathrm{A}$ to be anything in particular.

Recall that in the enriched setting, the algebra maps $\nu_\mathbb{c} : [\mathbb{c},\mathscr{A}] \longrightarrow [\mathsf{S}(\mathbb{c}),\mathscr{A}]$ making $\mathscr{A}$ an algebra are functors, not merely functions between sets. The functoriality conditions on maps ensure that certain naturality conditions between operators are automatically enforced. In the case of the first operation the condition is trivial, but for the second operation we obtain a commuting square for each map $f : \mathrm{A} \longrightarrow \mathrm{B}$:

[4]In fact, **LCCC** is closed under PIE-limits [nLa17].

$$\begin{tikzcd}
\partial_0 \mathtt{t}_{A} \arrow[r, "\mathtt{t}^0_f"] \arrow[d, "\mathtt{t}_{A}"'] & \partial_0 \mathtt{t}_{B} \arrow[d, "\mathtt{t}_{B}"] \\
\partial_1 \mathtt{t}_{A} \arrow[r, "\mathtt{t}^1_f"'] & \partial_1 \mathtt{t}_{B}
\end{tikzcd} \tag{2.2.1·4·1}$$

To make T into a terminal object is, then, we must add *equations* that ensure $\partial_0 \mathtt{t}_A = A$ and $\partial_1 \mathtt{t}_A = \mathsf{T}$, and to ensure that $\mathtt{t}_A$ is the unique map with this property. We explore this process in **(2.2.1·5)** and **(2.2.1·6)**.

(2.2.1·5) We will add a equations to ensure that $\mathtt{t}_A$ has the correct boundary. In fact, we will do more than this and ensure that $\mathtt{t}$ has the correct functorial behavior by adding two equations: one ensuring for every $f : A \longrightarrow B$ that $\mathtt{t}^0_f = f$ and the other ensures $\mathtt{t}^1_f = \mathbf{id}_{\mathsf{T}}$. We prepare to add these two equations by defining E at $\{\bullet \longrightarrow \bullet\}$ to be the following category:

$$\mathsf{E}(\bullet \longrightarrow \bullet) = \{\bullet \xrightarrow{\mathtt{t/eq/f\text{-}dom}} \bullet \qquad \bullet \xrightarrow{\mathtt{t/eq/f\text{-}cod}} \bullet\}$$

Recall that τ_1,τ_2 must be natural transformations from E to S_ω; this means that the component of τ_1 at $\{\bullet \longrightarrow \bullet\}$ must send an equation symbol from $\mathsf{E}(\{\bullet \longrightarrow \bullet\})$ to a derived term with one free variable. Hence, we extend the definition of τ_1,τ_2 at $\{\bullet \longrightarrow \bullet\}$ to specify the two equations:

$$\begin{aligned}
\tau_1(\mathtt{t/eq/f\text{-}dom}) &= (f : A \longrightarrow B \vdash \mathtt{t}^0_f) & \tau_1(\mathtt{t/eq/f\text{-}cod}) &= (f : A \longrightarrow B \vdash \mathtt{t}^1_f) \\
\tau_2(\mathtt{t/eq/f\text{-}dom}) &= (f : A \longrightarrow B \vdash f) & \tau_2(\mathtt{t/eq/f\text{-}cod}) &= (f : A \longrightarrow B \vdash \mathbf{id}_{\mathsf{T}})
\end{aligned}$$

(2.2.1·6) We also wish to ensure that $\mathtt{t}_A$ is the unique arrow from $A \longrightarrow \mathsf{T}$. This is done in a somewhat roundabout manner: we add an equation with no free variables ensuring that $\mathtt{t}_{\mathsf{T}} : \mathsf{T} \longrightarrow \mathsf{T}$ is the identity. When Diagram 2.2.1·4·1 is instantiated with $f : A \longrightarrow \mathsf{T}$, we have $\mathtt{t}_{\mathsf{T}} \circ \mathtt{t}^0_f = \mathtt{t}^1_f \circ \mathtt{t}_A$. We have just required $\mathtt{t}_{\mathsf{T}} = \mathbf{id}_{\mathsf{T}}$ and **(2.2.1·5)** ensures $\mathtt{t}^0_f = f$ and $\mathtt{t}^1_f = \mathbf{id}_{\mathsf{T}}$. Thus, we have $f = \mathtt{t}_A$ as required.

(2.2.1·7) The equations for the pullback operations are similar. For each cospan $A_0 \longrightarrow A_2 \longleftarrow A_1$ we obtain a square, and for each square we obtain a "gap map"; we add equations to fix the boundary of this square to extend the cospan, and to fix the boundary of the gap map. Finally, an equation is added to force the gap map the purported pullback square itself to be the identity; combined with naturality this equation ensures that the gap map is unique, and hence that the square is in fact a pullback square.

(2.2.1·8) In the case of (locally) Cartesian closed categories, the enrichment is taken over $\mathbf{Cat}_g$, not $\mathbf{Cat}$; this is a consquence of the contravariance involved in exponentials. Accordingly the functorial conditions for the operations are far weaker, with naturality squares guaranteed only for isomorphisms.

The consequence is that Diagram 2.2.1·4·1 must be added as an additional equation when defining (local) cartesian closure as a 2-monad, because it would no longer directly follow from the enrichment. We would also similarly add equations to enforce the appropriate functoriality of the transposition operation for (local) exponentials.

3 Sketching theories in a doctrine

(3·1) In Section 2.2 we outlined the proof that **LCCC** is finitarily 2-monadic over $\mathbf{Cat}_g$; hence we may define the free locally Cartesian closed category $\mathbf{T}(\mathscr{C})$ on any category $\mathscr{C}$. As a free object, $\mathbf{T}(\mathscr{C})$ enjoys the following universal property: naturally in a given locally Cartesian closed category $\mathscr{E}$, we have $\mathrm{Hom}_{\mathbf{Cat}_g}(\mathscr{C},\mathscr{E}) \cong \mathrm{Hom}_{\mathbf{LCCC}}(\mathbf{T}(\mathscr{C}),\mathscr{E})$. In other words, locally Cartesian closed functors out of $\mathbf{T}(\mathscr{C})$ are completely determined by their restriction to $\mathscr{C}$.

(3·2) On its own, the 2-monad of local Cartesian closure already allows us to construct particularly simple type theories. For instance, the *walking type theory* generated by a single morphism $\varpi : \dot{\mathscr{U}} \longrightarrow \mathscr{U}$ can be defined by the free locally Cartesian closed category generated by a single morphism.

(3·3) In order to construct more realistic theories, however, we must be able to specify particular connectives whose boundaries involve dependent products and pullbacks. For instance, to specify the closure of the "walking type theory" **(3·2)** under dependent products, one needs to add a cartesian morphism $\mathrm{P}_\varpi(\varpi) \longrightarrow \varpi$ where P_ϖ is the polynomial endofunctor on ϖ — but the definition P_ϖ mentions dependent products:

$$\mathrm{P}_\varpi(\mathrm{X}) = \textstyle\sum_{\mathrm{A}:\mathscr{U}} \prod_{x:\varpi[\mathrm{A}]} \mathrm{X}$$

In essence the difficulty is that one seems to need to interleave the specification of generators with the free extension by locally Cartesian closed structure. One way to do this is using a syntactic logical framework; an alternative method, which we explain here, is the theory of *sketches* [Law63; Ehr66; Ehr68b; Ehr68a].[5]

(3·4) The classical theory of sketches is restricted to developing theories involving finite (co)limits, but Kinoshita, Power, and Takeyama [KPT99] have given a generalization that works for 2-monads more generally. We now briefly recall the classical theory of finite limit sketches before introducing this generalization, and refer the reader to Adámek and Rosický [AR94] for a textbook account.

3.1 Finite limit sketches

(3.1·1) Let $\mathscr{D}$ be a category; the we write $\mathscr{D}^\triangleleft$ for the *cone* above $\mathscr{D}$, which is the *join* of categories $\{\mathbf{0}\} \star \mathscr{D}$ in the sense of Lurie [Lur09b]. Explicitly $\mathscr{D}^\triangleleft$ freely adjoins an initial object to $\mathscr{D}$.

(3.1·2) Given a category $\mathscr{C}$, a *finite cone* in $\mathscr{C}$ is just a functor $\mathbb{d}^\triangleleft \longrightarrow \mathscr{C}$ for some finitary category $\mathbb{d}$.

(3.1·3) A *finite limit sketch* is defined to be a category $\mathscr{S}$ (not necessarily finitely complete) together with a collection of finite cones $\{d_i : \mathbb{d}_i^\triangleleft \longrightarrow \mathscr{S}\}$; the cones d_i are called *marked*. We will often use $\mathscr{S}$ metonymically to refer to the pair $(\mathscr{S}, d_\bullet)$.

(3.1·4) The purpose of $\mathscr{S}$ is to specify the generating objects and morphisms for our theory; because the (intended) boundaries of generating morphisms frequently involve

[5] In some traditional accounts, the data of a sketch is concentrated in a *directed graph* equipped with a choice various subgraphs that will be realized by commutative diagrams in a category. We follow Kinoshita, Power, and Takeyama [KPT99] in starting directly from categories and ignoring the directed graph structure; our choices reflects the fact that it is not so difficult to freely generate a category by a collection of diagrams.

not only generating objects but finite limits thereof, however, we will include "dummy generators" for these boundaries that will subsequently be *marked* in the sense of **(3.1·3)**. This is the solution that the language of sketches offers to the problem of interleaving specification with free generation.

A finite cone that is "marked" does not necessarily have a universal property in $\mathscr{S}$, so there are no conditions to check when marking a cone. The chosen markings are then put to use in the definition of *models* of a sketch in **(3.1·5)**, in which marked cones are required to be realized by actual limit cones.

(3.1·5) We define a model of $\mathscr{S}$ in a finitely complete category $\mathscr{E}$ to be a functor $f : \mathscr{S} \longrightarrow \mathscr{E}$ such that $f \circ d_i$ is limiting for each marked cone d_i. We write $\mathbf{Mod}(\mathscr{S}, \mathscr{E})$ for the category of $\mathscr{E}$-valued models of $\mathscr{S}$ and natural transformations between them.

(3.1·6) In practice, this means that while the sketch of a group might not contain the product $\mathrm{G} \times \mathrm{G}$, we can mark the domain of the multiplication operation $m : \mathrm{P} \longrightarrow \mathrm{G}$ so that any model of the sketch must realize P as $\mathrm{G} \times \mathrm{G}$.

(3.1·7) The central theorem of finite limit sketches states that there is exists a finite limit category that completes a sketch $\mathscr{S}$ by adding the missing limits while ensuring that marked cones become proper limits. More precisely, there is a functor from the category of sketches to **Lex** that induces the following equivalence of categories:

$$\mathrm{Hom}_{\mathbf{Lex}}(\mathbf{Th}(\mathscr{S}), \mathscr{C}) \simeq \mathbf{Mod}(\mathscr{S}, \mathscr{C})$$

3.2 Generalized sketches

(3.2·1) In order to generalize **(3.1·7)** to locally Cartesian closed categories, we will replace finite limits with an arbitrary finitary 2-monad $\mathbf{T}$ following the work of Kinoshita, Power, and Takeyama [KPT99]. In particular, we generalize the theory of sketches to allow for diagrams to be marked with structure from $\mathbf{T}$, and replace finitely complete categories with $\mathbf{T}$-algebras.

When we instantiate $\mathbf{T}$ with the presentation of **Lex** discussed in Section 2.2.1, the results in the more general formulation will specialize to those of **(3.1·7)**.

(3.2·2) The most complex piece of the generalized sketching machinery is the replacement for marking limits. We accomplish this by breaking a marked diagram into two components. Rather than a single finitary category with an initial object, we require a pair of finitary categories $(\mathbb{c}, \mathbb{d})$ along with a commuting triangle:

$$\begin{tikzcd} \mathbb{c} \arrow[d, "j"'] \arrow[dr, "\eta_{\mathbb{c}}"] & \\ \mathbb{d} \arrow[r, "k"'] & \mathbf{T}(\mathbb{c}) \end{tikzcd} \tag{3.2·2·1}$$

The finitary category $\mathbb{c}$ contains the *generators* involved in the marked diagram; $\mathbb{d}$ is the actual shape of the diagram, and the functor $k : \mathbb{d} \longrightarrow \mathbf{T}(\mathbb{c})$ explains how the diagram is intended to be realized in terms of the $\mathbf{T}$-structure generated by $\mathbb{c}$. The fact that Diagram 3.2·2·1 commutes is a well-formedness condition on a marked diagram to ensure that the "generators" are taken to their avatars in the realization.

(3.2·3) A sketch for a 2-monad $\mathbf{T}$ is a category $\mathscr{S}$ equipped with a set of quadruples $\{(\mathbb{c}_i,\mathbb{d}_i,j_i,k_i) \mid k_i \circ j_i = \eta_{\mathbb{c}_i}\}$ in the sense of Diagram 3.2·2·1 together with functors functors $\phi_i : \mathbb{d}_i \longrightarrow \mathscr{S}$.

(3.2·4) A model of a sketch $\mathscr{S}$ in a $\mathbf{T}$-algebra $(\mathscr{A}, a)$ is a functor $f : \mathscr{S} \longrightarrow \mathscr{A}$ together with a collection of natural isomorphisms α_i witnessing the weak commutativity of the black square in Diagram 3.2·4·1 below:

$$\begin{tikzcd}
\mathbf{T}(\mathbb{c}_i) \arrow[r, "\mathbf{T}(j_i)"] & \mathbf{T}(\mathbb{d}_i) \arrow[r, "\mathbf{T}(\phi_i)"] & \mathbf{T}(\mathscr{S}) \arrow[d, "\mathbf{T}(f)"] \\
\mathbb{d}_i \arrow[u, "k_i"] \arrow[rr, "\mathbf{T}(f\phi_i j_i) \circ k_i"] \arrow[d, "\phi_i"'] \arrow[drr, phantom, "\alpha_i"] & & \mathbf{T}(\mathscr{A}) \arrow[d, "a"] \\
\mathscr{S} \arrow[rr, "f"'] & & \mathscr{A}
\end{tikzcd} \tag{3.2·4·1}$$

The diagram above states, in essence, that the marked diagrams in $\mathscr{S}$ are taken by the functor f to the *actual* structures they are intended to denote. We additionally require that filler α_i restricts to the identity along $j_i : \mathbb{c}_i \longrightarrow \mathbb{d}_i$; in particular, the black square in Diagram 3.2·4·2 below commutes on the nose:

$$\begin{tikzcd}
 & \mathbf{T}(\mathbb{c}_i) \arrow[r, "\mathbf{T}(j_i)"] & \mathbf{T}(\mathbb{d}_i) \arrow[r, "\mathbf{T}(\phi_i)"] & \mathbf{T}(\mathscr{S}) \arrow[d, "\mathbf{T}(f)"] \\
\mathbb{c}_i \arrow[ur, "\eta_{\mathbb{c}_i}"] \arrow[r, "j_i"] \arrow[dr, "\phi_i \circ j_i"'] & \mathbb{d}_i \arrow[u, "k_i"'] \arrow[rr, "\mathbf{T}(f\phi_i j_i) \circ k_i"] \arrow[d, "\phi_i", dashed] \arrow[drr, phantom, "\alpha_i"] & & \mathbf{T}(\mathscr{A}) \arrow[d, "a"] \\
 & \mathscr{S} \arrow[rr, "f"'] & & \mathscr{A}
\end{tikzcd} \tag{3.2·4·2}$$

(3.2·5) The purpose of the second condition of **(3.2·4)** governing the 2-cells α_i can be understood in plain English. The finite category $\mathbb{d}_i$ specifies the shape of a "derived arity" that will be realized by certain $\mathbf{T}$-algebra structure; the 2-cell α_i allows the realization of this arity to be off by an isomorphism, but the additional condition that Diagram 3.2·4·2 commute on the nose ensures that the *generators* involved in this derived arity are realized in $\mathscr{A}$ exactly by their interpretation under f.

The balance of strictness and weakness embodied in **(3.2·4)** is very important in practice: it would be absurd to insist that *(e.g.)* some marked cone be realized by the exact choice of limits determined under the algebra $\mathbf{T}(\mathscr{A}) \longrightarrow \mathscr{A}$. But when this marked cone involves a generator, it is perfectly sensible to insist that the realization involve this *exact* generator, and not some isomorph of it.

(3.2·6) Models of a sketch in a particular $\mathbf{T}$-algebra assemble into a category that we will call $\mathbf{Mod}(\mathscr{S}, \mathscr{A})$ whose morphisms are given by natural transformations in $\mathscr{A}$ that commute appropriately commute with the generators of the sketch. We refer the reader to Kinoshita, Power, and Takeyama [KPT99, Definition 4.4] for details.

(3.2·7) The generalization of **(3.1·7)** now states that there exist a functor **Th** which assigns to each sketch $\mathscr{S}$ a **T**-algebra satisfying the following equivalence:

$$\mathrm{Hom}_{\mathbf{T\text{-}Alg}}(\mathbf{Th}(\mathscr{S}),\mathrm{A}) \simeq \mathbf{Mod}(\mathscr{S},\mathrm{A})$$

(3.2·8) In particular, the machinery of sketches applies to the finitary 2-monad for locally Cartesian closed categories. While the fully formal definition of a sketch of any kind is quite technical, at an informal the theory of sketches allows one to freely use *(e.g.)* dependent products and finite limits while describing the generators of a free locally Cartesian closed category.

4 Sketching type theories

(4·1) We now demonstrate how the syntactic categories (categories of judgments) of a variety of type theories may be quickly sketched using this machinery.

(4·2) First, as was mentioned in the previous section the sketch of the "walking type theory" $\mathscr{S}$ can be presented by a category with a single generating morphism $\varpi : \dot{\mathcal{U}} \longrightarrow \mathcal{U}$. The universal property of $\mathbf{Th}(\mathscr{S})$ states that a locally Cartesian closed morphism $\mathscr{S} \longrightarrow \mathscr{E}$ is determined precisely by a morphism $\mathrm{E} \longrightarrow \mathrm{B}$ in $\mathscr{E}$.

4.1 Sketching dependent products

(4.1·1) We now show how one may sketch closure under dependent products. First, we extend $\mathscr{S}$ to the category generated by the following square:

$$\begin{tikzcd}
\dot{\mathcal{U}}^{\Pi} \arrow[r, "\mathsf{lam}"] \arrow[d, "\varpi^{\Pi}"'] & \dot{\mathcal{U}} \arrow[d, "\varpi"] \\
\mathcal{U}^{\Pi} \arrow[r, "\mathsf{pi}"'] & \mathcal{U}
\end{tikzcd} \tag{4.1·1·1}$$

We must carry out *two* separate markings, first to ensure that Diagram 4.1·1·1 is cartesian, and second to cause the left-hand map to be realized by the "generic dependent product family".

(4.1·2) First we will mark Diagram 4.1·1·1 as a pullback square. The generating shape $\mathbb{c}$ is the walking cospan, and the diagram shape $\mathbb{d}$ is the walking square, and the functor $j : \mathbb{c} \longrightarrow \mathbb{d}$ immerses one into the other like so:

$$\begin{tikzcd}
\mathsf{nw} \arrow[r, "\mathsf{n}"] \arrow[d, "\mathsf{w}"'] & j(\mathsf{ne}) \arrow[d, "j(\mathsf{e})"] \\
j(\mathsf{sw}) \arrow[r, "j(\mathsf{s})"'] & j(\mathsf{se})
\end{tikzcd}$$

The diagram $k : \mathbb{d} \longrightarrow \mathbf{T}(\mathbb{c})$ is then the following formal pullback square:

$$\begin{tikzcd}
k(\mathtt{nw}) \arrow[r, dotted, "k(\mathtt{n})"] \arrow[d, dotted, "k(\mathtt{w})"'] \arrow[dr, phantom, "\lrcorner", very near start] & \eta_{\mathbb{c}}(\mathtt{ne}) \arrow[d, "\eta_{\mathbb{c}}(\mathtt{e})"] \\
\eta_{\mathbb{c}}(\mathtt{sw}) \arrow[r, "\eta_{\mathbb{c}}(\mathtt{s})"'] & \eta_{\mathbb{c}}(\mathtt{se})
\end{tikzcd}$$

Finally, the diagram $\phi : \mathbb{d} \longrightarrow \mathscr{S}$ is Diagram 4.1·1·1 itself.

(4.1·3) Next we mark the left-hand map of Diagram 4.1·1·1 to be realized in models by $\mathrm{P}_{\varpi}(\varpi)$ where P_{ϖ} is the polynomial endofunctor of ϖ as explained by Awodey [Awo18b]. In this case the generating shape $\mathbb{c}$ is the walking arrow $\{\mathtt{f} : \mathtt{s} \longrightarrow \mathtt{t}\}$, the diagram shape $\mathbb{d}$ is a (disjoint) pair of arrows, and the functor $j : \mathbb{c} \longrightarrow \mathbb{d}$ identifies $\mathbb{c}$ with the right-hand arrow in $\mathbb{d}$:

$$\begin{tikzcd}
\mathtt{u} \arrow[d, "\mathtt{g}"'] & j(\mathtt{s}) \arrow[d, "j(\mathtt{f})"] \\
\mathtt{v} & j(\mathtt{t})
\end{tikzcd}$$

The diagram $k : \mathbb{d} \longrightarrow \mathbf{T}(\mathbb{c})$ is then the following pair of maps in $\mathbf{T}(\mathbb{c})$:

$$\begin{tikzcd}
\mathrm{P}_{\eta_{\mathbb{c}}(\mathtt{f})}(\eta_{\mathbb{c}}(\mathtt{s})) \arrow[d, dotted, "\mathrm{P}_{\eta_{\mathbb{c}}(\mathtt{f})}(\eta_{\mathbb{c}}(\mathtt{f}))"'] & \eta_{\mathbb{c}}(\mathtt{s}) \arrow[d, "\eta_{\mathbb{c}}(\mathtt{f})"] \\
\mathrm{P}_{\eta_{\mathbb{c}}(\mathtt{f})}(\eta_{\mathbb{c}}(\mathtt{t})) & \eta_{\mathbb{c}}(\mathtt{t})
\end{tikzcd}$$

Finally the diagram $\phi : \mathbb{d} \longrightarrow \mathscr{S}$ is the following pair of maps:

$$\begin{tikzcd}
\dot{\mathcal{U}}^{\Pi} \arrow[d, "\varpi^{\Pi}"'] & \dot{\mathcal{U}} \arrow[d, "\varpi"] \\
\mathcal{U}^{\Pi} & \mathcal{U}
\end{tikzcd}$$

(4.1·4) The universal property of this extended sketch now ensures that a locally Cartesian closed functor $\mathbf{Th}(\mathscr{S}) \longrightarrow \mathscr{E}$ is precisely equivalent to a realization of Diagram 4.1·1·1 as an actual dependent product classification situation in $\mathscr{E}$.

4.2 Sketching non-universal connectives

(4.2·1) Both dependent sums and products are governed by a universal property, which enables terse encodings in both natural models [Awo18a] and the framework exposed here. Types not determined by a universal property, such as the (weak) booleans, are more challenging to encode and cannot be captured by a single cartesian square. We begin by adding the formation and introduction rules for booleans to $\mathscr{S}$.

$$\begin{tikzcd}
\mathbf{1} \arrow[r, "\mathsf{tt},\mathsf{ff}"] \arrow[dr, "\mathsf{bool}"'] & \dot{\mathcal{U}} \arrow[d, "\varpi"] \\
& \mathcal{U}
\end{tikzcd} \tag{4.2·1·1}$$

(4.2·2) First we mark $\mathbf{1}$ to ensure that it is realized by the terminal object; the generating shape $\mathbb{c}$ is the empty category, and the marking shape $\mathbb{d}$ is the terminal category $\{\bullet\}$. We set $k(\bullet) = \mathbf{1}_{\mathbf{T}(\mathbb{c})}$, and set $\phi(\bullet) = \mathbf{1}$.

(4.2·3) In what follows, we will write $[\![\mathrm{X},\mathrm{Y}]\!]$ for the exponential Y^{X} when it exists.

(4.2·4) The purpose of the elimination rule for the booleans is to provide a way to construct elements $f : [\![\varpi[\mathsf{bool}],\dot{\mathcal{U}}]\!]$; because we are not considering a *strict* universal property for the booleans, we will characterize this object only weakly. The elimination rule for the booleans states that there exists a section to the "cartesian gap map" of the following square:

$$\begin{tikzcd}
{[\![\varpi[\mathsf{bool}],\dot{\mathcal{U}}]\!]} \arrow[r, "(\mathsf{tt}^*, \mathsf{ff}^*)"] \arrow[d, "{[\![\varpi[\mathsf{bool}],\varpi]\!]}"'] & \dot{\mathcal{U}} \times \dot{\mathcal{U}} \arrow[d, "\varpi \times \varpi"] \\
{[\![\varpi[\mathsf{bool}],\mathcal{U}]\!]} \arrow[r, "(\mathsf{tt}^*, \mathsf{ff}^*)"'] & \mathcal{U} \times \mathcal{U}
\end{tikzcd} \tag{4.2·4·1}$$

Unfolding into more elementary terms, the eliminator would be the section s below:

$$\begin{tikzcd}
{[\![\varpi[\mathsf{bool}],\dot{\mathcal{U}}]\!]} \arrow[dr, "r"] \arrow[drr, bend left, "(\mathsf{tt}^*, \mathsf{ff}^*)"] \arrow[ddr, bend right, "{[\![\varpi[\mathsf{bool}],\varpi]\!]}"'] & & \\
& \mathsf{Elim} \arrow[r, "q"] \arrow[d, "p"] \arrow[ul, dotted, bend left, "s"] \arrow[dr, phantom, "\lrcorner", very near start] & \dot{\mathcal{U}} \times \dot{\mathcal{U}} \arrow[d, "\varpi \times \varpi"] \\
& {[\![\varpi[\mathsf{bool}],\mathcal{U}]\!]} \arrow[r, "(\mathsf{tt}^*, \mathsf{ff}^*)"'] & \mathcal{U} \times \mathcal{U}
\end{tikzcd} \tag{4.2·4·2}$$

(4.2·5) We may sketch the situation described in **(4.2·4)**, because it involves only pullbacks and exponentials. It will be simpler if we first eliminate the combination of product and pullback from Diagram 4.2·4·2 in terms of a single finite limit; we first add the following objects and morphisms to $\mathscr{S}$ such that s is a section of r:

$$\begin{tikzcd}
 & \dot{\mathscr{U}}_{/\mathsf{bool}} \arrow[dl, "\dot{h}_0"'] \arrow[dr, "\dot{h}_1"] \arrow[d, bend left, "r"] & \\
\dot{\mathscr{U}} \arrow[d, "\varpi"'] & \mathsf{Elim} \arrow[l, "q_0"'] \arrow[r, "q_1"] \arrow[d, "p"] \arrow[u, bend left, "s"] & \dot{\mathscr{U}} \arrow[d, "\varpi"] \\
\mathscr{U} & \mathscr{U}_{/\mathsf{bool}} \arrow[l, "h_0"] \arrow[r, "h_1"'] & \mathscr{U}
\end{tikzcd} \tag{4.2·5·1}$$

Next we will mark Diagram 4.2·5·1, setting the generating shape $\mathbb{c}$ to be the shape of Diagram 4.2·1·1 and setting the marking shape $\mathbb{d}$ to be the disjoint union of $\mathbb{c}$ with the shape of Diagram 4.2·5·1 omitting s. The diagram $j : \mathbb{c} \longrightarrow \mathbb{d}$ is obvious, so we focus on defining $k : \mathbb{d} \longrightarrow \mathbf{T}(\mathbb{c})$; the restriction of k along j is already fixed, so it remains to choose the following subdiagram, writing $\lfloor - \rfloor$ for $\eta_{\mathbb{c}}$:

$$\begin{tikzcd}
 & \llbracket \lfloor \varpi \rfloor [\lfloor \mathsf{bool} \rfloor], \lfloor \dot{\mathscr{U}} \rfloor \rrbracket \arrow[dl, "\lfloor \mathsf{tt} \rfloor^*"'] \arrow[dr, "\lfloor \mathsf{ff} \rfloor^*"] \arrow[d, dotted, "\exists !"] \arrow[dd, bend right=60, "\lfloor \varpi \rfloor_!"'] & \\
\llbracket \lfloor \mathbf{1} \rfloor, \lfloor \dot{\mathscr{U}} \rfloor \rrbracket \arrow[d, "\lfloor \varpi \rfloor_!"'] & \mathrm{L} \arrow[l, dotted] \arrow[r, dotted] \arrow[d, dotted] & \llbracket \lfloor \mathbf{1} \rfloor, \lfloor \dot{\mathscr{U}} \rfloor \rrbracket \arrow[d, "\lfloor \varpi \rfloor_!"] \\
\llbracket \lfloor \mathbf{1} \rfloor, \lfloor \mathscr{U} \rfloor \rrbracket & \llbracket \lfloor \varpi \rfloor [\lfloor \mathsf{bool} \rfloor], \lfloor \mathscr{U} \rfloor \rrbracket \arrow[l, "\lfloor \mathsf{tt} \rfloor^*"] \arrow[r, "\lfloor \mathsf{ff} \rfloor^*"'] & \llbracket \lfloor \mathbf{1} \rfloor, \lfloor \mathscr{U} \rfloor \rrbracket
\end{tikzcd} \tag{4.2·5·2}$$

In Diagram 4.2·5·2 above, we intend the cone under L to be limiting. Finally, we define $\phi : \mathbb{d} \longrightarrow \mathscr{S}$ to be the subdiagram of Diagram 4.2·5·1 omitting the section s.

(4.2·6) The main subtlety of **(4.2·5)** is the discrepancy between $\llbracket \lfloor \mathbf{1} \rfloor, \lfloor \mathrm{X} \rfloor \rrbracket$ and $\lfloor \mathrm{X} \rfloor$; these are not isomorphic at "sketch-time", but they will ultimately be canonically isomorphic in any model of the sketch because we have elsewhere marked $\lfloor \mathbf{1} \rfloor$ as the terminal object. Hence any model of our sketch will support an eliminator of the form exposed in **(4.2·4)**.

(4.2·7) The *strict* booleans can be added to our theory by making s an isomorphism; of course, these booleans will not be realized as a true colimit, but they will *appear* to be so from the perspective of anything classified by $\mathscr{U}$.

4.3 Alternatives to sketching

(4.3·0) While we have emphasized the 2-monadic perspective on LCCCs, there are several alternative ways of presenting locally Cartesian closed categories.

(4.3·1) Recently Bidlingmaier [Bid20] used the machinery of model categories to isolate locally Cartesian closed categories as the fibrant objects of a model category.

The fibrant replacement operation then gives a free LCCC over a given sketch of a type theory. This method also endows **LCCC** with homotopical structure which, for instance, allows one to calculate the pushout of type theories by forming the *homotopy pushout* of their sketches.

(4.3·2) One may also use the (exhaustive) proof of the biequivalence between locally Cartesian closed categories and extensional Martin-Löf type theory [CCD17]. The free LCCC over some set of generators can be formed by taking the syntactic model of extensional type theory generated by this set of constants. This technique allows us to define a type theory by something akin to a signature in a logical framework.

(4.3·3) From here on out, we take for granted that such objects can be constructed in one of several well-understood and equivalent ways, depending on the preferences of the reader for semantic vs. syntactic methods.

5 Adequacy of LCCCs over representable map categories

(5·1) Let Σ be the signature of a type theory in Uemura's logical framework [Uem19]; this generates syntactic representable map category $\mathscr{T}$, but forgetting the difference between representable sorts and ordinary sorts, we also have a syntactic locally Cartesian closed category $\mathscr{E}$ using the results from Section 3.

(5·2) Regard $\mathscr{E}$ as a representable map category equipped with its maximal representable map structure; we have a representable map functor $\ell : \mathscr{T} \longrightarrow \mathscr{E}$, *i.e.* a lex functor that preserves pushforwards along representable maps. We will show that ℓ is full and faithful, using an argument explained by Taylor [Tay99].

5.1 The gluing construction

(5.1·1) The structure map $\ell : \mathscr{T} \longrightarrow \mathscr{E}$ induces a left exact nerve $\mathsf{N} : \mathscr{E} \longrightarrow \mathrm{Pr}(\mathscr{T})$ like so:

$$\mathsf{N}(\mathrm{E}) : \mathrm{X} \longmapsto \mathrm{Hom}_{\mathscr{E}}(\ell(\mathrm{X}),\mathrm{E})$$

(5.1·2) We form the Artin gluing of the nerve $\mathsf{N} : \mathscr{E} \longrightarrow \mathrm{Pr}(\mathscr{T})$ as follows:

$$\begin{tikzcd} \mathscr{G} \arrow[r] \arrow[d, "\mathsf{gl}"'] & \mathrm{Pr}(\mathscr{T}) \arrow[d, "\mathbf{id}_{\mathrm{Pr}(\mathscr{T})}"] \arrow[dl, phantom, "\swarrow"] \\ \mathscr{E} \arrow[r, "\mathsf{N}"'] & \mathrm{Pr}(\mathscr{T}) \end{tikzcd}$$

5.2 The Yoneda model

(5.2·1) We may define a functor $\mathsf{M} : \mathscr{T} \longrightarrow \mathscr{G}$ taking each $\mathrm{X} : \mathscr{T}$ to the computability structure $\mathsf{y}_{\mathscr{T}}(\mathrm{X}) \longrightarrow \mathsf{N}(\ell(\mathrm{X}))$ determined by the identity $\ell(\mathrm{X}) \longrightarrow \ell(\mathrm{X})$ under the following identification:

$$\mathrm{Hom}_{\mathrm{Pr}(\mathscr{T})}(\mathsf{y}_{\mathscr{T}}(\mathrm{X}),\mathsf{N}(\ell(\mathrm{X}))) \cong \mathrm{Hom}_{\mathscr{E}}(\ell(\mathrm{X}),\ell(\mathrm{X}))$$

Recalling the adjunction $\ell_! \dashv \ell^*$ we note that $\mathsf{N}(\ell(\mathrm{X})) \cong \ell^*\ell_!\mathsf{y}(\mathrm{X})$. Hence, we may give an explicit computation of $\mathsf{M}(\mathrm{X})$ as either the unit $\eta_{\mathsf{y}(\mathrm{X})} : \mathsf{y}(\mathrm{X}) \longrightarrow \ell^*\ell_!\mathsf{y}(\mathrm{X})$ or the transpose $\mathbf{id}^\sharp_{\ell_!\mathsf{y}(\mathrm{X})} : \mathsf{y}(\mathrm{X}) \longrightarrow \ell^*\ell_!\mathsf{y}(\mathrm{X})$.

(5.2·2) *The functor* M *is a model of* $\mathscr{T}$. Clearly M preserves finite limits, as these are determined "pointwise" in $\mathscr{G}$ and finite limits are preserved by the Yoneda embedding and $\ell : \mathscr{T} \longrightarrow \mathscr{E}$; it remains to show that M preserves pushforwards along representable maps. We fix a representable map $f : \mathrm{X} \longrightarrow \mathrm{Y}$ and a map $g : \mathrm{Z} \longrightarrow \mathrm{X}$, to show that $\mathsf{M}(f_*g) : \mathsf{y}(f_*g) \longrightarrow \mathsf{N}(\ell(f_*g))$ is the pushforward of $\mathsf{M}(g)$ along $\mathsf{M}(f)$. Accordingly, we will exhibit the following bijection natural in a family $p : \mathrm{A} \longrightarrow \mathrm{Y}$ in $\mathscr{G}$:

$$\mathrm{Hom}_{\mathscr{G}_{/\mathsf{M}(\mathrm{X})}}(p^*\mathsf{M}(f),\mathsf{M}(g)) \cong \mathrm{Hom}_{\mathscr{G}_{/\mathsf{M}(\mathrm{Y})}}(p,\mathsf{M}(f_*g)) \tag{5.2·2·1}$$

We may assume without loss of generality that the domain of A is representable, so we have $\mathrm{A} : \mathsf{y}(\tilde{\mathrm{A}}) \longrightarrow \mathsf{N}(\mathsf{gl}(\mathrm{A}))$. We first construct a map from the right-hand side to the left-hand side of Identity 5.2·2·1, fixing a morphism $\alpha : p \longrightarrow \mathsf{M}(f_*g) : \mathscr{G}_{/\mathsf{M}(\mathrm{Y})}$ that we unfold like so:

$$\begin{tikzcd}
\mathsf{y}(\tilde{\mathrm{A}}) \arrow[rr, "\tilde{\alpha}"] \arrow[dddd, "\mathrm{A}"'] \arrow[dr] & & \mathsf{y}(f_*g) \arrow[dddd, "\mathsf{M}(f_*g)"] \arrow[dl] \\
& \mathsf{y}(\mathrm{Y}) \arrow[d, dash] & \\
& \mathsf{M}(\mathrm{Y}) \arrow[d] & \\
& \mathsf{N}(\ell(\mathrm{Y})) & \\
\mathsf{N}(\mathsf{gl}(\mathrm{A})) \arrow[rr, "\mathsf{N}(\mathsf{gl}(\alpha))"'] \arrow[ur] & & \mathsf{N}(\ell(f_*g)) \arrow[ul]
\end{tikzcd} \tag{5.2·2·2}$$

Both the Yoneda embedding and ℓ preserve pushforwards along representable maps, so $\mathsf{N}(\ell(f_*g)) = \mathsf{N}(\ell(f)_*\ell(g))$ and $\mathsf{y}(f_*g) = \mathsf{y}(f)_*\mathsf{y}(g)$. Accordingly we may transpose $\mathsf{gl}(\alpha)$ and $\tilde{\alpha}$ to obtain commuting triangles in $\mathscr{T}$ and $\mathscr{E}$ respectively, writing $\lambda/\bar{\lambda}$ for the transposition isomorphism of the pushforward:

$$\begin{tikzcd}
\tilde{\mathrm{A}} \times_{\mathrm{Y}} \mathrm{X} \arrow[rr, "\bar{\lambda}(\tilde{\alpha})"] \arrow[dr] & & \mathrm{Z} \arrow[dl] & \mathsf{gl}(\mathrm{A}) \times_{\ell(\mathrm{Y})} \ell(\mathrm{X}) \arrow[rr, "\bar{\lambda}(\mathsf{gl}(\alpha))"] \arrow[dr] & & \mathsf{N}(\mathrm{Z}) \arrow[dl] \\
& \mathsf{y}(\mathrm{X}) & & & \ell(\mathrm{X}) &
\end{tikzcd} \tag{5.2·2·3}$$

It remains to prove that these diagrams assemble into a diagram of the following form, writing $-^\sharp$ for the transpose of the adjunction $\ell_! \dashv \ell^*$:

$$\begin{tikzcd}
\mathsf{y}(\tilde{\mathrm{A}} \times_{\mathrm{Y}} \mathrm{X}) \arrow[r, "\mathsf{y}(\bar{\lambda}(\tilde{\alpha}))"] \arrow[d, "(\mathrm{A},\mathbf{id}_{\ell(\mathrm{X})})^\sharp"'] & \mathsf{y}(\mathrm{Z}) \arrow[d, "\mathbf{id}^\sharp_{\ell(\mathrm{Z})}"] \\
\mathsf{N}(\mathsf{gl}(\mathrm{A}) \times_{\ell(\mathrm{Y})} \ell(\mathrm{X})) \arrow[r, "\mathsf{N}(\bar{\lambda}(\mathsf{gl}(\alpha)))"'] & \mathsf{N}(\ell(\mathrm{Z}))
\end{tikzcd} \tag{5.2·2·4}$$

Unfolding, this is equivalent to proving $\bar{\lambda}(\mathsf{gl}(\alpha)) \circ (\mathrm{A}, \mathbf{id}_{\ell(\mathrm{X})}) = \bar{\lambda}(\tilde{\alpha})$ as morphisms $\ell(\tilde{\mathrm{A}}) \times_{\ell(\mathrm{Y})} \ell(\mathrm{X}) \longrightarrow \ell(\mathrm{Z})$. By the naturality of tranposition, however, this is precisely equivalent to $\mathsf{gl}(\alpha) \circ \mathrm{A} = \tilde{\alpha}$, the content of the square in Diagram 5.2·2·2. Accordingly the bijective map sending $(\mathsf{gl}(\alpha),\tilde{\alpha})$ to $(\bar{\lambda}(\mathsf{gl}(\alpha)),\bar{\lambda}(\tilde{\alpha}))$ restricts to a map between the appropriate hom sets in $\mathscr{G}$. A similar argument shows that the inverse to the transposition restricts to a map of the following type:

$$\mathrm{Hom}_{\mathscr{G}_{/\mathsf{M}(\mathrm{X})}}(p^*\mathsf{M}(f),\mathsf{M}(g)) \longrightarrow \mathrm{Hom}_{\mathscr{G}_{/\mathsf{M}(\mathrm{Y})}}(p,\mathsf{M}(f_*g))$$

Therefore Identity 5.2·2·1 holds and hence M preserves dependent products along representable map products.

(5.2·3) Moreover, $\mathsf{M} : \mathscr{T} \longrightarrow \mathscr{G}$ is fully faithful; to check fullness, we fix a morphism $f : \mathsf{M}(\mathrm{X}) \longrightarrow \mathsf{M}(\mathrm{Y})$, *i.e.* a commuting square in $\mathrm{Pr}(\mathscr{T})$ configured like so:

$$\begin{tikzcd}
\mathsf{y}_{\mathscr{T}}(\mathrm{X}) \arrow[r, "\mathsf{y}_{\mathscr{T}}(\tilde{f})"] \arrow[d, "\mathsf{M}(\mathrm{X})"'] & \mathsf{y}_{\mathscr{T}}(\mathrm{Y}) \arrow[d, "\mathsf{M}(\mathrm{Y})"] \\
\mathsf{N}(\ell(\mathrm{X})) \arrow[r, "\mathsf{N}(\ell(\mathsf{gl}(f)))"'] & \mathsf{N}(\ell(\mathrm{Y}))
\end{tikzcd} \tag{5.2·3·1}$$

It remains to check that $\mathsf{M}(\mathsf{gl}(f))$ is the above square, which is the same as to show that $\ell(\tilde{f}) = \mathsf{gl}(f)$; but this is exactly the content of Diagram 5.2·3·1 instantiated at $\mathbf{id}_{\mathrm{X}}$, hence M is full. Faithfulness is clear, because the underlying $\mathscr{T}$-map can be projected from any such square.

5.3 The conservativity result

(5.3·1) We may now show that $\ell : \mathscr{T} \longrightarrow \mathscr{E}$ is fully faithful. First we observe that there is a locally Cartesian closed functor $\mathsf{L} : \mathscr{E} \longrightarrow \mathscr{G}$ extending the representable map functor $\mathsf{M} : \mathscr{T} \longrightarrow \mathscr{G}$ in the following configuration:

$$\begin{tikzcd}
 & \mathscr{E} \arrow[d, "\mathsf{L}"] & \\
\mathscr{T} \arrow[ur, "\ell"] \arrow[r, hook, "\mathsf{M}"] \arrow[dr, "\ell"'] & \mathscr{G} \arrow[r] \arrow[d, "\mathsf{gl}"] & \mathrm{Pr}(\mathscr{T}) \arrow[d, "\mathbf{id}_{\mathrm{Pr}(\mathscr{T})}"] \arrow[dl, Rightarrow, shorten=4mm] \\
 & \mathscr{E} \arrow[r, "\mathsf{N}"'] & \mathrm{Pr}(\mathscr{T})
\end{tikzcd}$$

(5.3·2) Because the gluing fibration $\mathsf{gl} : \mathscr{G} \longrightarrow \mathscr{E}$ preserves all locally Cartesian closed structure, we have a canonical isomorphism $\mathsf{gl} \circ \mathsf{L} \cong \mathbf{id}_{\mathscr{E}}$; because L is hence a section of the gluing fibration, it is faithful.

(5.3·3) We observe that $\ell : \mathscr{T} \longrightarrow \mathscr{E}$ is faithful using the fact that L is faithful and $\mathsf{M} = \mathsf{L} \circ \ell$ is faithful. Fix $f,g : \mathrm{X} \longrightarrow \mathrm{Y}$ such that $\ell(f) = \ell(g)$; hence $\mathsf{M}(f) = \mathsf{M}(g)$ and because M is faithful, we know that $f = g$. To see that ℓ is full, we fix an arbitrary morphism $f : \ell(\mathrm{X}) \longrightarrow \ell(\mathrm{Y})$; by functoriality we have a morphism $\mathsf{L}(f) : \mathsf{M}(\mathrm{X}) \longrightarrow \mathsf{M}(\mathrm{Y})$, but M is full so this must be the image of some $\mathrm{X} \longrightarrow \mathrm{Y}$ under M.

Acknowledgment

We thank Carlo Angiuli, Steve Awodey, Martin Bidlingmaier, Robert Harper, and Colin Zwanziger for helpful discussions during the production of these notes.

This work was supported in part by AFOSR under grants MURI FA9550-15-1-0053 and FA9550-19-1-0216. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the AFOSR.

References

[AR94] Jiří Adámek and Jiří Rosický. *Locally Presentable and Accessible Categories.* London Mathematical Society Lecture Note Series 189. Cambridge University Press, 1994 (cit. on p. 10).

[Awo18a] Steve Awodey. "A cubical model of homotopy type theory". In: *Annals of Pure and Applied Logic* 169.12 (2018). Logic Colloquium 2015, pp. 1270–1294. ISSN: 0168-0072. DOI: 10.1016/j.apal.2018.08.002 (cit. on p. 14).

[Awo18b] Steve Awodey. "Natural models of homotopy type theory". In: *Mathematical Structures in Computer Science* 28.2 (2018), pp. 241–286. DOI: 10.1017/S0960129516000268 (cit. on pp. 2, 14).

[Bid20] Martin E. Bidlingmaier. *An interpretation of dependent type theory in a model category of locally cartesian closed categories.* 2020. arXiv: 2007.02900 [math.CT] (cit. on p. 16).

[BKP89] R. Blackwell, G. M. Kelly, and A. J. Power. "Two-dimensional monad theory". In: *Journal of Pure and Applied Algebra* 59.1 (1989), pp. 1–41. DOI: 10.1016/0022-4049(89)90160-6 (cit. on pp. 6, 8).

[Car78] John Cartmell. "Generalised Algebraic Theories and Contextual Categories". PhD thesis. Oxford University, Jan. 1978 (cit. on pp. 1, 2, 4).

[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) (cit. on pp. 3, 17).

[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 (cit. on p. 2).

[Ehr66] Charles Ehresmann. *Introduction to the theory of structured categories.* Tech. rep. University of Kansas, 1966 (cit. on p. 10).

[Ehr68a] Charles Ehresmann. "Esquisses et types de structures algébriques". In: *Bul. Inst. Polit.* 14 (1968) (cit. on p. 10).

[Ehr68b] Charles Ehresmann. "Generalized structured categories". In: *Cahiers de Topologie et Géométrie Differential Catégoriques* 10.1 (1968), pp. 139–168 (cit. on p. 10).

[Fio12] Marcelo Fiore. *Discrete generalised polynomial functors*. Slides from talk given at ICALP 2012. 2012. URL: `https://www.cl.cam.ac.uk/~mpf23/talks/ICALP2012.pdf` (cit. on p. 2).

[Gar09] Richard Garner. "On the strength of dependent products in the type theory of Martin-Löf". In: *Annals of Pure and Applied Logic* 160.1 (2009), pp. 1–12. ISSN: 0168-0072. DOI: `j.apal.2008.12.003` (cit. on p. 3).

[Har20] Robert Harper. "A Semantic Logical Framework". Unpublished draft. Dec. 2020 (cit. on p. 4).

[HHP93] Robert Harper, Furio Honsell, and Gordon Plotkin. "A Framework for Defining Logics". In: *J. ACM* 40.1 (Jan. 1993), pp. 143–184. ISSN: 0004-5411. DOI: `10.1145/138027.138060`. URL: `http://doi.acm.org/10.1145/138027.138060` (cit. on pp. 1, 4).

[HL07] Robert Harper and Daniel R. Licata. "Mechanizing Metatheory in a Logical Framework". In: *Journal of Functional Programming* 17.4-5 (July 2007), pp. 613–673. ISSN: 0956-7968. DOI: `10.1017/S0956796807006430` (cit. on pp. 1, 4).

[Jac99] Bart Jacobs. *Categorical Logic and Type Theory*. Studies in Logic and the Foundations of Mathematics 141. Amsterdam: North Holland, 1999 (cit. on p. 1).

[Joy17] Andre Joyal. *Notes on Clans and Tribes*. 2017. arXiv: `1710.10238` (cit. on pp. 2, 3).

[Kel89] G.M. Kelly. "Elementary observations on 2-categorical limits". In: *Bulletin of the Australian Mathematical Society* 39.2 (1989), pp. 301–317. DOI: `10.1017/S0004972700002781` (cit. on p. 8).

[KP93] G.M. Kelly and A.J. Power. "Adjunctions whose counits are coequalizers, and presentations of finitary enriched monads". In: *Journal of Pure and Applied Algebra* 89.1 (1993), pp. 163–179. DOI: `10.1016/0022-4049(93)90092-8` (cit. on p. 6).

[KPT99] Yoshiki Kinoshita, John Power, and Makoto Takeyama. "Sketches". In: *Journal of Pure and Applied Algebra* 143.1 (1999), pp. 275–291. ISSN: 0022-4049. DOI: `10.1016/S0022-4049(98)00114-5` (cit. on pp. 1, 5, 6, 10–12).

[Law63] F. William Lawvere. "Functorial Semantics of Algebraic Theories". PhD thesis. Columbia University, 1963 (cit. on pp. 2, 10).

[Lur09a] Jacob Lurie. *Derived Algebraic Geometry V: Structured Spaces*. 2009. arXiv: `0905.0459`. URL: `https://arxiv.org/abs/0905.0459` (cit. on p. 3).

[Lur09b] Jacob Lurie. *Higher Topos Theory*. Princeton University Press, 2009. ISBN: 978-0-691-14049-0 (cit. on p. 10).

[Mar87] Per Martin-Löf. *The Logic of Judgements*. Workshop on General Logic, Laboratory for Foundations of Computer Science. Feb. 22, 1987 (cit. on pp. 1, 3).

[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 (cit. on p. 3).

[nLa17] nLab. “PIE-limit”. 2017. URL: `https://ncatlab.org/nlab/show/PIE-limit` (cit. on p. 8).

[NPS90] Bengt Nordström, Kent Peterson, and Jan M. Smith. *Programming in Martin-Löf's Type Theory*. Vol. 7. International Series of Monographs on Computer Science. NY: Oxford University Press, 1990 (cit. on pp. 1, 3, 4).

[Pow11] John Power. “Unicity of Enrichment over Cat or Grp”. In: *Applied Categorical Structures* 19 (2011), pp. 293–299 (cit. on p. 5).

[PS99] Frank Pfenning and Carsten Schürmann. “System Description: Twelf — A Meta-Logical Framework for Deductive Systems”. In: *Automated Deduction — CADE-16*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, pp. 202–206. ISBN: 978-3-540-48660-2 (cit. on pp. 1, 4).

[Sch87] Peter Schroeder-Heister. “Structural Frameworks with Higher-level Rules: Philosophical Investigations on the Foundations of Formal Reasoning”. Habilitation. Fachgruppe Philosophie, Universität Konstanz, 1987 (cit. on pp. 1, 3).

[Tay86] Paul Taylor. “Recursive Domains, Indexed Category Theory and Polymorphism”. PhD thesis. University of Cambridge, 1986 (cit. on p. 2).

[Tay99] Paul Taylor. *Practical Foundations of Mathematics*. Cambridge studies in advanced mathematics. Cambridge, New York (N. Y.), Melbourne: Cambridge University Press, 1999. ISBN: 0-521-63107-6 (cit. on pp. 2, 17).

[Uem19] Taichi Uemura. *A General Framework for the Semantics of Type Theory*. 2019. arXiv: `1904.04097`. URL: `https://arxiv.org/abs/1904.04097` (cit. on pp. 1–4, 17).

[Voe15] Vladimir Voevodsky. *A C-system defined by a universe category*. 2015. arXiv: `1409.7925`. URL: `https://arxiv.org/abs/1409.7925` (cit. on p. 2).